**Design document for GPIO**

 

# Outline

This document describes the GPIO control in Linux kernel of MVF TOWER BOARD (XTWR-VF600) with MVF SoC.

# Code to be added

In order to implement the GPIO control, the following source and header file is added to the Linux Kernel Source tree.

* arch/arm/plat-mxc/mvf\_gpio.c

GPIO Control Source File

* arch/arm/plat-mxc/include/mach/mvf\_gpio.h

Definitions for the GPIO Control

# Existing code to be changed

No modification will be made in source code since this is a newly added driver.

However, in order to build, definitions are added to some files including Makefile and Kconfig.

# API of new functions

Functions to be implemented are as follows.

# Initialization

Define the function below as initialization processing of GPIO. This function is executed just one time at the time of kernel initialization.

mvf\_gpio\_init()

* Initialization of management domain
* Initialization of interrupt info
* Initialization of GPIO setting value

# Port Control

Create the following functions for port control.

mvf\_gpio\_direction\_output()

To set output and its value of GPIO

mvf\_gpio\_direction\_input()

To set input of GPIO

mvf\_gpio\_get()

To attain value of port specified for input GPIO.

mvf\_gpio\_set()

To set output value (0or1) of port specified for output GPIO.

# Interrupt

Carry out implementation with irq\_chip struct for the case using GPIO as interrupt.

Create the following as irq\_chip struct member functions.

gpio\_ack\_irq

ACK processing when GPIO is specified as interrupt

gpio\_mask\_irq

MASK processing when GPIO is specified as interrupt

gpio\_unmask\_irq

MASK cancellation processing when GPIO is specified as interrupt

gpio\_set\_irq\_type

Set interrupt type from either EDGE (Up, Down, or Both) or LEVEL (High or Low).

gpio\_set\_wake\_irq

Switching control of Enable and Disable for interrupt

# Expected register settings

This driver controls/specifies GPIO, and controls register for each controller of GPIO, PORT and IOMUX.

Switching GPIO I/O:

Switch I/O of the port by PAD\_PAD register control of IOMUX controller.

Utilize a table to correlate PAD\_PAD register to GPIO number.

Interrupt control of GPIO:

Control GPIO interrupt by PORT\_PCRn register of PORT controller.

I/O of GPIO value:

Utilize GPIO\_PDOR register to set GPIO output value.

Utilize GPIO\_PDIR register to read GPIO input value.

# Expected functionality and usage

GPIO control I/F follows the IMX interface.

However, controllers of PORT and IOMUX will be used for register control other than GPIO, base addresses for them are necessary. Therefore, definition of mvf\_gpio\_port struct to be used as parameter for calling mvf\_gpio\_init() is as follows.

.chip.label: Specify GPIO chip name (A to E)

Ex) “gpio-a”

.gbase: Specify base address for GPIO controller

Ex) GPIOA = 0x40049000

.pbase: Specify base address for PORT controller

Ex) GPIOA = 0x400FF000, GPIOB = 0x400FF040

.ibase: Specify base address for IOMUX

Base address for IOMUX is the same for all.

.ibase = 0x40048000

Other initialization and usage is the same as the GPIO control of mxc.

# Any other pertinent information

None