

PROJECT REPORT

RB-A5D2X TESTING PERIPHERALS LIST

1. USER LEDS
2. USER SWITCH
3. ADC
4. ETHERNET
5. USB
6. SD CARD
7. USART3 (RS232)
8. UART6 (RS485)
9. EEPROM(I2C)
10. DIN
- 11.DOUT
- 12.PWM
- 13.RTC

1. LED TESTING

DESCRIPTION

- To test GPIO let us use the On-board LED on RuggedBoard. For this demonstration let's use LED_1 which is connected to GPIO PC13 (Refer hardware manual for more information).
- To test GPIO let us use the On-board LED on RuggedBoard. For this demonstration let's use LED_1 which is connected to GPIO PC13 (Refer hardware manual for more information).
- In this LED input voltages : 1.8v to 3v (dc).

PIN	LED No	SIGNAL NAME
1	LED_1 (D4)	PC13/GPIO_LED
2	LED_2 (D7)	PC17/GPIO_LED
3	LED_3 (D17)	PC19/GPIO_LED

README FILE

- The script is using a while loop to read input from the user through the "read" command.
- The user input is then checked against different cases using the "case" statement.
- For each case, the script exports a GPIO pin to the Linux filesystem using the "echo" command and sets its direction to output.
- It then sets the value of the GPIO pin to 0, effectively turning on an LED connected to the pin.
- The script also prints a message indicating which LED has been turned on.
- However, the script is missing a "break" statement after each case,
- which means that if one case is matched, the script will continue executing the next cases.
- It's likely that the script continues with additional code to turn off the LEDs or wait for user input to turn them off.
- It's also worth noting that the script is missing an "exit" statement, which means that it will run indefinitely until interrupted manually.

SHELL SCRIPT

```
#!/bin/sh
while :
do
read CH
{
case "$CH" in
"1")
echo 77 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio77/direction
echo 0 > /sys/class/gpio/gpio77/value
echo "*** LED1 ON ***"

;;

"2")
echo 81 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio81/direction
echo 0 > /sys/class/gpio/gpio81/value
echo "*** LED2 ON ***"
;;

"3")
echo 83 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio83/direction
echo 0 > /sys/class/gpio/gpio83/value
echo "*** LED3 ON ***"
;;

"4")
echo 77 > /sys/class/gpio/export
echo out > /sys/class/gpio/PC13/direction
echo 1 > /sys/class/gpio/PC13/value
echo "***** LED1 OFF *****"
;;

"5")
echo 81 > /sys/class/gpio/export
echo out > /sys/class/gpio/PC17/direction
echo 1 > /sys/class/gpio/PC17/value
echo "***** LED2 OFF *****"
;;

"6")
echo 83 > /sys/class/gpio/export
echo out > /sys/class/gpio/PC19/direction
```

```
echo 1 > /sys/class/gpio/PC19/value
echo    "**** LED3 OFF ****"
;;
esac
}
done
```

TEST LOG

```
done
root@rugged-board-a5d2x:/data/script# sh led.sh
1
**** LED1 ON ****
2
**** LED2 ON ****
3
**** LED3 ON ****
4
sh: write error: Resource busy
**** LED1 OFF ****
5
sh: write error: Resource busy
**** LED2 OFF ****
6
sh: write error: Resource busy
**** LED3 OFF ****
□
```

2. BUTTON/SWITCH TESTING

DESCRIPTION

- User_SW1 button is used for GPIO user level input.
 - If you press this switch that time 5v passing ,if you release the switch off condition.
- User_SW1 button is used for GPIO user level input.

PIN	Switch No	SIGNAL NAME
-----	-----------	-------------

1	sw1	PC12/GPIO_EN
---	-----	--------------

- To test the USER Switch functionality in Kernel with application.To test the USER Switch functionality in Kernel with application.

README FILE

- It starts by printing a set of asterisks, then a few empty lines, and then another set of asterisks to create a header.
- It then exports a GPIO pin with the number 76 to enable it for reading and writing.
- It reads the value of a button connected to the PC12 pin and saves it to a variable called "status".
- If the button is not pressed (i.e., if the value is 0), it prints "Button released", then some empty lines, "PASS", and more empty lines.
- Otherwise, if the button is pressed (i.e., if the value is 1), it prints "Button pressed", then some empty lines, "FAIL", and more empty lines.

SHELL SCRIPT

```
#!/bin/sh
echo "*****"
echo
echo "-----BUTTON/SWITCH TEST-----"
echo
echo "*****"
echo

echo 76 > /sys/class/gpio/export

echo "Status of Button: "
status=$(cat /sys/class/gpio/PC12/value)

if [ "$status" -eq 0 ]; then
    echo "Button released"
    echo
    echo "***** PASS *****"
    echo
else
    echo "Button pressed"
    echo
    echo "***** FAIL *****"
    echo
fi
```

TEST LOG

```
root@rugged-board-a5d2x:/data/script# sh BUTTON.sh
*****

-----BUTTON/SWITCH TEST-----

*****

sh: write error: Resource busy
Status of Button:
Button released

***** PASS *****
```

3. ADC TESTING

DESCRIPTION

- ADC (Analog to Digital Converter) testing in an embedded Linux system involves verifying the functionality and accuracy of the ADC driver, which is responsible for reading analog signals and converting them to digital data.
- To test the ADC functionality with Potentio meter To test the ADC functionality with Potentio meter.

Required Hardware

- phyCORE-A5d2x System on Module
- phyCORE-A5d2 Rugged Board
- USB cable
- Potentiometer

README FILE

- The first three lines just print some header text to the console.
- The fourth line checks if the ADC node exists by reading the raw voltage from the device file /sys/bus/iio/devices/iio:device0/in_voltage6_raw.
- If the node doesn't exist, this command will fail and the script will terminate.
- The fifth line prints another header to the console.

- The sixth line reads the raw voltage value from the ADC and stores it in the variable "var".
- The seventh line prints the raw voltage value to the console.
- The eighth and ninth lines print a message indicating that the ADC test has passed.

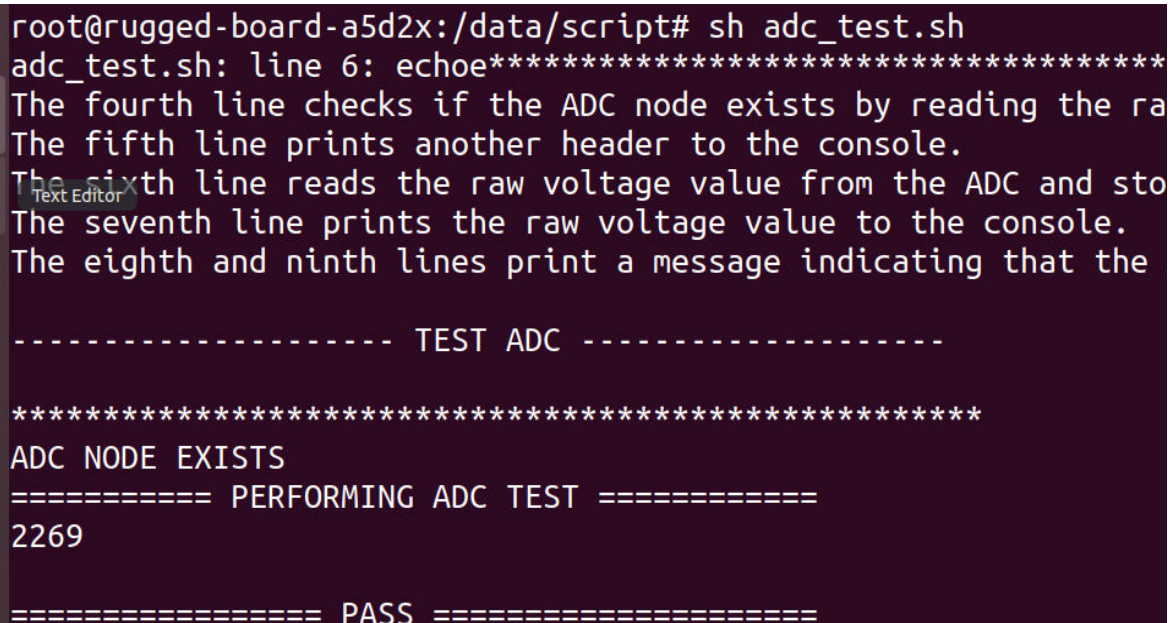
SHELL SCRIPT

```
#!/bin/sh
echo "*****"
echo
echo "----- TEST ADC -----"
echo
echo "*****"
echo "ADC NODE EXISTS"

var=$( cat /sys/bus/iio/devices/iio\:device0/in_voltage6_raw )

echo "===== PERFORMING ADC TEST ====="
echo $var
echo
echo "===== PASS ====="
```

TEST LOG



```
root@rugged-board-a5d2x:/data/script# sh adc_test.sh
adc_test.sh: line 6: echo: *****
The fourth line checks if the ADC node exists by reading the ra
The fifth line prints another header to the console.
The sixth line reads the raw voltage value from the ADC and sto
The seventh line prints the raw voltage value to the console.
The eighth and ninth lines print a message indicating that the

----- TEST ADC -----

*****
ADC NODE EXISTS
===== PERFORMING ADC TEST =====
2269

===== PASS =====
```

4. ETHERNET TESTING

DESCRIPTION

- To test the Ethernet ping functionality in Kernel.To test the Ethernet ping functionality in Kernel.
- **Here are some steps you can take to test Ethernet in RB:**
- Check the Ethernet driver
- Test the Ethernet driver
- Test the bandwidth
- Test the error rate

Required Hardware

- Ethernet Switch / Hub / Router, Ethernet cable.

README FILE

- The first few lines are just echo statements that print out a header to indicate the start of the Ethernet test.
- The "if" statement checks the return value of the udhcpc command. If udhcpc succeeds in configuring the eth0 interface, the "then" block is executed; otherwise, the "else" block is executed.
- If udhcpc succeeds, the script prints a message indicating that the eth0 interface was configured successfully. It then prints the carrier status of the interface by reading the value of the /sys/class/net/eth0/carrier file. Finally, it starts pinging Google's website using the ping command.
- After the ping command completes (either because the user pressed Ctrl+C to stop it or because it timed out), the script prints a "PASS" message to indicate that the test succeeded.
- If udhcpc fails to configure the eth0 interface, the script prints a "FAIL" message to indicate that the test failed.
-

SCRIPT FILE

```
#!/bin/bash
```

```
echo "*****"
echo
echo "----- TEST Ethernet -----"
echo
echo "*****"
```

```
if udhcpc -i eth0 ; then
```



```
echo "eth0 interface configured successfully"
echo "carrier status:"
cat /sys/class/net/eth0/carrier
echo "===== press ctrl+c to stop pinging ====="
```

```
ping google.com
```

```
echo
echo "===== PASS ====="
echo
else
echo "Failed to configure eth0 interface"
echo "===== FAIL ====="
```

```
fi
```

TEST LOG

```
root@rugged-board-a5d2x:/data/script# sh ethernet.sh
*****

----- TEST Ethernet -----

*****
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 10.1.12.46
udhcpc: lease of 10.1.12.46 obtained, lease time 3600
/etc/udhcpc.d/50default: Adding DNS 10.1.80.41
/etc/udhcpc.d/50default: Adding DNS 10.1.80.42
/etc/udhcpc.d/50default: line 86: can't create /etc/resolv.conf:
eth0 interface configured successfully
carrier status:
1
===== press ctrl+c to stop pinging =====
PING google.com (142.250.193.110): 56 data bytes
64 bytes from 142.250.193.110: seq=0 ttl=56 time=23.026 ms
64 bytes from 142.250.193.110: seq=1 ttl=56 time=23.111 ms
64 bytes from 142.250.193.110: seq=2 ttl=56 time=23.060 ms
64 bytes from 142.250.193.110: seq=3 ttl=56 time=23.082 ms
64 bytes from 142.250.193.110: seq=4 ttl=56 time=23.278 ms
64 bytes from 142.250.193.110: seq=5 ttl=56 time=23.117 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 23.026/23.112/23.278 ms
```

5. USB TESTING

DESCRIPTION

- To test the USB1 Port functionality using the dmesg in Kernel with Pen-drive.
- To test the USB1 Port functionality in Kernel with Pen-drive to read and write the data.
- To test the USB1 Port functionality in Kernel with USB host performance test.
- **Here are some steps you can take to test USB :**
- Verify the hardware connections
- Check the USB driver
- Test USB devices
- Test the bandwidth
- est the error rate

Required Hardware

USB PORT/Hub -USB Stick.USB PORT/Hub. -USB Stick.

README FILE

- This is a shell script written in Bash. The script tests the USB functiona
- Asks the user to insert a USB drive for testing.
- Waits for 5 seconds.
- Creates a text file named file.txt with the content "welcome to phytec".
- Checks if any of the USB devices (/dev/sda, /dev/sdb, /dev/sdc) are present on the board.
- If a USB device is found, it mounts the USB device at /mnt directory.
- Copies the file.txt to the USB device.
- Checks if the file was successfully copied to the USB device.
- Unmounts the USB device from the /mnt directory.
- Overall, the script is a basic USB test script that can be used to check the USB functionality of a board. However, the script has some syntax errors that need to be fixed before it can be executed properly.lity of a board. Here's a brief overview of what the script does:

SHELL SCRIPT

```

#!/bin/sh
echo -n "*****"
echo
echo -n "===== USB TEST ====="
echo
echo -n "*****"
echo
echo "Insert any pendrive to board for USB test"
echo
echo
echo "waiting for 5 sec ..."
sleep 5
echo "creating a text file"
touch file.txt
echo "welcome to phytec" > file.txt

if [ -e "/dev/sda" ] || [ -e "/dev/sdb" ] || [ -e "/dev/sdc" ]
then
    echo
    echo "USB device plugged in to board"
    echo "===== mounting the usb device ====="
echo
    mount /dev/sda /mnt
    cd /mnt || exit
    echo "copying file from board to usb device"
    echo
    cp /data/script/file.txt /mnt
    cd || exit
    if [ -e "/mnt/file.txt" ]
    then
        echo "copying of file from baord to usb device successfu
        echo
        echo "===== PASS===== "
        echo
    else
        echo "file not copied to usb device"
    fi
    echo "unmouting usb"
    umount /mnt
fi

```

TEST LOG

```
*****
----- TEST USB -----
*****

Insert any pen drive to board for USB Test

Waiting for 5 sec
Creating test file

USB device plugged in to board
----- Mounting the USB device -----

mount: can't find /dev/sda/mnt in /etc/fstab
Copying file from board to USB device

cp: can't create '/mnt/file.txt': Read-only file system
Files copied
===== PASS =====
root@rugged-board-a5d2x:/data/Testing#
```

6. SDCARD TESTING

DESCRIPTION

- To test the mini pcie connector on the carrier board. To test the mini pcie connector on the carrier board
- To test the SD card Port functionality in Kernel with SD card.
- To test the SD card Port functionality in Kernel with SD card and read the data from SD card.
- test the SD card Port functionality in Kernel with SD card and write the data to SD card

Here are some steps you can take to test the SD card :

- Verify the hardware connections
- Check the SD card driver
- Test the SD card
- Test the file system
- Test the error rate

README FILE

- data.txt in the directory /data/Testing/ using the touch command.

- The script then writes the text "Welcome on phytec" to data.txt.
- The script then checks if the file /dev/mmcblk1p2 exists, which is typically the location of the SD card on a Linux system.
- If the file exists, the script outputs the text "SD card found on the board", and proceeds to mount the SD card to the directory /mnt/.
- After that, the script changes the current directory to /mnt/ and copies the file data.txt from the board to the SD card using the cp command. If the file is successfully copied, the script outputs the text "copying of file from board to sd card successful" and "PASS".
- If the file is not copied, the script outputs "file not copied to sd card" and "FAIL".
- If the file /dev/mmcblk1p2 does not exist, the script outputs the text "No sd card found on the board".

SHEL SCRIPT

```

echo "*****"
echo
echo "===== TEST SD CARD ====="
echo
echo "*****"
touch /data/Testing/data.txt
echo " Welcome on phytec " > data.txt

if [ -e "/dev/mmcblk1p2" ]
then
    echo "SD card found on the board"
    echo "-----mounting the sd card-----"
    mount /dev/mmcblk1p2 /mnt/
    cd /mnt/
    echo "copying file from board to sd card"
    cp /data/Testing/data.txt /mnt/
    if [ -e "/mnt/data.txt" ]
    then
        echo "copying of file from board to sd card successful"
        echo "****PASS****"
    else
        echo "file not copied to sd card"
        echo "****FAIL****"
    fi
else
    echo "No sd card found on the board"
fi

```

TEST LOG

```
root@rugged-board-a5d2x:/data/script# sh sd__card.sh
*****

===== TEST SD CARD =====

*****
touch: /data/Testing/data.txt: No such file or directory
SD card found on the board
-----mounting the sd card-----
mount: mounting /dev/mmcblk1p2 on /mnt/ failed: Resource busy
copying file from board to sd card
cp: can't stat '/data/Testing/data.txt': No such file or directory
copying of file from board to sd card successful
****PASS****
```

7. UART(RS232) TESTING

DESCRIPTION

- To check UART 3 node in /dev/.
- To test the UART3 Port functionality in Kernel by shorting Transmitter and receiver pins.
- To test the UART3 Port functionality in Kernel by connecting Transmitter pin of first board to receiver pin of second board and connect Receiver pin of first board to transmitter pin of second board using patch cards.
- INPUT:RS232 connector
- Software Node:/dev/ttyS1
- PIN NUM_TX: PB27
- PIN NUM_RX: PB26
- SHORT PINS :TX _PIN&RX_PIN
- BOADRATE:115200
- HARDWARE FLOW CONTROL :NO
- **Required Hardwar**

- UART PORT .
- PATCH CORDS

README FILE

- This is a shell script that performs a loopback test on a UART (Universal Asynchronous Receiver-Transmitter) device.
- The script first prints a few lines of text to indicate the start of the test. It then checks for the existence of the UART device `"/dev/ttyS3"`.
- If the device exists, the script prints a message indicating the UART mode is available and proceeds to perform a loopback test.
- The loopback test likely involves running some code to send data from the device's output to its input and verifying that the data is correctly received.
- If the test is successful, the script prints a "pass" message; otherwise, it prints a "fail" message.
- Overall, the script appears to be a simple test script to check the functionality of a UART device and verify that it is capable of transmitting and receiving data.

SHELL SCRIPT

```
#!/bin/sh

echo -n "*****"

echo

echo -n"----- test uart -----"

echo

echo -n"*****"

echo

echo "-----checking for uart-----"

echo

if [ -e"/dev/ttyS3" ]

then

    echo "uart mode exist"
```

```

        echo
        echo "-----performing loopback test-----"
        echo
#./uart

        echo
        echo "=====pass=====
        echo
else
        echo "=====fail=====
        echo
fi

```

TEST LOG

```

root@rugged-board-a5d2x:/data/script# sh uart.sh
*****
-n----- test uart -----
-n*****
-----checking for uart-----
uart mode exist
-----performing loopback test-----
=====pass=====

```

8. RS485 TESTING

DESCRIPTION

- To test the RS485/UART Device functionality using two boards.To test the RS485/UART Device functionality using two boards.
- INPUT:RS485 connector
- Software Node:/ls /dev/ttyS2
- PIN NUM_TX: PD23
- PIN NUM_RX: PD24
- SHORT PINS :TX _PIN&RX_PIN
- BOADRATE:115200
- HARDWARE FLOW CONTROL :NO
-

Required Hardware

- RS485 Phy should be mounted on Board.
- Patch cords

Here are some steps you can take to test RS485:

- Verify the hardware connections
- Check the RS485 driver
- Test the RS485 port
- Test the baud rate
- Test the error rate

README FILE

- This is a shell script that checks if an RS485 node exists and performs a test if it does.
- The script starts by printing out a header message, then checks if the file "/dev/ttyS2" exists. If it does, it prints a message indicating that the RS485 node exists and proceeds to run the "microcom" command with a baud rate of 115200 and the device "/dev/ttyS2" as its argument.
- If "/dev/ttyS2" does not exist, the script prints a message indicating failure.
- After the microcom command is executed, the script prints a message indicating that the test has passed.

- Overall, this script is useful for testing the functionality of an RS485 node and checking for its existence.

SHELL SCRIPT

```
#!/bin/sh
echo "*****"

echo

echo "----- RS485 TEST -----"

echo

echo "*****"

echo

echo "-----checking for RS485-----"

echo

if [ -e "/dev/ttyS2" ]
then
    echo " RS485 Node exists"

    echo

    echo " ----- Performing 485-----"

    echo

    echo "-----pass-----"

    echo " press Ctrl+C to exit to microcom "

    microcom -s 115200 /dev/ttyS2

    echo

    else

    echo "-----fail-----"

fi
```

TEST LOG

```

root@rugged-board-a5d2x:/data/script# sh rs485.sh
*****

----- RS485 TEST -----

*****atmel*****

-----checking for RS485-----

RS485 Node exists

----- Performing 485-----

-----pass-----
press Ctrl+C to exit to microcom
usart_serial atmel_usart_serial.2.auto: using dma0chan7 for rx DMA transfers
atmel_usart_serial atmel_usart_serial.2.auto: using dma0chan8 for tx DMA transfers

```

9. ETHERNET TESTING

DESCRIPTION

- To test the Ethernet ping functionality in Kernel.To test the Ethernet ping functionality in Kernel.
- 1ST STEP :network configuration
- connection : Host & Target (RJ45)
- INPUT: set the ip address
- COMMAND:ifconfig eth0 ip address up
- COMMAND:ping ip address
- Iperf -s
- iperf -c ip address

- **Required Hardware**

Ethernet Switch / Hub / Router, Ethernet cable.

- **Here are some steps you can take to test Ethernet:**

Verify the hardware connections

Check the Ethernet driver:

Test the Ethernet interface

Test the bandwidth

Test the error rate:

README FILE

- The first few lines are just echo statements that print out a header to indicate the start of the Ethernet test.
- The "if" statement checks the return value of the udhcpc command. If udhcpc succeeds in configuring the eth0 interface, the "then" block is executed; otherwise, the "else" block is executed.
- If udhcpc succeeds, the script prints a message indicating that the eth0 interface was configured successfully. It then prints the carrier status of the interface by reading the value of the /sys/class/net/eth0/carrier file. Finally, it starts pinging Google's website using the ping command.
- After the ping command completes (either because the user pressed Ctrl+C to stop it or because it timed out), the script prints a "PASS" message to indicate that the test succeeded.
- If udhcpc fails to configure the eth0 interface, the script prints a "FAIL" message to indicate that the test failed.
-

SCRIPT FILE

```
#!/bin/bash

echo "*****"
echo
echo "----- TEST Ethernet -----"
echo
echo "*****"

if udhcpc -i eth0 ; then
    echo "eth0 interface configured successfully"
    echo "carrier status:"
    cat /sys/class/net/eth0/carrier
    echo "===== press ctrl+c to stop pinging ====="

    ping google.com

    echo
```

```

echo "===== PASS ====="
echo
else
echo "Failed to configure eth0 interface"
echo "===== FAIL ====="
fi

```

TEST LOG

```

root@rugged-board-a5d2x:/data/script# sh ethernet.sh
*****

----- TEST Ethernet -----

*****
udhcpd: started, v1.27.2
udhcpd: sending discover
udhcpd: sending select for 10.1.12.46
udhcpd: lease of 10.1.12.46 obtained, lease time 3600
/etc/udhcpd.d/50default: Adding DNS 10.1.80.41
/etc/udhcpd.d/50default: Adding DNS 10.1.80.42
/etc/udhcpd.d/50default: line 86: can't create /etc/resolv.conf:
eth0 interface configured successfully
carrier status:
1
===== press ctrl+c to stop ping =====
PING google.com (142.250.193.110): 56 data bytes
64 bytes from 142.250.193.110: seq=0 ttl=56 time=23.026 ms
64 bytes from 142.250.193.110: seq=1 ttl=56 time=23.111 ms
64 bytes from 142.250.193.110: seq=2 ttl=56 time=23.060 ms
64 bytes from 142.250.193.110: seq=3 ttl=56 time=23.082 ms
64 bytes from 142.250.193.110: seq=4 ttl=56 time=23.278 ms
64 bytes from 142.250.193.110: seq=5 ttl=56 time=23.117 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 23.026/23.112/23.278 ms

```

10. DIGITAL INPUT TESTING

DESCRIPTION

- To test the digital input pin on the carrier board. To test the digital input pin on the carrier board.
- INPUT: P5 CONNECTOR
- PIN NUM1:DIN(0-24V)_01
- PIN NUM5:DGND_ISO_IN
- GPIO PIN:PC20,GROUND

- GPIO EXPORT:84
- GPIO DIRECTION :PC20

Required Hardware

- phyCORE-A5d2x System on Module
- phyCORE-A5d2 Rugged Board
- USB cable
- Patch card

Here are some steps you can take to test digital input:

Verify the hardware connections

Check the digital input driver:

Test the digital input port

Test the signal quality

README FILE

- The script exports the GPIO pin by writing its number (84) to the `"/sys/class/gpio/export"` file.
- The script then checks whether the GPIO pin has been successfully exported by testing whether the `"/sys/class/gpio/PC20"` directory exists.
- After exporting the GPIO pin, the script sets its direction to "in" by writing "in" to the `"/sys/class/gpio/PC20/direction"` file.
- The script then waits for 2 seconds using the "sleep" command to give the input some time to stabilize.
- Next, the script reads the value of the GPIO pin by reading the contents of the `"/sys/class/gpio/PC20/value"` file and storing it in the "gpio_value" variable.
- Finally, the script checks whether the GPIO pin is high or low by testing whether the "gpio_value" variable is equal to 0. If it is, the script prints a message indicating that the digital input is connected, and exits with a status of 0 (success). If the GPIO pin is not connected, the script prints a message indicating that the digital input is not connected, and exits with a status of 1 (failure).

SHELL SCRIPT

```
#!/bin/bash
echo
echo -n "-----check digital input -----"
echo
echo -n "*****"
echo
    echo "check if R63 register is mounted or not"
    echo
    echo " -----check pin1- GPIO5-5-----"
    echo
    echo 84 > /sys/class/gpio/export
    echo
    if [ -d "/sys/class/gpio/PC20" ]
    then
        echo "gpio 84 exported"
    else
        echo "fail to export gpio84"
    fi
    echo in > /sys/class/gpio/PC20/direction
    sleep 2
    gpio_value=$(cat /sys/class/gpio/PC20/value)
    if [ "$gpio5_value" == 0 ]
then
    echo "gpio5 din connected"
    echo
    echo "=====pass======"
    echo
else
    echo "gpio5 din not connected"
    echo
    echo "=====fail======"
    echo
fi
```

TEST LOG

```
root@rugged-board-a5d2x:/data/script# sh din.sh

-----check digital input -----
*****
check if R63 register is mounted or not

 -----check pin1- GPIO5-5-----

sh: write error: Resource busy

gpio 84 exported
gpio5 din connected

=====pass=====
```

11. DIGITALOUT TESTING

DESCRIPTION

- To test the digital output pins on carrier boardTo test the digital output pins on carrier board

Required Hardware

- phyCORE-A5d2x System on Module
- phyCORE-A5d2 Rugged Board
- USB cable
- External LED

README FILE

- The first three lines just print some header text to the console.
- The fourth line checks if the ADC node exists by reading the raw voltage from the device file `/sys/bus/iio/devices/iio:device0/in_voltage6_raw`.
- If the node doesn't exist, this command will fail and the script will terminate.
- The fifth line prints another header to the console.
- The sixth line reads the raw voltage value from the ADC and stores it in the variable "var".
- The seventh line prints the raw voltage value to the console.
- The eighth and ninth lines print a message indicating that the ADC test has passed.

SHELL SCRIPT

```
#!/bin/sh
echo "-----check digital output-----"
echo "=====
echo "Check if r63 resistor is mounted, and then connect 6-pin connector"
echo "-----"
echo "Checking pin2-gpio1_io15"
echo "-----"
echo "Exporting gpio97"
echo 97 > /sys/class/gpio/export
echo "Setting direction of gpio97 to output"
echo "out" > /sys/class/gpio/PD1/direction
echo "Setting value of gpio97 to 1"
echo 1 > /sys/class/gpio/PD1/value
sleep 2
echo "Setting value of gpio97 to 0"
```



```

echo 0 > /sys/class/gpio/PD1/value
sleep 2
gpio97_value=$(cat /sys/class/gpio/PD1/value)
if [ "$gpio97_value" = 1 ]
then
    echo "gpio97 dout connected"
    echo "=====pass======"
else
    echo "gpio97 dout not connected"
    echo "=====fail======"
fi

```

TEST LOG

```

root@rugged-board-a5d2x:/data/script# sh dout.sh
-----check digital output-----
=====
Check if r63 resistor is mounted, and then connect 6-pin connector
-----
Checking pin2-gpio1_io15
-----
Exporting gpio97
sh: write error: Resource busy
Setting direction of gpio97 to output
Setting value of gpio97 to 1
Setting value of gpio97 to 0
gpio97 dout not connected
=====fail=====

```

12. PWM TESTING

DESCRIPTION

- To test the PWM functionality in Kernel with External LED.To test the PWM functionality in Kernel with External LED.
- INPUT: M1 CONNECTOR
- PIN NUM :16 (LED)
- PWM TIME PERIOD: 50000
- PWM DUTY_CYCLE: 4000
- OUTPUT: LED BRITNESS

Required Hardware

- phyCORE-A5d2x System on Module
- phyCORE-A5d2 Rugged Board
- USB cable
- External LED

README FILE

- This is a shell script that tests the functionality of a PWM (Pulse-Width Modulation) device.
- The script first prints a few lines of text to indicate the start of the test.
- It then checks for the existence of the PWM device `"/sys/class/pwm/pwmchip0/"`.
- If the device exists, the script exports the `"pwm0"` channel, sets the PWM time period, sets the duty cycle, enables the PWM output, and waits for 3 seconds.
- Finally, the script checks the status of the PWM output to verify that it is enabled.
- If the output is enabled, the script prints a `"pass"` message; otherwise, it prints a `"fail"` message.
- If the PWM device does not exist, the script prints a `"fail"` message.
- Overall, the script appears to be a simple test script to check the functionality of a PWM device and verify that it is capable of generating a pulse-width modulated signal.

SHELL SCRIPT

```
#!/bin/sh

echo

echo "-----checking for pwmchip0-----"

echo

if [ -d "/sys/class/pwm/pwmchip0/" ]

then

    echo "pwmchip0 exist"
```

```
echo
echo "exporting pwm0"
echo
echo 0 > /sys/class/pwm/pwmchip0/export
echo " PWM TEST "
echo
sleep 3
echo "setting pwm time period"
echo 50000 > /sys/class/pwm/pwmchip0/pwm0/period
echo
echo "setting duty cycle"
echo 2500 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
echo
sleep 3
echo 1200 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
echo "enabling pwm"
echo
echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
echo
echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
sleep 3
echo
status=$(cat /sys/class/pwm/pwmchip0/pwm0/enable)
    if [ "$status" == 1 ]
    then
        echo
        echo "-----pass-----"
        echo
```

```

else

    echo

    echo "-----fail-----"

    echo

fi

else

    echo "no pwm chip found"

    echo "=====fail===== "

fi
else
    echo "no pwm chip found"
    echo "=====fail===== "
fi

```

TEST LOG

```

root@rugged-board-a5d2x:/data/script# sh PWM.sh
*****
----- PWM Test-----
*****
PWM Exist
PWM Test
Setting PWM Time Period
Setting duty cycle
Enabling PWM

===== PASS =====

```

13. RTC TESTING

DESCRIPTION

- To test the RTC node functionality in Kernel.To test the RTC node functionality in Kernel.
- RTC Node :ls /dev/rtc0
- OUTPUT: SHOW the date and time

README FILE

- This is a shell script that performs the following tasks:Checks if the RTC (Real-Time Clock) device is available in the system.
- Restarts the chronyd service.
- Sets the timezone to Asia/Kolkata.
- Prints the current date and time.
- Performs a simple test by comparing the output of the "date" command with itself.
- The script is commented out the line that restarts the chronyd service, so it won't actually restart the service if executed. Also, the if statement in the test doesn't seem to make sense, as it compares the variable with itself.
- Assuming that the missing comparison in the if statement was a typo, the script appears to be a basic system check script that verifies the RTC device and the system's timezone, and checks if the "date" command is working correctly.

SCRIPT FILE

```
#!/bin/sh
echo
echo "----- RTC TEST -----"
echo

if [ -e "/dev/rtc0" ]
then
    echo "RTC device found: /dev/rtc0"
else
    echo "No RTC device found"
fi
else
    echo "No RTC device found"
fi
```

```
echo
echo "Restarting chronyd service..."
#systemctl restart chronyd
echo
echo "Setting timezone to Asia/Calcutta..."
ln -sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime
```

```
echo
echo "Current date and time:"
date
```

```
echo
var=$(date)
if [ "$var" == "$var" ]
then
    echo "*** PASS ***"
else
    echo "*** FAIL ***"
fi
```

TEST LOG

```
root@rugged-board-a5d2x:/data/script# sh rtc.sh

----- RTC TEST -----

RTC device found: /dev/rtc0

Restarting chronyd service...

Setting timezone to Asia/Calcutta...
ln: /etc/localtime: Read-only file system

Current date and time:
Sun Jan  1 02:25:15 UTC 2012

*** PASS ***
```

