SDIO platform support guide

WLAN bring-up on SDIO

About this document

Scope and purpose

This document provides an overview of the Secure Digital Input Output (SDIO) and helps to set Wi-Fi on the SDIO interface conveniently with a host of your choice. It also enables you to configure it based on your application.

Intended audience

This document is primarily intended for those using Infineon® Wi-Fi solutions with a Linux host of their choice. It is recommended that you have prior experience with the SDIO Protocol, Linux kernel networking, or knowledge of the boot flow of a Linux host processor.

Document construction

This document is divided into three sections:

- The first section (Section 1) explains the SDIO protocol that includes the following topics: signal pins structure, operating modes, max bus speed for a particular mode, Command & Response format, SDIO protocol initialization sequence, and the merits and demerits of In-Band & OOB interrupt
- The second section (Sections 2-4) explains WLAN bring-up on the SDIO interface, changes required in the host (DTS File & Kernel), verification of the SDIO card detection, and steps to build and load the FMAC driver.
- The final section (Section 5) includes the Reference section for debugging, which helps in enabling more logs, to compare with successful MMC driver boot logs, and the Init source code flow

Table of contents

1 Introduction to SDIO

The SDIO interface is designed as an extension for the existing SD card standard, to allow connecting different peripherals to the host with the standard SD controller. It is extensively used to connect Wi-Fi/Bluetooth® chips on boards such as NVIDIA Jetson Xavier, i.MX8M Nano , Raspberry Pi 3/4, and so on.

1.1 SDIO signal pins

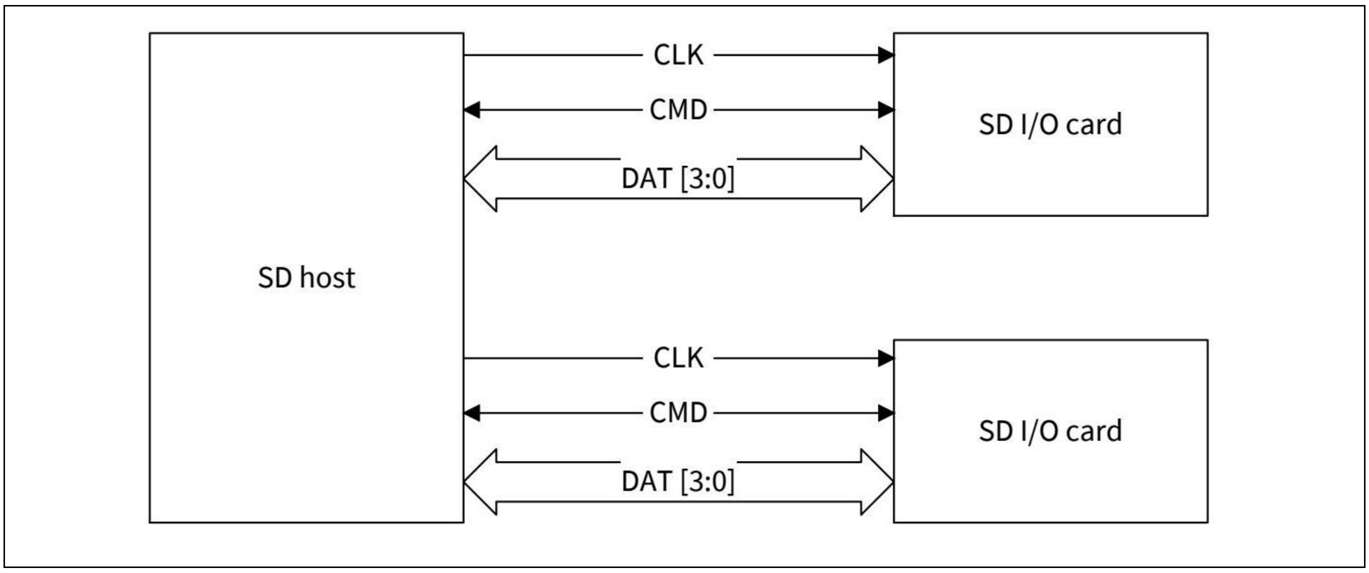


Figure 1 Signal connection to two 4-bit SDIO cards

Table 1 SDIO pin definitions

Pin	SD 4-bit mode		SD 1-bit mode		SPI mode	
	Pin role	**Description**	**Pin role**	**Description**	**Pin role**	**Description**
1	CD/DAT[3]	Data line 3	N/C	Not Used	CS	Card Select
2	CMD	Command line	CMD	Command line	DI	Data input
3	VSS1	Ground	VSS1	Ground	VSS1	Ground
4	VDD	Supply voltage	VDD	Supply voltage	VDD	Supply voltage
5	CLK	Clock	CLK	Clock	CLK	Clock
6	VSS2	Ground	VSS2	Ground	VSS2	Ground
7	DAT[0]	Data line 0	DATA	Data line	DO	Data output
8	DAT[1]	Data line 1 or Interrupt (optional)	IRQ	Interrupt	IRQ	Interrupt
9	DAT[2]	Data line 2 or Read Wait (optional)	RW	Read Wait (optional)	N/C	Not Used

1.2 SDIO bus speeds

Bus speed mode	Max frequency	Signal voltage	Bus max perf	Spec version
Default Speed (DS)	25 MHz	3.3 V	12.5 MBps	1.01
High Speed (HS)	50 MHz	3.3 V	25 MBps	1.10
UHS-I SDR12	25 MHz	1.8 V	12.5 MBps	3.01
UHS-I SDR25	50 MHz	1.8 V	25 MBps	3.01
UHS-I SDR50	100 MHz	1.8 V	50 MBps	3.01
UHS-I SDR104	208 MHz	1.8 V	104 MBps	3.01
UHS-I DDR50	50 MHz	1.8 V	50 MBps	3.01

1.3 SDIO device operating modes

- SPI mode (Mandatory support)
  - In this mode DAT [1] is used as the interrupt pin.
- 1-bit SD mode (Mandatory support)
  - In this mode, data is transferred on the DAT [0] pin only.
  - In this mode DAT [1] is used as the interrupt pin and DAT [2] is used as the read wait pin.
- 4-bit SD mode (Mandatory in UHS mode; Optional in low-speed mode)
  - In this mode, data is transferred on all four data pins (DAT [3:0]).
  - In this mode, the interrupt pin is not available for exclusive use as it is utilized as a data transfer line. Thus, if the interrupt function is required, special timing is required to provide interrupts.

The 4-bit SD mode provides the more data transfer.

1.4 SDIO protocol

Communication over the SD bus is based on command and data bit streams that are initiated by a start bit and terminated by a stop bit.

A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.

Each command token is preceded by a start bit (0) and succeeded by an end bit (1). The total length is 48 bits. Each token is protected by CRC bits so that transmission errors can be detected, and the operation may be repeated.

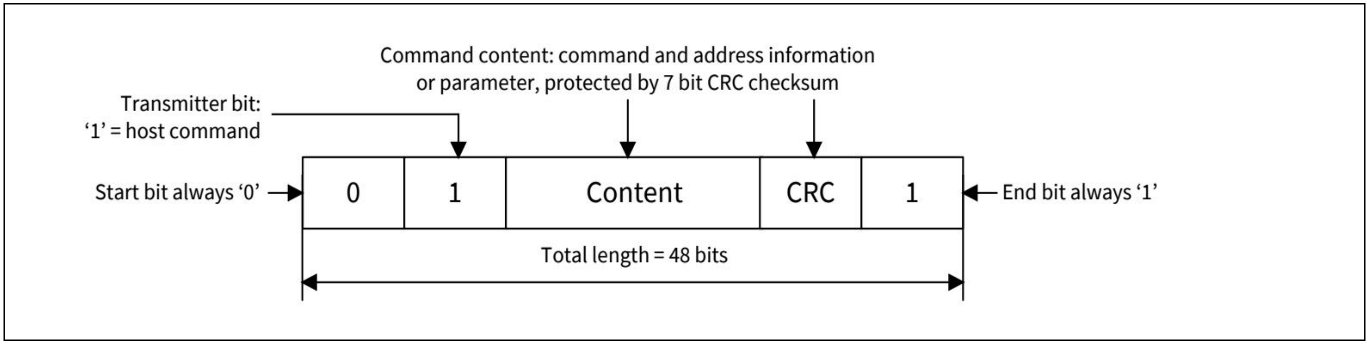

Command content: command and address information or parameter, protected by 7 bit CRC checksum
Transmitter bit: '1' = host command
Start bit always '0'
0
1
Content
CRC
1
End bit always '1'
Total length = 48 bits

Figure 2 **Command format**

Response is a token that is sent from an addressed card, or (synchronously) from all connected cards to the hosts as an answer to a previously received command. A response is transferred serially on the CMD line.

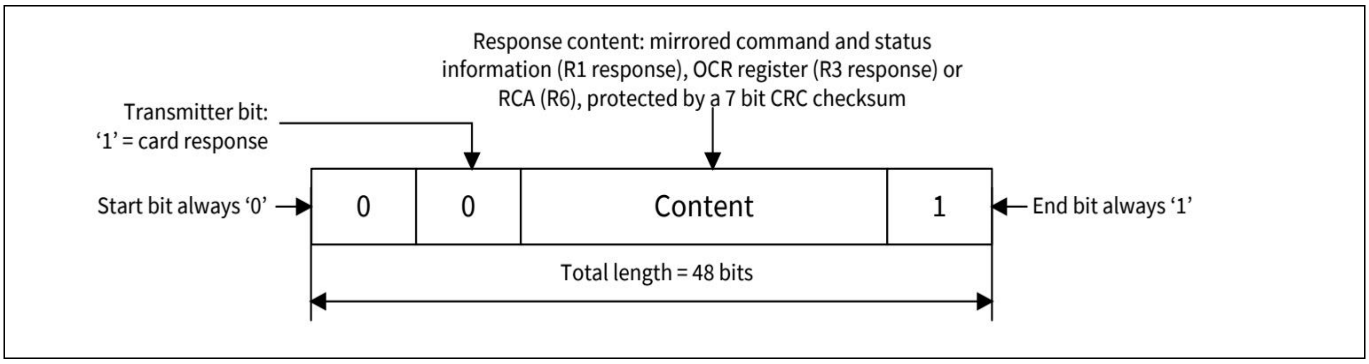

Response content: mirrored command and status information (R1 response), OCR register (R3 response) or RCA (R6), protected by a 7 bit CRC checksum
Transmitter bit: '1' = card response
Start bit always '0'
0
0
Content
1
End bit always '1'
Total length = 48 bits

Figure 3 **Response format**

1.4.1 Commands

CMD 5(IO_SEND_OP_COND)

- The CMD5 for the SDIO card enquires about the I/O card voltage range.
- The normal response to CMD5 must be read from R4.

CMD52(IO_RW_DIRECT)

- This command reads or writes 1 byte using only one command/response pair. A common use is to initialize the register or monitor status values for the I/O function. This command is the fastest means to read or write a single I/O register, as it requires only a single command/response pair.
- Response for CMD52 will be in a R5

CMD53(IO_RW_EXTENDED)

- This data transfer command is used to read or write multiple I/O registers with a single command. It provides the highest possible transfer rate.
- Response to CMD53 must be in a R5.

CMD3(SEND_RELATIVE_ADDR)

- This command is used for asking the card to publish a new Relative Card Address (RCA).
- Response (RCA) for CMD3 must be in R6.

1.4.2 Response

R4 (IO_SEND_OP_COND Response)

The SDIO Card receiving CMD5 will respond with the SDIO unique response, which will have info such as I/O Operation Condition register (OCR), Memory Preset, and number of I/O functions.

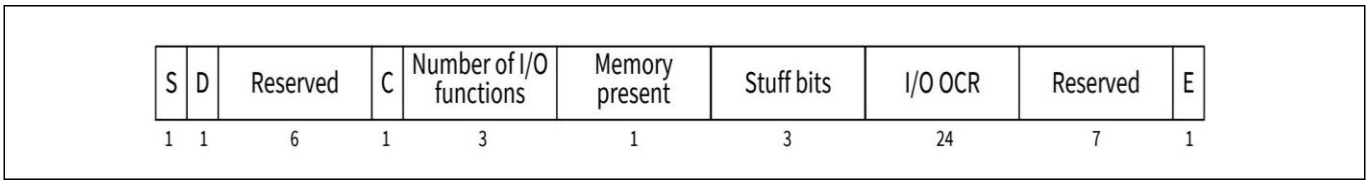


Figure 4 **R4 response format**

R5 (IO_RW_DIRECT Response)

The SDIO card's response to CMD52/CMD53 will have a response Flag (status of the CMD) and Read or Write Data.

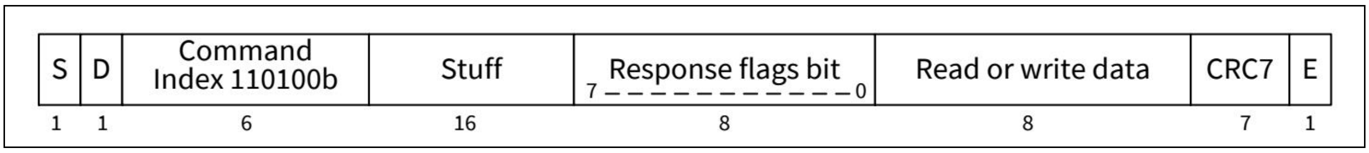
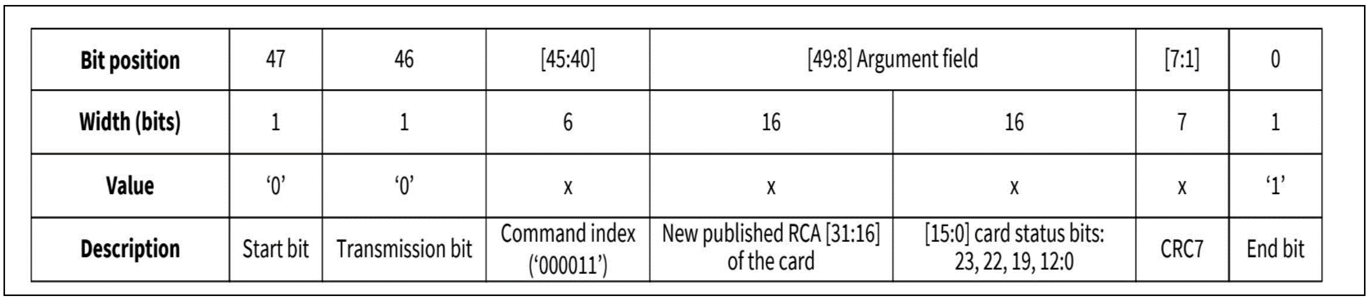


Figure 5 R5 response format

R6 (Published RCA response)

The 16 MSB bits of the argument field are used for the published RCA number.



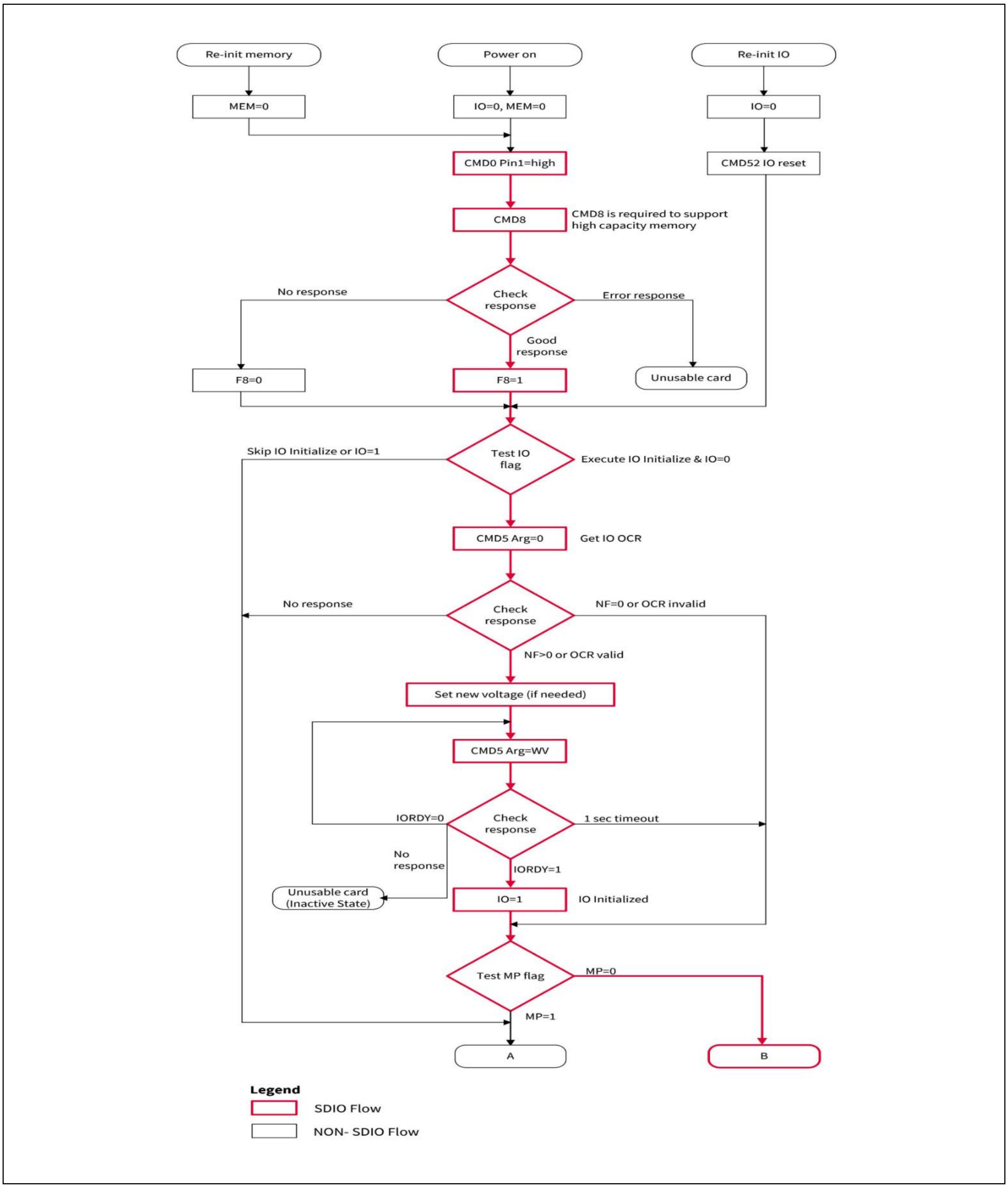
Bit position	47	46	[45:40]	[49:8] Argument field		[7:1]	0
Width (bits)	1	1	6	16	16	7	1
Value	‘0’	‘0’	x	x	x	x	‘1’
Description	Start bit	Transmission bit	Command index (‘000011’)	New published RCA [31:16] of the card	[15:0] card status bits: 23, 22, 19, 12:0	CRC7	End bit

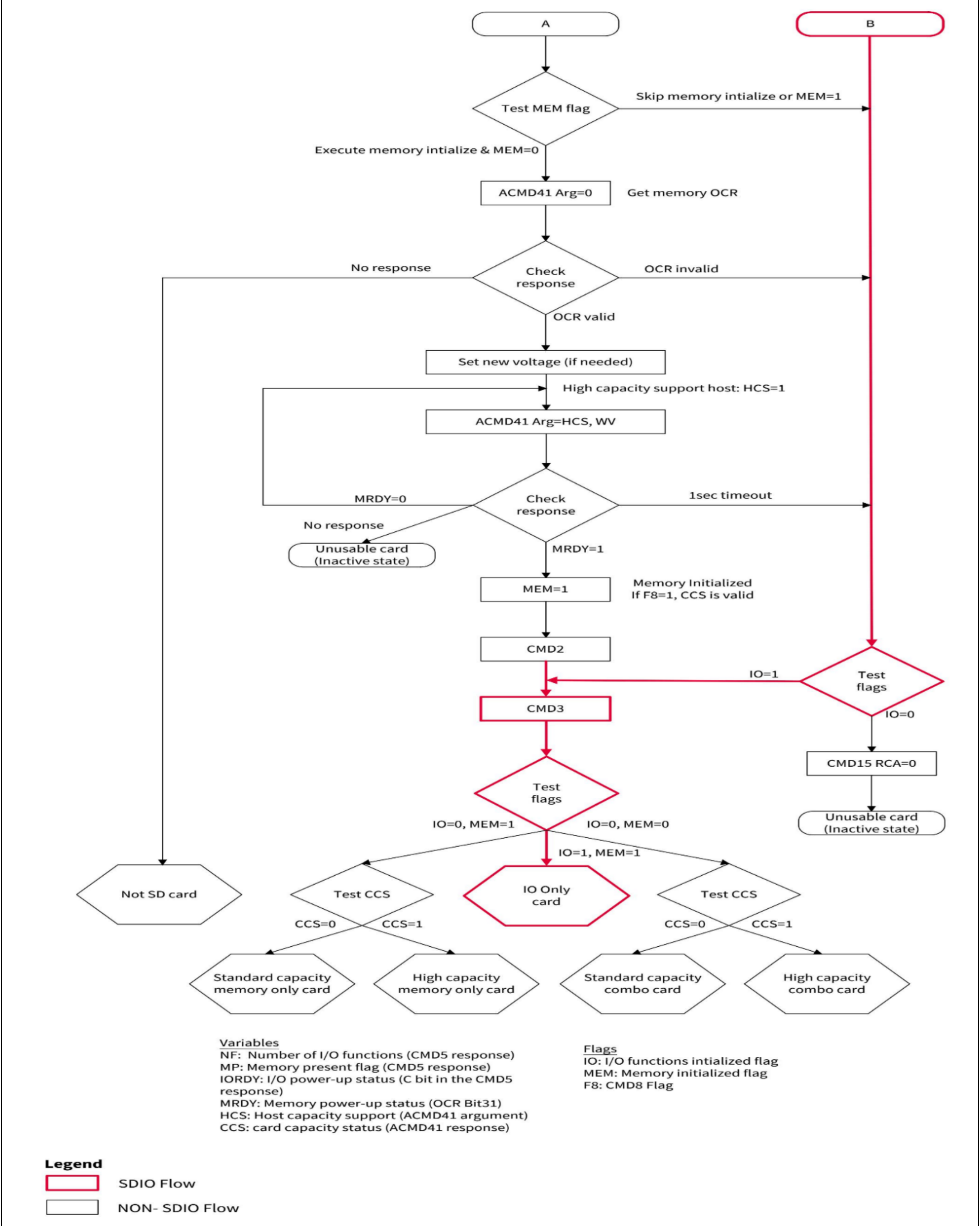
Figure 6 R6 response format

1.4.3 Data

Data can be transferred from the card to the host or vice versa. Data is transferred via the data lines.

1.5 SDIO initialization sequence




A
B
Test MEM flag
Skip memory intialize or MEM=1
Execute memory intialize & MEM=0
ACMD41 Arg=0
Get memory OCR
No response
Check response
OCR invalid
OCR valid
Set new voltage (if needed)
High capacity support host: HCS=1
ACMD41 Arg=HCS, WV
MRDY=0
Check response
1sec timeout
No response
Unusable card (Inactive state)
MRDY=1
MEM=1
Memory Initialized
If F8=1, CCS is valid
CMD2
IO=1
Test flags
IO=0
CMD3
CMD15 RCA=0
Unusable card (Inactive state)
Test flags
IO=0, MEM=1
IO=0, MEM=0
IO=1, MEM=1
Not SD card
Test CCS
IO Only card
Test CCS
CCS=0
CCS=1
CCS=0
CCS=1
Standard capacity memory only card
High capacity memory only card
Standard capacity combo card
High capacity combo card
Variables
NF: Number of I/O functions (CMD5 response)
MP: Memory present flag (CMD5 response)
IORDY: I/O power-up status (C bit in the CMD5 response)
MRDY: Memory power-up status (OCR Bit31)
HCS: Host capacity support (ACMD41 argument)
CCS: card capacity status (ACMD41 response)
Flags
IO: I/O functions intialized flag
MEM: Memory initialized flag
F8: CMD8 Flag
Legend
SDIO Flow
NON- SDIO Flow

SDIO initialization

- An SDIO host sends CMD5 arg = 0 as part of the initialization sequence after either Power ON or CMD52 with I/O set to 0. Thus, it receives a valid Operation Condition Register (OCR) in the R4 response to CMD5 and continues to initialize the card.
- A host that supports UHS-1 sets switching to 1.8 V, Request(S18R) to 1 in the argument of CMD5 to request a change of the signal voltage to 1.8 V. If the card supports UHS-1 and the current signal voltage is 3.3 V, switching to 1.8 V Accepted(S18A) is set to 1 in the R4 response. If the signal is already 1.8 V, the card sets the S18A to 0 so that the host maintains the current signal voltage.
- If the I/O portion of a card has not received CMD5, it remains inactive and does not respond to any commands except CMD5.
- If the I/O host sends a CMD5 to the card, the card responds with R4. The host then reads that R4 value and knows the number of the available I/O functions.
- SDIO card will be detected as the I/O-only card.

1.6 Interrupt mechanism – In-band or OOB

The Wi-Fi device is connected over SDIO to the host processor. There are two ways to route the interrupts from the Wi-Fi device to the host.

- The in-band mechanism uses the SDIO DATA1 line to signal the interrupts.
- The out-of-band (OOB) mechanism requires a dedicated GPIO pin. Make sure that the pin multiplexing is taken care of and the pin is working as a GPIO only.

You can opt for in-band or OOB depending on the application. To achieve the best low-power numbers, it is recommended to use OOB signaling methods. In that mode, the SDIO bus is in a suspended mode unless a triggered interrupt is on the WLAN_HOST_WAKE line when a packet is received.

- If no spare GPIOs are available in the host processor, use the in-band interrupt method where the DATA1 line is repurposed to work as the interrupt. This prevents the bus from being suspended, which increases the power burden.

Based on the availability of the GPIO pin on the host, the appropriate pin can be configured for the OOB.

2 Changes required to bring up WLAN on SDIO

2.1 Device tree source (DTS) file

In Linux, a DTS file refers to a Device Tree Source file. It is used in the Linux kernel to describe the hardware configuration and properties of a device or a system.

- Configuring the GPIO pin for OOB interrupt
  - i.MX8M Nano Host:

The following is the GPIO pin configuration in imx8mm-ea-ucom-kit_v3.dts:

```
interrupt-parent = <&gpio2>;
interrupts = <9 IRQ_TYPE_LEVEL_LOW>;  /* WL_HOST_WAKE = GPIO2_IO09 active low for M.2. */
interrupt-names = "host-wake";
```

NVIDIA Jetson Xavier host:

The following is the GPIO pin configuration in tegra194-comms-p3668.dtsi:

```
interrupts = <TEGRA194_MAIN_GPIO(Q, 6) IRQF_TRIGGER_RISING>; /* gpio = 422 (16*8 + 6 + 288) */
```

Configuring the SDIO bus speed.

i.MX8M Nano Host:

“pinctrl-names” in usdhc1 node is used to configure SDIO bus speed modes.

- “default” is SDR25 at 50 MHz
- “state_100mhz” configures the device in DDR50 at 100 MHz
- "state_200mhz" configures the device in SDR104 at 200 MHz

The following is the SDIO bus configuration for SDR104:

```
pinctrl-names = "state_200mhz”; /* Configuring to SDR104 */
```

Raspberry Pi host:

Make the following changes in the */boot/config.txt* file.

sdio_overclock is used to configure the SDIO bus speed.

```
dtoverlay=sdio, sdio_overclock=<val>
```

Changes required to bring up WLAN on SDIO

The following is the SDIO bus configuration for SDR25:

```
sdio_overclock=50
```

NVIDIA Jetson Xavier host:

uhs-mask is used to mask a particular SDIO speed mode and the following are the masking values.

- Mask HS200 mode: 0x20
- Mask HS400 mode: 0x40
- Mask SDR104 mode: 0x10
- Mask SDR50 mode: 0x4

The following is the SDIO bus configuration for SDR104

```
uhs-mask = <0x08>; /* Configuring to SDR104 */
```

The complete change for SDIO bus speed and interrupt configuration on i.MX8M Nano is as follows: (imx8mm-ea-ucom-kit_v3.dts).

```
/* M.2 connector */
&usdhc1 {
#address-cells = <1>;
#size-cells = <0>;
pinctrl-names = "default", "state_100mhz","state_200mhz";
pinctrl-0 = <&pinctrl_usdhc1>, <&pinctrl_usdhc1_gpio>;
pinctrl-1 = <&pinctrl_usdhc1_100mhz>, <&pinctrl_usdhc1_gpio>;
pinctrl-2 = <&pinctrl_usdhc1_200mhz>, <&pinctrl_usdhc1_gpio>;
keep-power-in-suspend;
non-removable;
pm-ignore-notify;
cap-power-off;
mmc-pwrseq = <&usdhc1_pwrseq>;
status = "okay";
brcmf: bcrmf@1 {
reg = <1>;
compatible = "brcm,bcm4329-fmac";
interrupt-parent = <&gpio2>;
interrupts = <9 IRQ_TYPE_LEVEL_LOW>;  /* WL_HOST_WAKE = GPIO2_IO09 active
low for M.2. */
interrupt-names = "host-wake";
};
};
```

The complete change for SDIO bus speed and interrupt configuration on the NVIDIA Jetson Xavier board is as follows: (tegra194-comms-p3668.dtsi).

```
/*
 * Common include DTS file for CVM:P3668-0001 and CVB:P3449-0000 variants.
 *
 * Copyright (c) 2019, NVIDIA CORPORATION.  All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; version 2 of the License.
 *
 * This program is distributed in the hope that it will be useful, but
WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
for
 * more details.
 */

#include "dt-bindings/gpio/tegra194-gpio.h"

/ {
    wifi: wifi@1 {
            compatible = "brcm,bcm4329-fmac";
            wlreg_on-supply = <&wlreg_on>;
    };
};

&sdmmc3 {
    status = "okay";
    only-1-8-v;
    uhs-mask = <0x08>; /* SDR104 */
    wifi-host;
    nvidia,disable-rtpm;
     #address-cells = <1>;
    #size-cells = <0>;
    no-sd;
    pm-ignore-notify;
    keep-power-in-suspend;
    brcmfmac: bcrmfmac@1 {
        reg = <1>;
        compatible = "brcm,bcm4329-fmac";
        interrupt-parent = <&tegra_main_gpio>;
        interrupts = <TEGRA194_MAIN_GPIO(Q, 6) IRQF_TRIGGER_RISING>; /*
gpio = 422 (16*8 + 6 + 288) */
        interrupt-names = "host-wake";
    };
};
```

The voltage-level configuration on the NVIDIA Jetson Xavier board is as follows:

```
regulator-min-microvolt = <1800000>;
regulator-max-microvolt = <1800000>;
```

The complete changes for voltage configuration on the NVIDIA Jetson Xavier board are as follows:

```
diff --git a/common/tegra194-fixed-regulator-p3509-0000-a00.dtsi
b/common/tegra194-fixed-regulator-p3509-0000-a00.dtsi
index 2a804b9..d4ecc42 100644
--- a/common/tegra194-fixed-regulator-p3509-0000-a00.dtsi
+++ b/common/tegra194-fixed-regulator-p3509-0000-a00.dtsi
@@ -16,6 +16,17 @@

 / {
      fixed-regulators {
+            //Fake regulator provides WL_REG_ON signal for wireless
interface
+            wlreg_on: regulator@140 {
+                  compatible = "regulator-fixed";
+                  regulator-min-microvolt = <1800000>;
+                  regulator-max-microvolt = <1800000>;
+                  regulator-name = "wlreg_on";
+                  //gpio = <TEGRA194_AON_GPIO(CC, 2) GPIO_ACTIVE_LOW>;
+                  enable-active-high;
+                  startup-delay-us = <100>;
+            };
+
             hdr40_vdd_3v3: p3509_vdd_3v3_cvb: regulator@101 {
                    compatible = "regulator-fixed";
                    reg = <101>;
diff --git a/common/tegra194-p3668-common.dtsi b/common/tegra194-p3668-
common.dtsi
index 36ebc4d..af643e1 100644
--- a/common/tegra194-p3668-common.dtsi
+++ b/common/tegra194-p3668-common.dtsi
@@ -24,6 +24,7 @@
 #include "tegra194-power-tree-p3668.dtsi"
 #include "tegra194-thermal-p3668.dtsi"
 #include <t19x-common-platforms/tegra194-no-pll-aon-clock.dtsi>
+#include "tegra194-comms-p3668.dtsi"

 / {
        nvidia,fastboot-usb-vid = <0x0955>;

diff --git a/common/tegra194-power-tree-p3668.dtsi b/common/tegra194-power-
tree-p3668.dtsi
index 8043234..ee8ad35 100644
--- a/common/tegra194-power-tree-p3668.dtsi
+++ b/common/tegra194-power-tree-p3668.dtsi
@@ -27,6 +27,10 @@
                vmmc-supply = <&p3668_vdd_sdmmc1_sw>;
        };

+       sdhci@3440000 {
+               vmmc-supply = <&wlreg_on>;
+       };
+
        ether_qos@2490000 {
                vddio_sys_enet_bias-supply = <&battery_reg>;
                vddio_enet-supply = <&battery_reg>;
```

The above changes must be done on the appropriate dtsi file of the host platform.

2.2 Kernel

- If the host kernel version is less than 4.11 and the host platform has not added UHS-I mode support, these code changes are required to set SDIO in UHS-I mode. The kernel version is greater than 4.11 by default to support UHS-1 mode.
- Changes are made to switch the voltage level to 1.8 V.

```
diff --git a/drivers/mmc/core/core.c b/drivers/mmc/core/core.c
index bbf1505..e4ac0a6 100644
--- a/drivers/mmc/core/core.c
+++ b/drivers/mmc/core/core.c
@@ -1997,6 +1997,94 @@ int mmc_set_signal_voltage(struct mmc_host *host,
int signal_voltage, u32 ocr)
    return err;
 }

+int mmc_host_set_uhs_voltage(struct mmc_host *host, u32 ocr)
+{
+   u32 clock;
+
+   /*
+     * During a signal voltage level switch, the clock must be gated
+     * for 5 ms according to the SD spec
+     */
+   clock = host->ios.clock;
+   host->ios.clock = 0;
+   mmc_set_ios(host);
+
+   if (mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_180, ocr))
+       return -EAGAIN;
+
+   /* Keep clock gated for at least 10 ms, though spec only says 5 ms */
+   mmc_delay(10);
+   host->ios.clock = clock;
+   mmc_set_ios(host);
+
+   return 0;
+}
+
+int mmc_set_uhs_voltage(struct mmc_host *host, u32 ocr)
+{
+   struct mmc_command cmd = {};
+   int err = 0;
+
+   /*
+     * If we cannot switch voltages, return failure so the caller
+     * can continue without UHS mode
```

Changes required to bring up WLAN on SDIO

```
+     */
+    if (host->ops->start_signal_voltage_switch)
+        return -EPERM;
+    if (!host->ops->card_busy)
+        pr_warn("%s: cannot verify signal voltage switch\n",
+            mmc_hostname(host));
+
+    cmd.opcode = SD_SWITCH_VOLTAGE;
+    cmd.arg = 0;
+    cmd.flags = MMC_RSP_R1 | MMC_CMD_AC;
+
+    err = mmc_wait_for_cmd(host, &cmd, 0);
+    if (err)
+        goto power_cycle;
+
+    if (!mmc_host_is_spi(host) && (cmd.resp[0] & R1_ERROR))
+        return -EIO;
+
+    /*
+     * The card should drive cmd and dat[0:3] low immediately
+     * after the response of cmd11, but wait 1 ms to be sure
+     */
+    mmc_delay(1);
+    if (host->ops->card_busy && host->ops->card_busy(host)) {
+        err = -EAGAIN;
+        goto power_cycle;
+    }
+
+    if (mmc_host_set_uhs_voltage(host, ocr)) {
+        /*
+         * Voltages may not have been switched, but we have already
+         * sent CMD11, so a power cycle is required anyway
+         */
+        err = -EAGAIN;
+        goto power_cycle;
+    }
+
+    /* Wait for at least 1 ms according to spec */
+    mmc_delay(1);
+
```

Changes required to bring up WLAN on SDIO

```
+    /*
+     * Failure to switch is indicated by the card holding
+     * dat[0:3] low
+     */
+    if (host->ops->card_busy && host->ops->card_busy(host))
+        err = -EAGAIN;
+
+power_cycle:
+    if (err) {
+        pr_debug("%s: Signal voltage switch failed, "
+            "power cycling card\n", mmc_hostname(host));
+        mmc_power_cycle(host, ocr);
+    }
+
+    return err;
+}
+
 /*
  * Select timing parameters for host.
  */
diff --git a/drivers/mmc/core/core.h b/drivers/mmc/core/core.h
index c8f5172..b3fe27d 100644
--- a/drivers/mmc/core/core.h
+++ b/drivers/mmc/core/core.h
@@ -104,5 +104,6 @@ static inline void mmc_register_pm_notifier(struct
mmc_host *host) { }
 static inline void mmc_unregister_pm_notifier(struct mmc_host *host) { }
 #endif

+int mmc_set_uhs_voltage(struct mmc_host *host, u32 ocr);
 #endif

diff --git a/drivers/mmc/core/sdio.c b/drivers/mmc/core/sdio.c
index a2be7a3..333d519 100644
--- a/drivers/mmc/core/sdio.c
+++ b/drivers/mmc/core/sdio.c
@@ -636,9 +636,15 @@ static int mmc_sdio_init_card(struct mmc_host *host,
u32 ocr,
      * systems that claim 1.8v signalling in fact do not support
      * it.
      */
```

Changes required to bring up WLAN on SDIO

```
-   if (!powered_resume && (rocr & ocr & R4_18V_PRESENT)) {
-       err = mmc_set_signal_voltage(host, MMC_SIGNAL_VOLTAGE_180,
-                   ocr_card);
+   if (rocr & ocr & R4_18V_PRESENT) {
+       pr_err("mmc_sdio_init_card: powered_resume for Index:
%d!!!!!\n",host->index);
+       if (host->index == 0) {
+           pr_err("mmc_sdio_init_card: Skipping 1.8 V setting for Index:
%d!!!!!\n",host->index);
+           err = 0;
+       } else {
+           pr_err("mmc_sdio_init_card: Setting 1.8 V for Index:
%d!!!!!\n",host->index);
+           err = mmc_set_uhs_voltage(host, ocr_card);
+       }
        if (err == -EAGAIN) {
            sdio_reset(host);
            mmc_go_idle(host);
@@ -1282,3 +1288,15 @@ int sdio_reset_comm(struct mmc_card *card)
    return err;
 }
 EXPORT_SYMBOL(sdio_reset_comm);
+
+void mmc_sdio_force_remove(struct mmc_host *host)
+{
+        mmc_sdio_remove(host);
+
+        mmc_claim_host(host);
+        mmc_detach_bus(host);
+        mmc_power_off(host);
+        mmc_release_host(host);
+}
+EXPORT_SYMBOL_GPL(mmc_sdio_force_remove);
+
diff --git a/drivers/mmc/host/sdhci-tegra.c b/drivers/mmc/host/sdhci-
tegra.c
index c7a6f85..94817f0 100644
--- a/drivers/mmc/host/sdhci-tegra.c
+++ b/drivers/mmc/host/sdhci-tegra.c
@@ -2212,6 +2212,7 @@ static void sdhci_delayed_detect(struct work_struct
*work)
        pm_runtime_disable(mmc_dev(host->mmc));
 }

+static struct mmc_host *wifi_mmc_host;
 static int sdhci_tegra_probe(struct platform_device *pdev)
 {
    const struct of_device_id *match;
@@ -2222,6 +2223,7 @@ static int sdhci_tegra_probe(struct platform_device
*pdev)
    struct clk *clk;
    struct sdhci_tegra_clk_src_data *clk_src_data;
    int rc;
+   struct device_node *np = pdev->dev.of_node;

    match = of_match_device(sdhci_tegra_dt_match, &pdev->dev);
    if (!match)
@@ -2414,6 +2416,11 @@ static int sdhci_tegra_probe(struct platform_device
*pdev)
    if (tegra_host->en_periodic_calib)
        host->quirks2 |= SDHCI_QUIRK2_PERIODIC_CALIBRATION;

+   if (of_get_property(np, "wifi-host", NULL)) {
+               wifi_mmc_host = host->mmc;
+               dev_info(mmc_dev(host->mmc), "assigned as wifi host\n");
+        }
+
    schedule_delayed_work(&tegra_host->detect_delay,
                 msecs_to_jiffies(tegra_host->boot_detect_delay));
    return 0;
@@ -2538,6 +2545,19 @@ static void tegra_sdhci_post_resume(struct
sdhci_host *host)
        tegra_sdhci_post_init(host);
 }

+void mmc_sdio_force_remove(struct mmc_host *host);
+void wifi_card_detect(bool on)
+{
+        WARN_ON(!wifi_mmc_host);
+        if (on) {
+               mmc_detect_change(wifi_mmc_host, 0);
+        } else {
+               if (wifi_mmc_host->card)
```

Changes required to bring up WLAN on SDIO

```
+                               mmc_sdio_force_remove(wifi_mmc_host);
+           }
+}
+EXPORT_SYMBOL_GPL(wifi_card_detect);
+
 static int sdhci_tegra_card_detect(struct sdhci_host *host, bool req)
 {
    struct sdhci_pltfm_host *pltfm_host = sdhci_priv(host);
```

3 Verify SDIO detection

On booting, the kernel can detect the SD card as an SDIO with the mode in which it is configured at the appropriate MMC slot. The following logs confirm the SDIO card detection.

For example, SDIO is configured for SDR50.

```
mmc0: new ultra high speed SDR50 SDIO card at address 0001
```

More details about SDIO configuration are available at*: /sys/kernel/debug/mmc0/ios*

```
ifx@ifxhost:~$ sudo cat /sys/kernel/debug/mmc0/ios
clock: 100000000 Hz
actual clock: 97625098 Hz
vdd: 7 (1.65 - 1.95 V)
bus mode: 2 (push-pull)
chip select: 0 (don't care)
power mode: 2 (on)
bus width: 2 (4 bits)
timing spec: 5 (sd uhs SDR50)
signal voltage: 1 (1.80 V)
driver type: 0 (driver type B)
```

4 FMAC compile and load

4.1 FMAC compile

This section explains how to compile the FMAC backport driver sources.

Contact your local Infineon® Technologies distribution channel (FAE or local sales representative) or community page to get Infineon®'s FMAC backport driver source files. Refer to the Wi-Fi software user guide and README documents available in the FMAC Release package for build and debugging.

1. Decompress the FMAC backport source package.

```
Ex: $ tar xvzf cypress-backports-v5.10.9-2022_0909-module-
src.tar.gz
```

2. Set the path to the device kernel headers.

```
Ex(Nvidia kernel): $ export MY_KERNEL=/usr/src/linux-headers-
4.9.253-tegra-ubuntu18.04_aarch64/kernel-4.9/
```

3. Generate a configuration file (.config) from the FMAC driver configuration specified in defconfigs/brcmfmac.

```
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL defconfig-brcmfmac
```

4. Compile and generate kernel modules.

```
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL modules
```

5. The kernel modules are available at the following paths.

```
./compat/compat.ko
./net/wireless/cfg80211.ko
./drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
./drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko
```

4.2 Load the FMAC driver and firmware

Use the following commands to load the WLAN FMAC driver and firmware.

```
sudo insmod compat.ko
sudo insmod cfg80211.ko
sudo insmod brcmutil.ko
sudo insmod brcmfmac.ko
```

Refer to the Wi-Fi software user guide and README documents available in the FMAC release package for more information on building and loading the FMAC.

5 Debugging and reference logs

5.1 Kernel debug flags

Enable the following debug configs in the kernel config file and build the kernel:

- CONFIG_MMC_DEBUG
- CONFIG_DEBUG_FS
- CONFIG_BRCMDBG
- CONFIG_PRINTK

5.2 MMC driver boot logs

Reference logs for debugging the SDIO detection issues.

```
[    2.402000] sdhci: Secure Digital Host Controller Interface driver
[    2.408191] sdhci: Copyright(c) Pierre Ossman
[    2.412601] sdhci-pltfm: SDHCI platform and OF driver helper
[   2.418999] sdhci_pltfm_init
[   2.422514] sdhci_alloc_host
[    2.429422] sdhci: =========== REGISTER DUMP (mmc1) ===========
[    2.435323] sdhci: Sys addr: 0x00000000 | Version:  0x00000002
[    2.441163] sdhci: Blk size: 0x00000000 | Blk cnt:  0x00000001
[    2.447026] sdhci: Argument: 0x00000000 | Trn mode: 0x00000000
[    2.452884] sdhci: Present:  0x01f88088 | Host ctl: 0x00000000
[    2.458723] sdhci: Power:    0x00000000 | Blk gap:  0x00000080
[    2.464578] sdhci: Wake-up:  0x00000008 | Clock:    0x0000800f
[    2.470416] sdhci: Timeout:  0x00000080 | Int stat: 0x00000000
[    2.476269] sdhci: Int enab: 0x007f1003 | Sig enab: 0x007f1003
[    2.482126] sdhci: AC12 err: 0x00000000 | Slot int: 0x00000302
[    2.487962] sdhci: Caps:     0x07eb0000 | Caps_1:   0x0000b407
[    2.493816] sdhci: Cmd:      0x00000000 | Max curr: 0x00ffffff
[    2.499651] sdhci: Host ctl2: 0x00000000
[    2.503594] sdhci: ADMA Err: 0x00000000 | ADMA Ptr: 0x00000000
[    2.509428] sdhci: ===========================================
[    2.571499] mmc_rescan
[    2.574395] mmc1: mmc_rescan_try_freq: trying to init card at 400000 Hz
[    2.581012] mmc1: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    2.589314] mmc1: SDHCI controller on 2194000.usdhc [2194000.usdhc]
using ADMA
[    2.590400] sdhci_pltfm_init
[    2.590466] sdhci_alloc_host
[    2.590602] sdhci-esdhc-imx 2198000.usdhc: Got WP GPIO
[    2.590678] sdhci-esdhc-imx 2198000.usdhc: allocated mmc-pwrseq
[    2.594113] sdhci: =========== REGISTER DUMP (mmc2) ===========
[    2.594120] sdhci: Sys addr: 0x00000000 | Version:  0x00000002
[    2.594123] sdhci: Blk size: 0x00000000 | Blk cnt:  0x00000001
[    2.594127] sdhci: Argument: 0x00000000 | Trn mode: 0x00000000
[    2.594131] sdhci: Present:  0x01f88088 | Host ctl: 0x00000000
[    2.594135] sdhci: Power:    0x00000000 | Blk gap:  0x00000080
[    2.594138] sdhci: Wake-up:  0x00000008 | Clock:    0x0000800f
[    2.594142] sdhci: Timeout:  0x00000080 | Int stat: 0x00000000
[    2.594146] sdhci: Int enab: 0x007f1003 | Sig enab: 0x007f1003
[    2.594150] sdhci: AC12 err: 0x00000000 | Slot int: 0x00000302
[    2.594154] sdhci: Caps:     0x07eb0000 | Caps_1:   0x0000b407
[    2.594157] sdhci: Cmd:      0x00000000 | Max curr: 0x00ffffff
[    2.594160] sdhci: Host ctl2: 0x00000000
[    2.594163] sdhci: ADMA Err: 0x00000000 | ADMA Ptr: 0x00000000
[    2.594165] sdhci: ===========================================
[    2.662727] mmc2: SDHCI controller on 2198000.usdhc [2198000.usdhc]
using ADMA
[    2.663195] sdhci_pltfm_init
[    2.663258] sdhci_alloc_host
[    2.663366] sdhci-esdhc-imx 219c000.usdhc: could not get ultra high
speed state, work on normal mode
[    2.663394] sdhci-esdhc-imx 219c000.usdhc: Got CD GPIO
[    2.663409] sdhci-esdhc-imx 219c000.usdhc: Got WP GPIO
[    2.676496] sdhci: =========== REGISTER DUMP (mmc3) ===========
[    2.676503] sdhci: Sys addr: 0x00000000 | Version:  0x00000002
[    2.676507] sdhci: Blk size: 0x00000200 | Blk cnt:  0x00000001
[    2.676511] sdhci: Argument: 0x00007e61 | Trn mode: 0x00000000
[    2.676514] sdhci: Present:  0x01f88088 | Host ctl: 0x00000002
[    2.676518] sdhci: Power:    0x00000000 | Blk gap:  0x00000080
[    2.676522] sdhci: Wake-up:  0x00000008 | Clock:    0x0000011f
[    2.676525] sdhci: Timeout:  0x0000008b | Int stat: 0x00000000
[    2.676529] sdhci: Int enab: 0x007f1003 | Sig enab: 0x007f1003
[    2.676532] sdhci: AC12 err: 0x00000000 | Slot int: 0x00000302
[    2.676536] sdhci: Caps:     0x07eb0000 | Caps_1:   0x0000b407
[    2.676540] sdhci: Cmd:      0x0000113a | Max curr: 0x00ffffff
[    2.676542] sdhci: Host ctl2: 0x00000000
[    2.676546] sdhci: ADMA Err: 0x00000000 | ADMA Ptr: 0x00000000
[    2.676548] sdhci: ===========================================
[    2.751732] mmc3: SDHCI controller on 219c000.usdhc [219c000.usdhc]
using ADMA
```

```
[    2.754128] galcore: clk_get 2d core clock failed, disable 2d/vg!
[    2.754348] Galcore version 6.2.2.93313
[    2.954162] mmc1: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    2.971521] mmc_rescan
[    2.978853] mmc2: mmc_rescan_try_freq: trying to init card at 400000 Hz
[    2.997160] mmc2: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    3.012982] mmc1: mmc_rescan_try_freq: trying to init card at 300000 Hz
[    3.052623] mmc2: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    3.060056] mmc_rescan
[    3.063026] mmc3: mmc_rescan_try_freq: trying to init card at 400000 Hz
[    3.069646] mmc3: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    3.077949] mmc1: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    3.091466] mmc1: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    3.091491] mmc2: mmc_select_voltage with ocr:30ffff00
[    3.091496] mmc_select_voltage ocr:30ffff00
[    3.091499] mmc2: selected voltage:00040000
[    3.091504] mmc2: mmc_sdio_init_card: send CMD5 with Arg=<ocr voltage>
ocr:00040000
[    3.132346] mmc1: mmc_rescan_try_freq: trying to init card at 200000 Hz
[    3.150800] mmc2: new high speed SDIO card at address 0001
[    3.191470] mmc1: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    3.220161] mmc1: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    3.260454] mmc1: mmc_rescan_try_freq: trying to init card at 100000 Hz
[    3.311459] mmc3: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    3.321465] mmc1: mmc_rescan_try_freq: sdio_reset-> CMD52 performing
sdio card reset
[    3.337293] mmc_select_voltage ocr:40ff8000
[    3.373357] mmc1: mmc_attach_sdio: sending CMD5 with arg=0 to get OCR
info
[    3.406702] mmc3: new high speed SDHC card at address aaaa
[    3.451528] mmcblk3: mmc3:aaaa SL08G 7.40 GiB
[    3.457199]  mmcblk3: p1 p2
```

5.3 init sequence code flow

Module: <kernel_src>/drivers/mmc/

```
sdhci-pltfm.c
→  sdhci_pltfm_register(..)
  →  sdhci_pltfm_init(..)
      →  sdhci_alloc_host(..)
         →  mmc_alloc_host(..)
             →  mmc_rescan
 →  mmc_rescan_try_freq(..)
     →  sdio_reset(..)                                                    →
Sending CMD52
        →  mmc_attach_sdio(..)
             →  mmc_send_io_op_cond(..)                         →  Sending
CMD5 with Arg=0 to   get OCR (Operation conditions register)
                  →  mmc_select_voltage(..)
                      →  mmc_sdio_init_card(..)
                           →  mmc_send_io_op_cond(..)             →  Sending
CMD5 with Arg=<selected voltage from OCR info>. SDIO is initialized.
                                  →  [Reads OCR information further]:
                                                    *Check if MEMORY
capable card
                                                    *Check if COMBO card
(SD Memory + SDIO)
                                                    *Check if UHS-I mode
supported, then enable 1.8 V signaling.
                                      →  [Reads CCCR Register]
                                         →  [Reads CIS tuples]
                                            →  Check and enable the card to
High speed or UHS based on voltage & signaling supported by the card.
                                                →  [Reads OCR to get
Number of IO Functions]
                                                       →  Init functions
and allocate memory
                                                             →  mmc_add_ca
rd(..)        →  Add the MMC card to the driver model.
                                                                    →  mmc
_add_card_debugfs(..)      →  Add a debug filesystem if CONFIG_DEBUG_FS is
enabled.
                                                                          →
device_add(..)                            →  Add the MMC device; Create
filesystem; Notifies platform and filesystem about the addition of new
device.
```

5.4 SDIO bus analyzer

The PGY-SSM SD/SDIO/eMMC Protocol Analyzer has multiple features to capture and debug communication between the host and WI-FI card. This protocol analyzer supports SD, SDIO, and eMMC for data rates up to 200-MHz DDR mode.

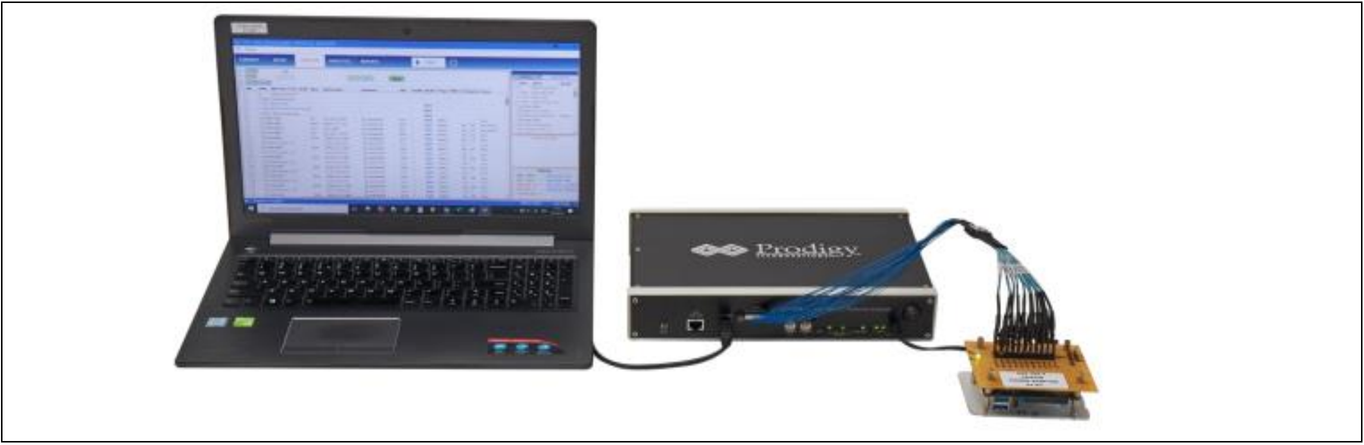
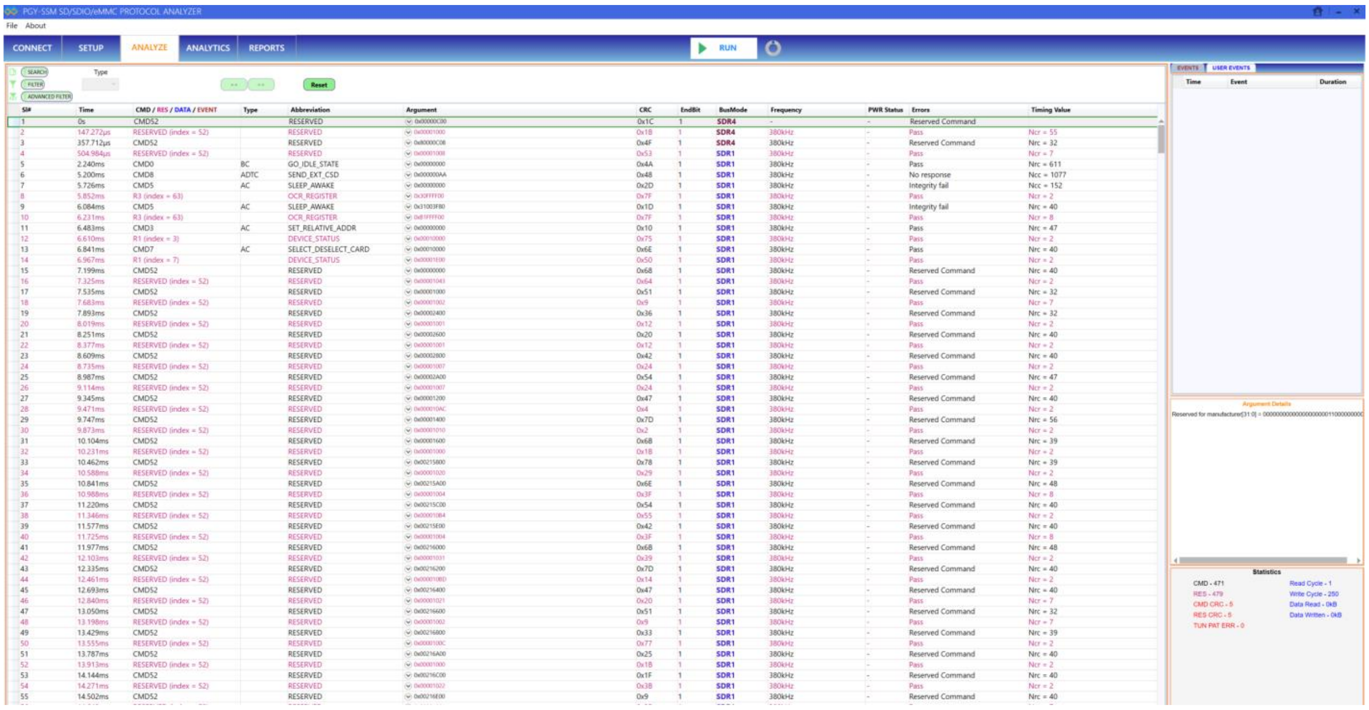


Figure 7 SDIO analyzer and device setup

The software application and more details about the analyzer are available here.

The following is the reference trace capture for the SDR50 configuration.



Sl#	Time	CMD / RES / DATA / EVENT	Type	Abbreviation	Argument	CRC	EndBit	BusMode	Frequency	PWR Status	Errors	Timing Value
1	0s	CMD52		RESERVED	0x00000C00	0x1C	1	SDR4	-	-	Reserved Command	
2	147.272µs	RESERVED (index = 52)		RESERVED	0x00001000	0x18	1	SDR4	380kHz	-	Pass	Ncr = 55
3	357.712µs	CMD52		RESERVED	0x80000C08	0x4F	1	SDR4	380kHz	-	Reserved Command	Nrc = 32
4	504.984µs	RESERVED (index = 52)		RESERVED	0x00001008	0x53	1	SDR1	380kHz	-	Pass	Ncr = 7
5	2.240ms	CMD0	BC	GO_IDLE_STATE	0x00000000	0x4A	1	SDR1	380kHz	-	Pass	Nrc = 611
6	5.200ms	CMD8	ADTC	SEND_EXT_CSD	0x000000AA	0x48	1	SDR1	380kHz	-	No response	Ncc = 1077
7	5.726ms	CMD5	AC	SLEEP_AWAKE	0x00000000	0x2D	1	SDR1	380kHz	-	Integrity fail	Ncc = 152
8	5.852ms	R3 (index = 63)		OCR_REGISTER	0x30FFFF00	0x7F	1	SDR1	380kHz	-	Pass	Ncr = 2
9	6.084ms	CMD5	AC	SLEEP_AWAKE	0x31003F80	0x1D	1	SDR1	380kHz	-	Integrity fail	Nrc = 40
10	6.231ms	R3 (index = 63)		OCR_REGISTER	0x81FFFF00	0x7F	1	SDR1	380kHz	-	Pass	Ncr = 8
11	6.483ms	CMD3	AC	SET_RELATIVE_ADDR	0x00000000	0x10	1	SDR1	380kHz	-	Pass	Nrc = 47
12	6.610ms	R1 (index = 3)		DEVICE_STATUS	0x00010000	0x75	1	SDR1	380kHz	-	Pass	Ncr = 2
13	6.841ms	CMD7	AC	SELECT_DESELECT_CARD	0x00010000	0x6E	1	SDR1	380kHz	-	Pass	Nrc = 40
14	6.967ms	R1 (index = 7)		DEVICE_STATUS	0x00001E00	0x50	1	SDR1	380kHz	-	Pass	Ncr = 2
15	7.199ms	CMD52		RESERVED	0x00000000	0x68	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
16	7.325ms	RESERVED (index = 52)		RESERVED	0x00001043	0x64	1	SDR1	380kHz	-	Pass	Ncr = 2
17	7.535ms	CMD52		RESERVED	0x00001000	0x51	1	SDR1	380kHz	-	Reserved Command	Nrc = 32
18	7.683ms	RESERVED (index = 52)		RESERVED	0x00001002	0x9	1	SDR1	380kHz	-	Pass	Ncr = 7
19	7.893ms	CMD52		RESERVED	0x00002400	0x36	1	SDR1	380kHz	-	Reserved Command	Nrc = 32
20	8.019ms	RESERVED (index = 52)		RESERVED	0x00001001	0x12	1	SDR1	380kHz	-	Pass	Ncr = 2
21	8.251ms	CMD52		RESERVED	0x00002600	0x20	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
22	8.377ms	RESERVED (index = 52)		RESERVED	0x00001001	0x12	1	SDR1	380kHz	-	Pass	Ncr = 2
23	8.609ms	CMD52		RESERVED	0x00002800	0x42	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
24	8.735ms	RESERVED (index = 52)		RESERVED	0x00001007	0x24	1	SDR1	380kHz	-	Pass	Ncr = 2
25	8.987ms	CMD52		RESERVED	0x00002A00	0x54	1	SDR1	380kHz	-	Reserved Command	Nrc = 47
26	9.114ms	RESERVED (index = 52)		RESERVED	0x00001007	0x24	1	SDR1	380kHz	-	Pass	Ncr = 2
27	9.345ms	CMD52		RESERVED	0x00001200	0x47	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
28	9.471ms	RESERVED (index = 52)		RESERVED	0x000010AC	0x4	1	SDR1	380kHz	-	Pass	Ncr = 2
29	9.747ms	CMD52		RESERVED	0x00001400	0x7D	1	SDR1	380kHz	-	Reserved Command	Nrc = 56
30	9.873ms	RESERVED (index = 52)		RESERVED	0x00001010	0x2	1	SDR1	380kHz	-	Pass	Ncr = 2
31	10.104ms	CMD52		RESERVED	0x00001600	0x6B	1	SDR1	380kHz	-	Reserved Command	Nrc = 39
32	10.231ms	RESERVED (index = 52)		RESERVED	0x00001000	0x1B	1	SDR1	380kHz	-	Pass	Ncr = 2
33	10.462ms	CMD52		RESERVED	0x00215800	0x78	1	SDR1	380kHz	-	Reserved Command	Nrc = 39
34	10.588ms	RESERVED (index = 52)		RESERVED	0x00001020	0x29	1	SDR1	380kHz	-	Pass	Ncr = 2
35	10.841ms	CMD52		RESERVED	0x00215A00	0x6E	1	SDR1	380kHz	-	Reserved Command	Nrc = 48
36	10.988ms	RESERVED (index = 52)		RESERVED	0x00001004	0x3F	1	SDR1	380kHz	-	Pass	Ncr = 8
37	11.220ms	CMD52		RESERVED	0x00215C00	0x54	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
38	11.346ms	RESERVED (index = 52)		RESERVED	0x000010B4	0x55	1	SDR1	380kHz	-	Pass	Ncr = 2
39	11.577ms	CMD52		RESERVED	0x00215E00	0x42	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
40	11.725ms	RESERVED (index = 52)		RESERVED	0x00001004	0x3F	1	SDR1	380kHz	-	Pass	Ncr = 8
41	11.977ms	CMD52		RESERVED	0x00216000	0x6B	1	SDR1	380kHz	-	Reserved Command	Nrc = 48
42	12.103ms	RESERVED (index = 52)		RESERVED	0x00001031	0x39	1	SDR1	380kHz	-	Pass	Ncr = 2
43	12.335ms	CMD52		RESERVED	0x00216200	0x7D	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
44	12.461ms	RESERVED (index = 52)		RESERVED	0x000010BD	0x14	1	SDR1	380kHz	-	Pass	Ncr = 2
45	12.693ms	CMD52		RESERVED	0x00216400	0x47	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
46	12.840ms	RESERVED (index = 52)		RESERVED	0x00001021	0x20	1	SDR1	380kHz	-	Pass	Ncr = 7
47	13.050ms	CMD52		RESERVED	0x00216600	0x51	1	SDR1	380kHz	-	Reserved Command	Nrc = 32
48	13.198ms	RESERVED (index = 52)		RESERVED	0x00001002	0x9	1	SDR1	380kHz	-	Pass	Ncr = 7
49	13.429ms	CMD52		RESERVED	0x00216800	0x33	1	SDR1	380kHz	-	Reserved Command	Nrc = 39
50	13.555ms	RESERVED (index = 52)		RESERVED	0x0000100C	0x77	1	SDR1	380kHz	-	Pass	Ncr = 2
51	13.787ms	CMD52		RESERVED	0x00216A00	0x25	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
52	13.913ms	RESERVED (index = 52)		RESERVED	0x00001000	0x1B	1	SDR1	380kHz	-	Pass	Ncr = 2
53	14.144ms	CMD52		RESERVED	0x00216C00	0x1F	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
54	14.271ms	RESERVED (index = 52)		RESERVED	0x00001022	0x3B	1	SDR1	380kHz	-	Pass	Ncr = 2
55	14.502ms	CMD52		RESERVED	0x00216E00	0x9	1	SDR1	380kHz	-	Reserved Command	Nrc = 40

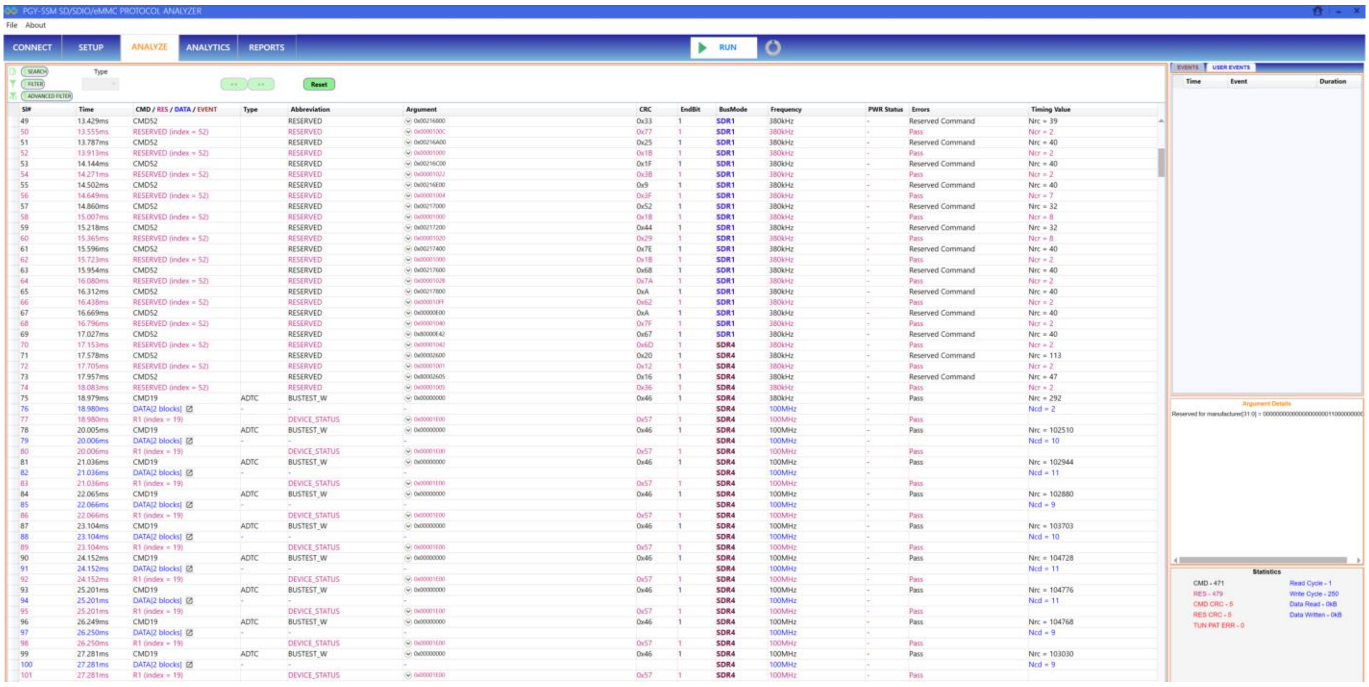
Time	Event	Duration

Argument Details

Reserved for manufacture[31:0] = 00000000000000000000110000000000

Statistics

CMD - 471	Read Cycle - 1
RES - 479	Write Cycle - 250
CMD CRC - 5	Data Read - 0kB
RES CRC - 5	Data Written - 0kB
TUN PAT ERR - 0	



Sl#	Time	CMD / RES / DATA / EVENT	Type	Abbreviation	Argument	CRC	EndBit	BusMode	Frequency	PWR Status	Errors	Timing Value
49	13.429ms	CMD52		RESERVED	0x00216800	0x33	1	SDR1	380kHz	-	Reserved Command	Nrc = 39
50	13.555ms	RESERVED (index = 52)		RESERVED	0x0000100C	0x77	1	SDR1	380kHz	-	Pass	Ncr = 2
51	13.787ms	CMD52		RESERVED	0x00216A00	0x25	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
52	13.913ms	RESERVED (index = 52)		RESERVED	0x00001000	0x1B	1	SDR1	380kHz	-	Pass	Ncr = 2
53	14.144ms	CMD52		RESERVED	0x00216C00	0x1F	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
54	14.271ms	RESERVED (index = 52)		RESERVED	0x00001022	0x3B	1	SDR1	380kHz	-	Pass	Ncr = 2
55	14.502ms	CMD52		RESERVED	0x00216E00	0x9	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
56	14.649ms	RESERVED (index = 52)		RESERVED	0x00001004	0x3F	1	SDR1	380kHz	-	Pass	Ncr = 7
57	14.860ms	CMD52		RESERVED	0x00217000	0x52	1	SDR1	380kHz	-	Reserved Command	Nrc = 32
58	15.007ms	RESERVED (index = 52)		RESERVED	0x00001000	0x1B	1	SDR1	380kHz	-	Pass	Ncr = 8
59	15.218ms	CMD52		RESERVED	0x00217200	0x44	1	SDR1	380kHz	-	Reserved Command	Nrc = 32
60	15.365ms	RESERVED (index = 52)		RESERVED	0x00001020	0x29	1	SDR1	380kHz	-	Pass	Ncr = 8
61	15.596ms	CMD52		RESERVED	0x00217400	0x7E	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
62	15.723ms	RESERVED (index = 52)		RESERVED	0x00001000	0x1B	1	SDR1	380kHz	-	Pass	Ncr = 2
63	15.954ms	CMD52		RESERVED	0x00217600	0x68	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
64	16.080ms	RESERVED (index = 52)		RESERVED	0x00001028	0x7A	1	SDR1	380kHz	-	Pass	Ncr = 2
65	16.312ms	CMD52		RESERVED	0x00217800	0xA	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
66	16.438ms	RESERVED (index = 52)		RESERVED	0x000010FF	0x62	1	SDR1	380kHz	-	Pass	Ncr = 2
67	16.669ms	CMD52		RESERVED	0x00000E00	0xA	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
68	16.796ms	RESERVED (index = 52)		RESERVED	0x00001040	0x7F	1	SDR1	380kHz	-	Pass	Ncr = 2
69	17.027ms	CMD52		RESERVED	0x80000E42	0x67	1	SDR1	380kHz	-	Reserved Command	Nrc = 40
70	17.153ms	RESERVED (index = 52)		RESERVED	0x00001042	0x6D	1	SDR4	380kHz	-	Pass	Ncr = 2
71	17.578ms	CMD52		RESERVED	0x00002600	0x20	1	SDR4	380kHz	-	Reserved Command	Nrc = 113
72	17.705ms	RESERVED (index = 52)		RESERVED	0x00001001	0x12	1	SDR4	380kHz	-	Pass	Ncr = 2
73	17.957ms	CMD52		RESERVED	0x80002605	0x16	1	SDR4	380kHz	-	Reserved Command	Nrc = 47
74	18.083ms	RESERVED (index = 52)		RESERVED	0x00001005	0x36	1	SDR4	380kHz	-	Pass	Ncr = 2
75	18.979ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	380kHz	-	Pass	Nrc = 292
76	18.980ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 2
77	18.980ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
78	20.005ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 102510
79	20.006ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 10
80	20.006ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
81	21.036ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 102944
82	21.036ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 11
83	21.036ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
84	22.065ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 102880
85	22.066ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 9
86	22.066ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
87	23.104ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 103703
88	23.104ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 10
89	23.104ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
90	24.152ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 104728
91	24.152ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 11
92	24.152ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
93	25.201ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 104776
94	25.201ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 11
95	25.201ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
96	26.249ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 104768
97	26.250ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 9
98	26.250ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	
99	27.281ms	CMD19	ADTC	BUSTEST_W	0x00000000	0x46	1	SDR4	100MHz	-	Pass	Nrc = 103030
100	27.281ms	DATA[2 blocks]	-	-	-			SDR4	100MHz	-		Ncd = 9
101	27.281ms	R1 (index = 19)		DEVICE_STATUS	0x00001E00	0x57	1	SDR4	100MHz	-	Pass	

Time	Event	Duration

Argument Details

Reserved for manufacture[31:0] = 00000000000000000000011000000000

Statistics

CMD - 471	Read Cycle - 1
RES - 479	Write Cycle - 250
CMD CRC - 5	Data Read - 0kB
RES CRC - 5	Data Written - 0kB
TUN PAT ERR - 0	

References

[1] SDIO specification.

Glossary

DTS	*device tree source*
GPIO	*general purpose input / output*
MMC	*Multi Media Card*
OOB	*out of band*
RCA	*Relative Card Address*
SDIO	*Secure Digital Input Output*
S18A	*Switching to 1.8 V Accept.*
S18R	*Switching to 1.8-V Request*
UHS	*Ultra-High Speed*

Revision history

Document revision	Date	Description of changes
**	2023-12-11	New user guide.

Edition 2023-12-11

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-37025 Rev. **