SBC21/NSD21/EC21 Series

GPIO

Application Note

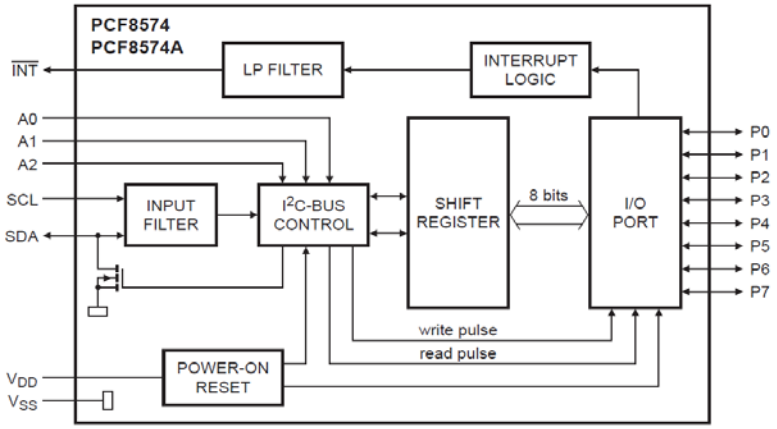
Release Notes

Version	Release Date	Notes
1.0	Oct 7th , 2013	Initial release
1.1	Oct 30th , 2013	Correct some errors
1.2	Jan 24th , 2014	Add the SBC/NSD/EC2104, 2105 series pin definition

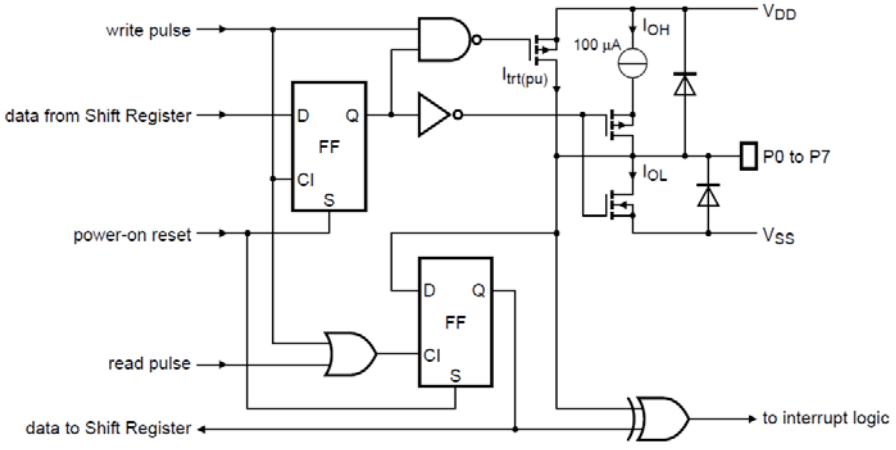
Table of Contents

1. GPIO Controller General Description

The SBC/NSD/EC21 series contains a GPIO controller named PCF8574 which provides an extra 8 usable 8 I/O ports. The PCF8574/74A provides general-purpose remote I/O expansion via the two-wire bidirectional I2C bus. The controller consists of eight quasi-bidirectional ports, a 100 kHz I2C-bus interface, three hardware address inputs, and an interrupt output operating between 2.5 V and 6 V. The quasi-bidirectional port can be independently assigned as an input to monitor interrupt status or keypads, or as an output to activate indicator devices such as LEDs. The system master can read from the input port or write to the output port through a single register. The low current consumption of 25 mA (typical, static) is great for mobile applications, and the latched output ports can directly drive LEDs.



Controller Block Diagram



Simplified Schematic Diagram of P0 to P7

2. GPIO Electrical Characteristics

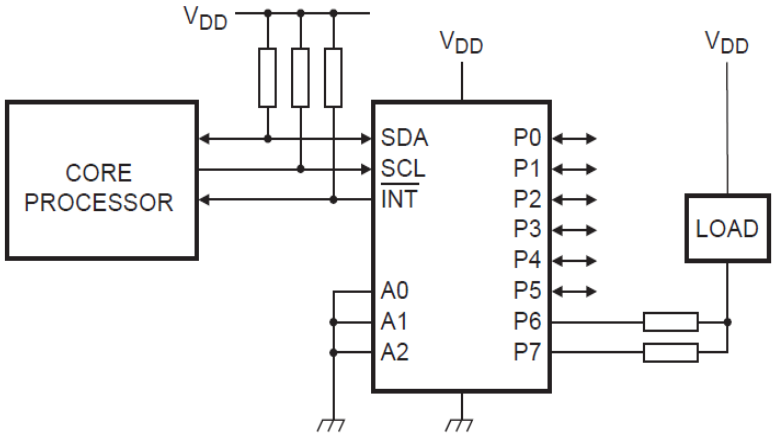
PARAMETER		TEST CONDITIONS	VCC	MIN	TYP(1)	MAX	UNIT
V_{IK} Input diode clamp voltage		I_I = –18 mA V	2.5 V to 6 V	–1.2			V
V_{POR} Power-on reset voltage(2)		V_I = V_{CC} or GND, IO = 0	6 V		1.3	2.4	V
I_{OH}	P port	V_O = GND	2.5 V to 6 V	30		300	mA
I_{OHT}	P-port transient pullup current	High during acknowledge, V_{OH} = GND	2.5 V		–1		mA
I_{OL}	P port	V_O = 1 V	5 V	10	25		mA
I_I	A0, A1, A2 ±5	V_I = V_{CC} or GND	2.5 V to 6 V			±5	µA
I_{IHL}	P port	$V_I \geq V_{CC}$ or $V_I \leq$GND	2.5 V to 6 V			±400	mA
C_{io}	P port	V_{IO} = V_{CC} or GND	2.5 V to 6 V		4	10	pF

(1) All typical values are at V_{CC} = 5 V, T_A = 25°C.

(2) The power-on reset circuit resets the I2C bus logic with $V_{CC} < V_{POR}$ and sets all I/Os to logic high (with current source to V_{CC}).

3. GPIO High Current-Drive Load Applications

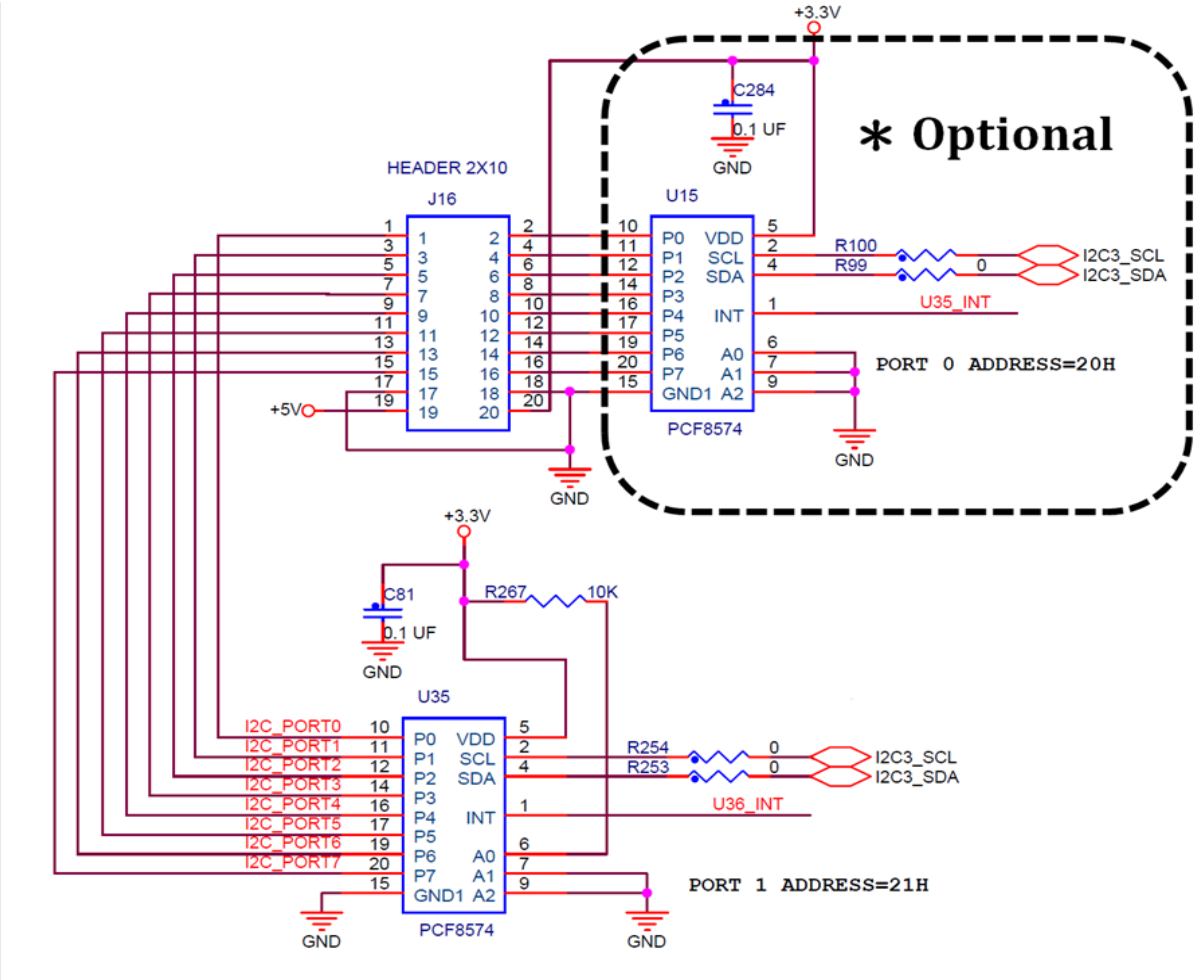
The GPIO has a guaranteed minimum sinking current of 10 mA per bit at 5 V. For applications requiring additional drive, two port pins may be connected together to sink up to 20 mA current. Both bits must then be always turned on or off together. Up to five pins can be connected together to drive 80 mA, which is the device recommended total limit. Each pin needs its own limiting resistor as shown below to to prevent damage to the device should all ports not be turned on at the same time.



High current-drive load application

4. GPIO Pin Assignment

SBC/NSD/EC2107,2110 GPIO circuit:



GPIO Schematic

- Note that 0-P0 to 0-P7 are optional GPIO pins
- J16 Header Pin Definitions:

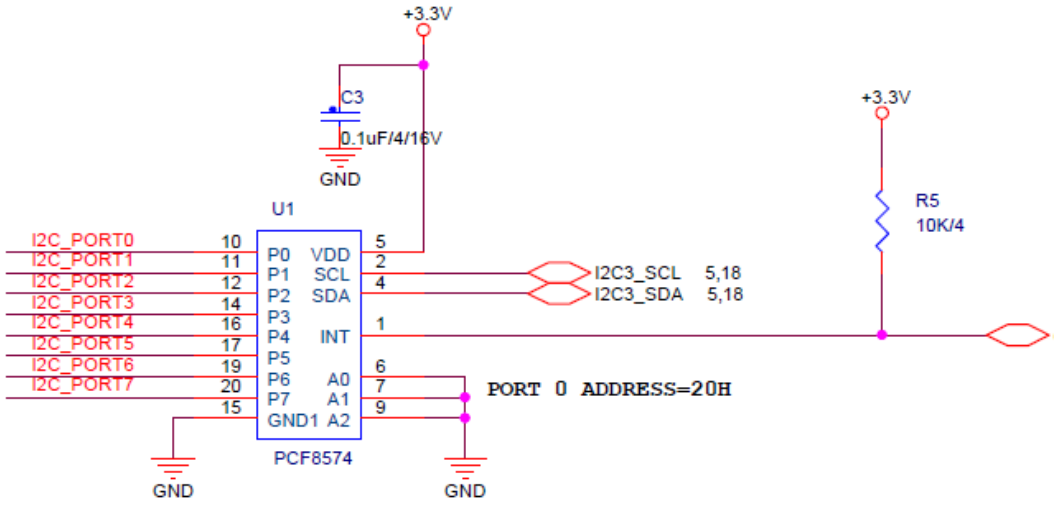
Header Pin	1	3	5	7	9	11	13	15	17	19
Pin Number	1-P0	1-P1	1-P2	1-P3	1-P4	1-P5	1-P6	1-P7	GND	5V
Header Pin	2	4	6	8	10	12	14	16	18	20
Pin Number	0-P0	0-P1	0-P2	0-P3	0-P4	0-P5	0-P6	0-P7	GND	3.3V

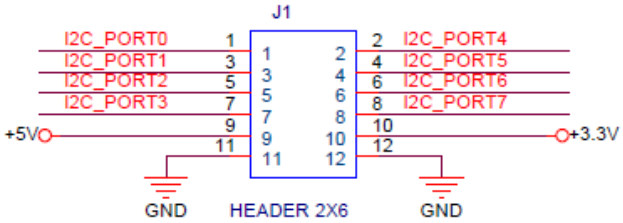
The SBC/NSD/EC2107, 2110 series has 8 GPIO pins available for the customer to use. An extra 8 GPIO pins are optional for ordering.

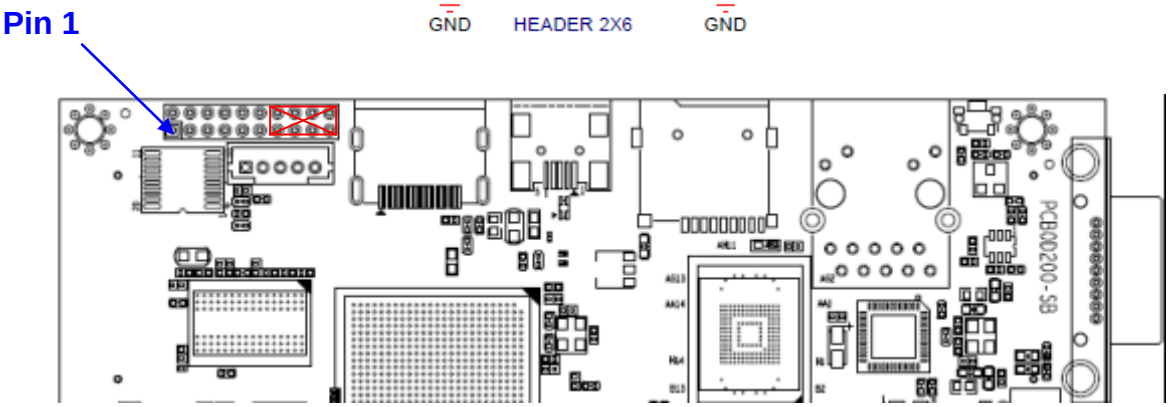
SBC/NSD2104, 2105 GPIO circuit:

The GPIO pin header provides user to connect up to 8 GPIO devices (+3.3V signal level).

The GPIO pins are available on J1 pin header.

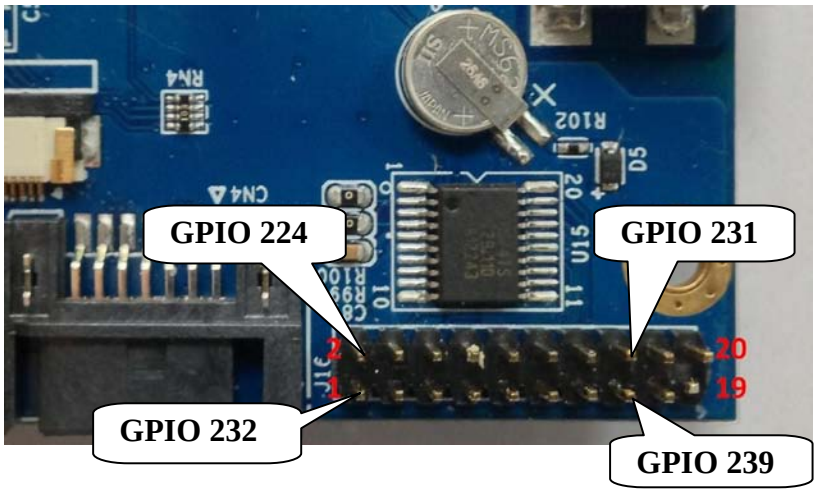




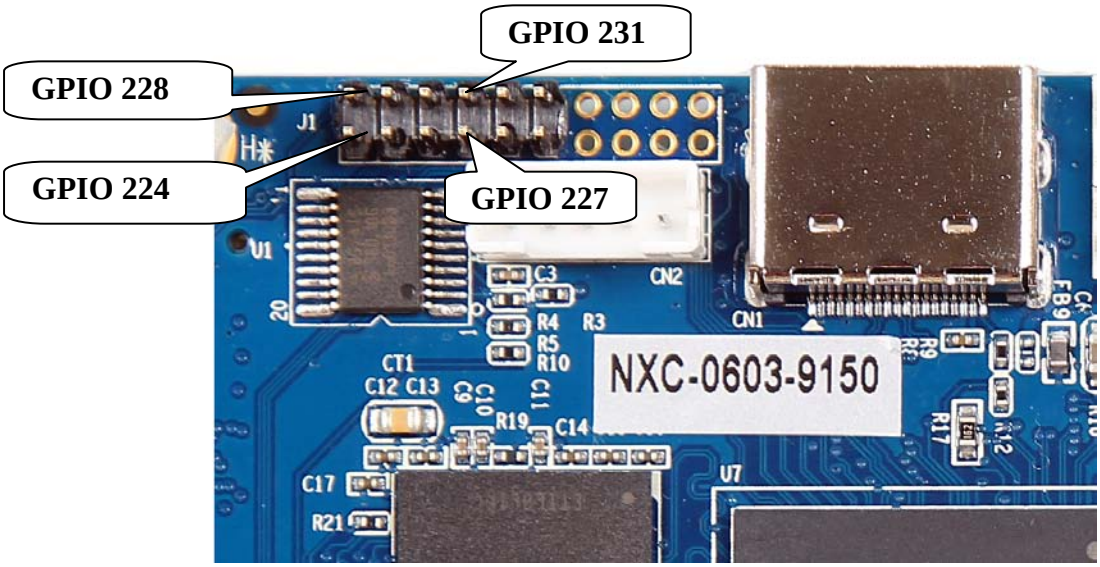


Header Pin	2	4	6	8
Pin number	P4	P5	P6	P7
Header Pin	1	3	5	7
Pin number	P0	P1	P2	P3

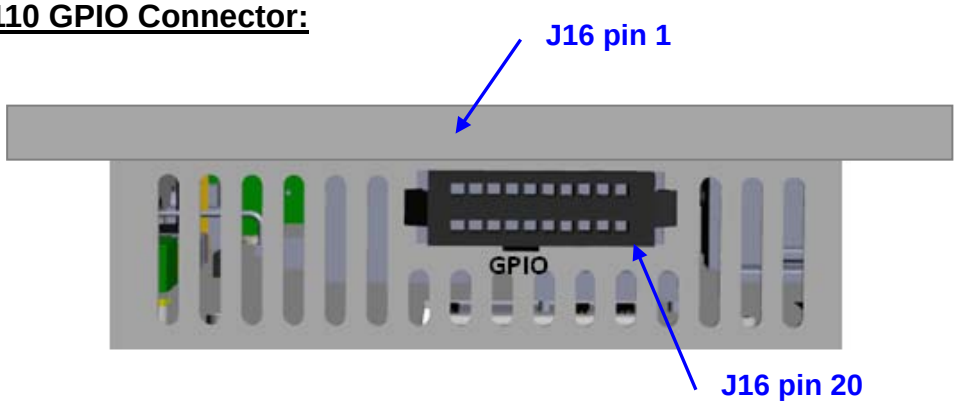
SBC/NSD2107, 2110 GPIO Connector:



SBC/NSD2104,2105 GPIO Connector:



EC2107, 2110 GPIO Connector:



SBC/NSD/EC2107, 2110 GPIO pin assignment:

The GPIO pin numbers on the **SBC/NSD/EC2107, 2110 series** are defined in the range from 224 to 239. When you are configuring the GPIO for application, you must make sure that the header pin number corresponds to the **correct GPIO Pin Number**. Otherwise you might get no response or an error. Shown below are the GPIO pin number definitions.

J16 pin #	1	3	5	7	9	11	13	15	17	19
Port-pin	1-P0	1-P1	1-P2	1-P3	1-P4	1-P5	1-P6	1-P7	GND	5V
GPIO #	**232**	**233**	**234**	**235**	**236**	**237**	**238**	**239**	**---**	**---**

J16 pin #	2	4	6	8	10	12	14	16	18	20
Port-pin	0-P0	0-P1	0-P2	0-P3	0-P4	0-P5	0-P6	0-P7	GND	3.3V
GPIO #	**224**	**225**	**226**	**227**	**228**	**229**	**230**	**231**	**---**	**---**

SBC/NSD2104,2105 GPIO pin assignment:

The GPIO pin numbers on the **SBC/NSD2104, 2105 series** are defined in the range from 224 to 231. When you are configuring the GPIO for application, you must make sure that the header pin number corresponds to the **correct GPIO Pin Number**. Otherwise you might get no response or an error. Shown below are the GPIO pin number definitions.

J1 pin #	1	3	5	7	9	11
Port-pin	P0	P1	1-P2	1-P3	1-P4	1-P5
GPIO #	**224**	**225**	**226**	**227**	**5V**	**GND**

J1 pin #	2	4	6	8	10	12
Port-pin	0-P0	0-P1	0-P2	0-P3	0-P4	0-P5
GPIO #	**228**	**229**	**230**	**231**	**3.3V**	**GND**

5. GPIO Control Command List

Read GPIO status

Note : The SBC/NSD/EC2107, 2110 Series default port address is **0x21**

The SBC/NSD2104, 2105 Series default port address is **0x20**

#**sudo pcfa 1 0x20**

0xFF means 0-P0 to 0-P7 are all pull-high

0x00 means 0-P0 to 0-P7 are all pull-low

#**sudo pcfa 1 0x21**

0xFF means 1-P0 to 1-P7 are all pull-high

0x00 means 1-P0 to 1-P7 are all pull-low

Set GPIO status

#**sudo pcfa 0 0x20 0xFF**

Set all GPIO pull-high (0-P0 to 0-P7)

#**sudo pcfa 0 0x20 0x00**

Set all GPIO pull-low (0-P0 to 0-P7)

#**sudo pcfa 0 0x21 0xFF**

Set all GPIO pull-high (1-P0 to 1-P7)

#**sudo pcfa 0 0x21 0x00**

Set all GPIO pull-low (1-P0 to 1-P7)

6. GPIO Control, Sample Code (Android)

For example, when controlling GPIO 224 :

```
private Process su;

void USBGPIOInit() {

        boolean GPIODir = (new File("/sys/class/gpio/gpio224")).exists();
        //Detect whether GPIO sysfs has been built

        if(GPIODir == false) {
        //If GPIO sysfs has not been built, create gpio directory
            try {
                String cmd = "echo 224 > /sys/class/gpio/export \n";
                su.getOutputStream().write(cmd.getBytes());
                //Create gpio F6 directory
                cmd = "echo out > /sys/class/gpio/gpio224/direction \n";
                su.getOutputStream().write(cmd.getBytes());
                //Set gpio F6 to output and value as 0
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

    }



void USBGPIOSet(int type) {
        String cmd = "";
        if(type == USBOn) {
            cmd = "echo 1 > /sys/class/gpio/gpio224/value \n";
            //Set gpio F6 value to 1
        }
        else if(type == USBOff) {
```

```
            cmd = "echo 0 > /sys/class/gpio/gpio224/value \n";
            //Set gpio F6 value to 0
        }
        else {
            cmd = "echo 1 > /sys/class/gpio/gpio224/value \n";
            //Set gpio F6 value to 1
        }

        try {
            su.getOutputStream().write(cmd.getBytes());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

int USBGPIOGet() {
        FileReader GPIO_Reader;
        String data1 = "0";
        try {
            GPIO_Reader = new FileReader("/sys/class/gpio/gpio224/value");
            BufferedReader bfr = new BufferedReader(GPIO_Reader);
            data1 = bfr.readLine();
            bfr.close();
            //Read gpio F6 value
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        //Toast.makeText(this, "GPIO:" + data1,Toast.LENGTH_LONG).show();

        if(data1.compareTo("0") == 0)
            return 0;
        else
            return 1;
```

```
}
```

7. GPIO Control, Sample Code (Ubuntu)

For example, when controlling GPIO 224 :

```
void sysfs_init()
{
	//Detect whether GPIO sysfs has been built
	FILE *fPtr = fopen("/sys/class/gpio/gpio224", "w");
	if (fPtr != NULL)
		fclose(fPtr);
	else
	{
		system("echo 224 > /sys/class/gpio/export");
		system("echo out > /sys/class/gpio/gpio224/direction");
	}
}

void sysfs_deinit()
{
	FILE *fPtr = fopen("/sys/class/gpio/gpio224", "w");
	if (fPtr != NULL)
		fclose(fPtr);
	else
		system("echo 224 > /sys/class/gpio/unexport");
}

void GPIO_Set(int value)
{
	//Set GPIO value as input
	FILE *fPtr = fopen("/sys/class/gpio/gpio224/value", "w");
	fprintf(fPtr, "%d", value);
	fclose(fPtr);
}

int GPIO_Get(int value)
```

```
{
        //Get GPIO value
        char data;
        FILE *fPtr = fopen("/sys/class/gpio/gpio224/value", "r");
        fscanf(fPtr, " %c ", &data);
        fclose(fPtr);
        return data;
}
```