GPIO HowTo

Using GPIO on Fujitsu's OEM mainboards

Rainer König

GPIO HowTo
Using GPIO on Fujitsu's OEM mainboards

Author Rainer König *Rainer.Koenig@ts.fujitsu.com*

This document explains how to use the General Purpose I/O features of current Fujitsu mainboards with various Linux distributions.


DRAFT

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then

click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above: username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting.

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.

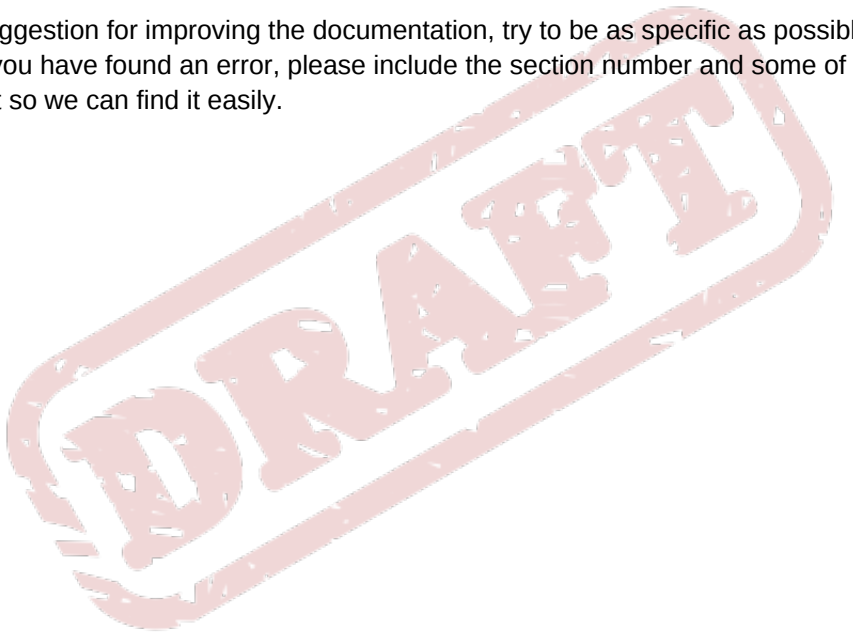
Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you!

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.




DRAFT

Introduction

1.1. General information

Fujitsu industrial mainboards provide GPIO connector with 8 pins that can be programmed as input or output pins and that can be controlled by the PCA953x kernel module.

The kernel module should be shipped with all Linux distributions that have a kernel version >=3.1.x.

This HowTo describes the steps to setup the GPIO interface and how to use it.

1.2. Required prerequisites and skills

To do the configuration steps described in this document you need *root access* for your Linux operating system.

Important

All shell commands must be executed with root privileges.

If your distribution kernel does not provide the required device driver module then you need to compile it by yourself. Therefore you need to know how to compile kernel modules for your distribution kernel. This also requires that your system has the complete toolchain for building kernel modules installed.

You need to know how to locate and install packages for your Linux distribution.


DRAFT

Fujitsu OEM mainboard overview

2.1. Mainboards and kernel drivers

This table shows you which mainboards you can use for GPIO and what kernel-driver is needed to access the SM-Bus interface on those mainboards.

Table 2.1. Fujitsu mainboards

Board	SM-Bus driver	Comment
D3003-Sx	i2c-piix4	
D3076-Sx	i2c-i801	
D3236-Sx	i2c-i801	see southbridge drivers
D313-Sx	piix4	
D3433-Sx	i2c-i801	see southbridge drivers
D3434-Sx	i2c-i801	see southbridge drivers
D3441-Sx	i2c-i801	see southbridge drivers
D3446-Sx	i2c-i801	see southbridge drivers

2.2. Southbridge drivers

The most common driver for the SMBus interface of the southbridge is i2c_i801 which comes with the kernel.

You can check if your hardware is supported with the follwoing steps:

First execute an 'lspci' and search for the string 'smb':

```
# lspci | grep -i smb
00:1f.3 SMBus: Intel Corporation 82801JI (ICH10 Family) SMBus Controller
```

Then you need to look a bit closer at this device by using the bus adress as a parameter:

```
# lspci -s 1f.3 -vvv
00:1f.3 SMBus: Intel Corporation 82801JI (ICH10 Family) SMBus Controller
 Subsystem: Fujitsu Technology Solutions Device 114d
 Control: I/O+ Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr+ ...
 Status: Cap- 66MHz- UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- ...
 Interrupt: pin B routed to IRQ 17
 Region 0: Memory at f7307000 (64-bit, non-prefetchable) [size=256]
 Region 4: I/O ports at 1c00 [size=32]
 Kernel driver in use: i801_smbus
```

As you see in the last line of this example in this case the driver that is used is called i801_smbus.

When you have an output for 'Kernel driver in use' then you're halfway done. There is still a chance, that your driver doesn't work because newer systems assign also an ACPI node to that SMBus interface and then you might get a resource conflict between the PCI resources required by the driver and the resources reserved by ACPI.

To find out if you're affected by such an issue have a look at the dmesg output and search for "conflicts".

```
# dmesg | grep conflicts
```

If you see a line that looks like

> ACPI Warning: SystemIO range 0x000000000000F040-0x000000000000F05F
> conflicts with OpRegion 0x000000000000F040-0x000000000000F04F
> (_SB_.PCI0.SBUS.SMBI) (20150619/utaddress-254)

then you are affected by this problem. In that case you need the following boot parameter as a work around: *acpi_enforce_resources=lax*

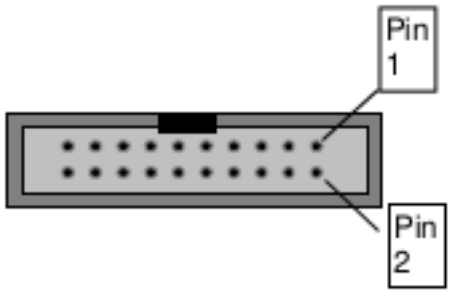
See the explanation of this boot parameter:

```
acpi_enforce_resources= [ACPI]
   { strict | lax | no }
   Check for resource conflicts between native drivers
   and ACPI OperationRegions (SystemIO and SystemMemory
   only). IO ports and memory declared in ACPI might be
   used by the ACPI subsystem in arbitrary AML code and
   can interfere with legacy drivers.
   strict (default): access to resources claimed by ACPI
   is denied; legacy drivers trying to access reserved
   resources will fail to bind to device using them.
   lax: access to resources claimed by ACPI is allowed;
   legacy drivers trying to access reserved resources
   will bind successfully but a warning message is logged.
   no: ACPI OperationRegions are not marked as reserved,
   no further checks are performed.
```

If you use lax, then the kernel is still warning, but your PCI driver is allowed to use the resources and you can access the SMBus.

2.3. Pin assignment of the GPIO connector

The following graphics shows the GPIO pinout.



Pinout of the GPIO connector

The pin assignement of the GPIO connector is defined in the following table.

Table 2.2. GPIO connector pinout

Pin	Signal	Type
1	GPIO0	

Pin	Signal	Type
2	GPIO1	
3	GPIO2	
4	GPIO3	
5	GPIO4	
6	GPIO5	
7	GPIO6	
8	GPIO7	
9	+3.3V	Power
10	GND	Power
11	+3.3V	Power
12	+5V standby	Power
13	Reserved	
14	GND	Power
15	Reserved	
16	GND	Power
17	GND	Power
18	+5V	Power
19	+12V	Power
20	+12V	Power


DRAFT

Using GPIO

3.1. Scan the smbus

First check if the needed kernel driver from the mainboard table is loaded.

```
 # lsmod | grep i2c
i2c_i801               22444  0
i2c_algo_bit           13413  2 igb,i915
```

You should find the driver listed in the mainboard table on the output of the **lsmod** command. In the example above you see the module listed on line 1.

Important

The next important step is to load the i2c-dev driver. This driver is needed to probe the I²C bus.

```
# modprobe i2c-dev
```

Now you need to locate the smbus. The **i2cdetect** will show a list of all available I2C buses.

```
# i2detect -l
i2c-0   i2c             i915 gmbus ssc                          I2C adapter
i2c-1   i2c             i915 gmbus vga                          I2C adapter
i2c-2   i2c             i915 gmbus panel                        I2C adapter
i2c-3   i2c             i915 gmbus dpc                          I2C adapter
i2c-4   i2c             i915 gmbus dpb                          I2C adapter
i2c-5   i2c             i915 gmbus dpd                          I2C adapter
i2c-6   i2c             DPDDC-B                                 I2C adapter
i2c-7   i2c             DPDDC-C                                 I2C adapter
i2c-8   i2c             DPDDC-D                                 I2C adapter
i2c-9   smbus           SMBus I801 adapter at f040              SMBus adapter
```

In the example above the smbus is located on the I²C bus number 9 (shown in line 10).

Now you need to scan the smbus. Use the bus nubmer found above (in our example its 9) as a parameter to **i2cdetect**.

i2cdetect will ask for confirmation (answer "Y" here) and then produce some output like this:

```
# i2cdetect 9
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-9.
I will probe address range 0x00-0x7f.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00 -- -- -- -- -- -- -- 08 -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
30: -- 31 -- 33 -- -- -- -- -- -- -- -- 3c -- 3e --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- 51 -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: 60 61 -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- 73 -- -- -- -- -- -- -- -- -- -- -- --
```

The GPIO controller should be located on bus address `0x3C`, we have an entry on this address so now we need to check what is there.

The bus number (in our example 9) and the bus address `0x3c` are provided as paramters to the **`i2cdump`** command. The output should look like this:

```
# i2cdump 9 0x3c
No size specified (using byte-data access)
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-9, address 0x3c, mode byte
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: ff ff 00 ff XX XX XX XX XX XX XX XX XX XX XX XX    ....XXXXXXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

The addresses `0x00` to `0x03` (the first four bytes) contain the registers and default values of the PCA9554 chip.

3.2. Initializing the GPIO pins

Now you need to load and initialize the GPIO driver for the PCA9554.

```
# modprobe gpio-pca953x
```

If you get an error when executing this command this means, that your distribution kernel doesn't provide a compiled module for the GPIO driver. In that case you have to build your own kernel and set the kernel configuration parameter "CONFIG_GPIO_PCA953x=m". This needs to be done e.g. for Fedora 20, other distributions (e.g. openSUSE 12.3) have that module already compiled into the kernel. You can also check if there is a module named "pca953x.ko" which appears to be the name for the driver used in older kernels.

Once the kernel driver is successfully loaded you need to configure the GPIO device with the following command.

```
# echo pca9554 0x3c > /sys/bus/i2c/devices/i2c-9/new_device
```

Important

In the example above we used the bus number 9 to construct the directory name `i2c-9`.

When this command was successful a new directory **`/sys/class/gpio`** should exist. Lets have a look what we find there:

```
# cd /sys/class/gpio/
# ls -l
total 0
--w------- 1 root root 4096 Jan  7 11:02 export
lrwxrwxrwx 1 root root    0 Jan  7 10:46 gpiochip248 -> ../../devices/
pci0000:00/0000:00:1f.3/i2c-9/9-003c/gpio/gpiochip248
--w------- 1 root root 4096 Jan  7 11:11 unexport
```

There are 2 files (**`export`** and **`unexport`**) and a directory named **`gpiochip248`**. Lets inspect that directory:

```
# ls -l gpiochip248
total 0
-r--r--r-- 1 root root 4096 Jan  7 10:46 base
lrwxrwxrwx 1 root root    0 Jan  7 10:46 device -> ../../../9-003c
-r--r--r-- 1 root root 4096 Jan  7 10:46 label
-r--r--r-- 1 root root 4096 Jan  7 10:46 ngpio
drwxr-xr-x 2 root root    0 Jan  7 10:46 power
lrwxrwxrwx 1 root root    0 Jan  7 10:46 subsystem -> ../../../../../../../class/gpio
-rw-r--r-- 1 root root 4096 Jan  7 10:45 uevent
```

The file **`base`** contains the base number of the first GPIO that can be used. In our example this is 248:

```
# cat gpiochip248/base
248
```

The file **`ngpio`** contains the number of usable GPIO pins. In our example this is 8:

```
# cat gpiochip248/ngpio
8
```

The file **`label`** contains the string `"pca9554"` which we provided in the **`echo`** above.

```
# cat gpiochip248/label
pca9554
```

At this point we know that the GPIO pins will be numbered from 248 to 255 (`base` + (`ngpio`-1). So GPIO pin 0 will have th number 248 in our driver setup and GPIO pin 7 will have the number 255.

Now we have to enable all the connector pins you want to use. So, if we want to use the pin 0 as an output pin you need to submit the following commands:

```
# cd /sys/class/gpio/
# echo 248 > export
```

The command in line 2 creates a new directory **gpio248** which contains the files we need to setup.

```
# ls -l
total 0
--w------- 1 root root 4096 Jan  7 11:32 export
lrwxrwxrwx 1 root root    0 Jan  7 11:32 gpio248 -> ../../devices/pci0000:00/0000:00:1f.3/
i2c-9/9-003c/gpio/gpio248
lrwxrwxrwx 1 root root    0 Jan  7 10:46 gpiochip248 -> ../../devices/
pci0000:00/0000:00:1f.3/i2c-9/9-003c/gpio/gpiochip248
--w------- 1 root root 4096 Jan  7 11:11 unexport

# ls -l gpio248/
total 0
-rw-r--r-- 1 root root 4096 Jan  7 11:32 active_low
lrwxrwxrwx 1 root root    0 Jan  7 11:32 device -> ../../../9-003c
-rw-r--r-- 1 root root 4096 Jan  7 11:32 direction
drwxr-xr-x 2 root root    0 Jan  7 11:32 power
lrwxrwxrwx 1 root root    0 Jan  7 11:32 subsystem -> ../../../../../../../class/gpio
-rw-r--r-- 1 root root 4096 Jan  7 11:32 uevent
-rw-r--r-- 1 root root 4096 Jan  7 11:32 value
```

Now we need to define that pin as an output pin:

```
# echo out > gpio248/direction
```

The file **direction** defines if the pin is used as an output ("out") or an input ("in").

You can use the same initialization scheme for all other GPIO pins. Just echo the number (range 248..255) to the file **export** and a new directory for this GPIO will be created. If you want to erase the configuration of a pin echo its number to the file **unexport**.

3.3. Setting values of output pins

If you have defined one GPIO pin as output you can set its value to 0 or 1 by writing the value to the value file of the according directory:

```
# cd /sys/class/gpio/
# echo 0 > gpio248>/active_low
# echo 1 > gpio248/value
```

The file **active_low** defines if the pin on the connector will be "high" or "low" if the file **value** contains a 1. In the example above the pin will be "high", if you set **active_low** to 1 then the result will be "low". The default setting for **active_low** is 0.

Online resources

4.1. Actual version of this HowTo

The actual version of this HowTo can be downlaoded from *ftp://ftp.ts.fujitsu.com/tobedefined*

4.2. Access GPIO from Linux user space

http://falsinsoft.blogspot.de/2012/11/access-gpio-from-linux-user-space.html

4.3. GPIO interface access conventions on Linux

http://www.michaelhleonard.com/gpio-interface-access-conventions-on-linux/


DRAFT

Appendix A. Revision History

Revision 0-9 2014-01-07 Rainer König

First draft

Revision 1-0 2014-08-06 Rainer König

Updated with more driver details and new board D3313-Sx

Revision 1-1 2016-04-14 Rainer König

Updated for Skylake industrial plattform


DRAFT