vxbGpioLib

NAME

vxbGpioLib - vxBus GPIO interfaces module

LAYER

VXBUS

ROUTINES

vxbGpioGetByFdtIndex( ) - get the GPIO ID from FDT node.
vxbGpioBind( ) - bind the device to specified GPIO and configure it
vxbGpioUnBind( ) - unbind the device to specified GPIO
vxbGpioAddCtlr( ) - add GPIO controller to GPIO library
vxbGpioAlloc( ) - allocate speicific GPIO pin
vxbGpioIsFree( ) - check if the GPIO pin is available
vxbGpioFree( ) - free the GPIO pin
vxbGpioSetDir( ) - set GPIO pin's direction
vxbGpioGetDir( ) - get GPIO pin's direction
vxbGpioGetValue( ) - get GPIO pin's value
vxbGpioSetValue( ) - set GPIO pin's value
vxbGpioSetDebounce( ) - set GPIO debounce time
vxbGpioIntConfig( ) - config interrupt trigger mode and polarity
vxbGpioIntConnect( ) - connect GPIO interrupt
vxbGpioIntDisconnect( ) - disconnect device's interrupt
vxbGpioIntEnable( ) - enable device's interrupt
vxbGpioIntDisable( ) - disable device's interrupt
vxbGpioShow( ) - show GPIO controller information

DESCRIPTION

This library provides the GPIO specific interfaces.

Generally, the GPIO are organized as many banks, each bank contain certain number of pins. In this library, we assume that each bank contain same number of pins. And we number the first pin of the first bank as *0*, and the first pin of the second bank as *bank_length*.

for example, a chip have 4 GPIO banks, each have 4 pins, then the pin id will like:

```
          <----pins---->
BANK A    0   1   2   3
BANK B    4   5   6   7
BANK C    8   9   10  11
```

```
            BANK D   12  13  14  15
```

In the library, a bank will be treated as a GPIO controller, that is because in device tree file (.dts) each GPIO bank is treated as a device, so these devices will be instantiated as vxbus device.

This library provides GPIO hog mechanism.

GPIO hog configuration makes the controller to allocate and configure the GPIO automatically in register function (vxbGpioAddCtlr) during bootup, and no other devices can request it.

GPIO hog requires below properties:

```
gpio-hog:       Specify this child node is a GPIO hog.

gpios:          Specify this GPIO hog local gpioId.

Only one of the below properties should be specified.
These properties will be searched in the below order. So if more than
one property is specified, the first match will be used.

input:          Specify to set the GPIO direction as input.

output-low      Specify to set the GPIO direction as output with
                low value.

output-high     Specify to set the GPIO direction as output with
                high value.
```

Usage information:

To use a GPIO, you need call vxbGpioAlloc first to request it (for specific pin), if it success then you can call other APIs to finish specific jobs. And after what you want have done, you should free this pin by call **vxbGpioFree( )**.

For interrupts, if the GPIO controller support it, the driver should supply the callback functions, all user need to do is just knowing the GPIO pin number, and then call vxbGpioIntXxx. The driver should done manage GPIO interrupts. Most GPIO controller share the same interrupt number for one bank, so GPIO drivers should correctly handle this case.

INCLUDE FILES

vxbGpioLib.h

Kernel API Reference: VXBUS Routines

Top

vxbGpioGetByFdtIndex( )

NAME

vxbGpioGetByFdtIndex( ) - get the GPIO ID from FDT node.

LAYER

VXBUS

SYNOPSIS

```
int vxbGpioGetByFdtIndex
    (
    VXB_DEV_ID   pDev,       /* the device node  wish to bind gpio*/
    const char * propName,   /* property name */
    int          index       /* gpio property index */
    )
```

DESCRIPTION

This routine is designed to get the GPIO pin information from FDT node.

the required FDT property may like:

```
gpio-binds = <&gpio2 25 1 1 100
              &gpio2 26 1 0 100
              &gpio2 27 0 0 100>;
```

The **gpio-binds** is the gpio property name which should be described in client device driver's document. And for the property value, you can easily under stand the &gpio2 is the gpio2 controller's phandle. the following numbers are driver specified parameters set which is restricted by **#gpio-cells**. A recommand parameter set may like:
localID direction vlue delay
So this routine will try to get the global GPIO ID from those parameters.

RETURNS

GPIO ID or **ERROR** if failed.

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioBind( )

NAME

vxbGpioBind( ) - bind the device to specified GPIO and configure it

LAYER

VXBUS

SYNOPSIS

```
int vxbGpioBind
    (
    VXB_DEV_ID   pDev,       /* the device node  wish to bind gpio*/
    const char * propName,   /* property name */
    int          index       /* gpio property index */
    )
```

DESCRIPTION

This routine is designed to help users smiplify their code. It is for these case:
The client code only need do very simple operation on GPIO, for example, output a level signal from this GPIO to power up some device.

So, this routine will first get the GPIO config from FDT node, and then tell the GPIO controller driver to operate base on this config. the required FDT property will like:

```
gpio-binds = <&gpio2 25 1 1 100
              &gpio2 26 1 0 100
              &gpio2 27 0 0 100>;
```

Here the &gpio2 is the gpio2 controller's phandle. the following numbers are driver specified parameters set which is restricted by **#gpio-cells**. A recommand parameter set may like:
localID direction vlue delay
Also, each row represents one GPIO. So in this case there are 3 GPIOs are bind.

Note: if bind operation successful, this GPIO will be allocated.

RETURNS

GPIO ID or **ERROR** if failed.

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioUnBind( )

NAME

vxbGpioUnBind( ) - unbind the device to specified GPIO

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioUnBind
    (
    VXB_DEV_ID   pDev,       /* device node should contains the property */
    const char * propName,   /* property name */
    int          index       /* the proper should be a list */
    )
```

DESCRIPTION

This is the reverse routine of vxbGpioBind, but it will not do any config on GPIO again, it just free it and the GPIO controller driver should bring it to a known state.

RETURNS

OK or **ERROR** if failed.

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioAddCtlr( )

NAME

vxbGpioAddCtlr( ) - add GPIO controller to GPIO library

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioAddCtlr
    (
    VXB_GPIOCTRL * pCtrl
    )
```

DESCRIPTION

This routine adds controller instance to GPIO library.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioAlloc( )

NAME

vxbGpioAlloc( ) - allocate speicific GPIO pin

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioAlloc
    (
    UINT32 id
    )
```

DESCRIPTION

This routine allocates specific GPIO pin.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIsFree( )

NAME

vxbGpioIsFree( ) - check if the GPIO pin is available

LAYER

VXBUS

SYNOPSIS

```
BOOL vxbGpioIsFree
    (
    UINT32 id
    )
```

DESCRIPTION

This routine check if the GPIO pin is available.

RETURNS

TRUE or **FALSE**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioFree( )

NAME

vxbGpioFree( ) - free the GPIO pin

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioFree
    (
    UINT32 id
    )
```

DESCRIPTION

This routine frees the speicific GPIO pin.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioSetDir( )

NAME

vxbGpioSetDir( ) - set GPIO pin's direction

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioSetDir
    (
    UINT32 id,
    UINT32 dir
    )
```

DESCRIPTION

This routine sets the speicific GPIO pin's direction.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioGetDir( )

NAME

vxbGpioGetDir( ) - get GPIO pin's direction

LAYER

VXBUS

SYNOPSIS

```
UINT32 vxbGpioGetDir
    (
    UINT32 id
    )
```

DESCRIPTION

This routine gets the specified GPIO pin's current direction.

RETURNS

GPIO_DIR_INPUT/GPIO_DIR_INPUT or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioGetValue( )

NAME

vxbGpioGetValue( ) - get GPIO pin's value

LAYER

VXBUS

SYNOPSIS

```
UINT32 vxbGpioGetValue
    (
    UINT32 id
    )
```

DESCRIPTION

This routine gets the speicific GPIO value.

RETURNS

GPIO_VALUE_LOW/GPIO_VALUE_HIGH or **GPIO_VALUE_INVALID** if failed

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioSetValue( )

NAME

vxbGpioSetValue( ) - set GPIO pin's value

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioSetValue
    (
    UINT32 id,
    UINT32 value
    )
```

DESCRIPTION

This routine sets the speicific GPIO pin's value.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioSetDebounce( )

NAME

vxbGpioSetDebounce( ) - set GPIO debounce time

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioSetDebounce
    (
    UINT32 id,
    UINT32 us
    )
```

DESCRIPTION

This routine sets the GPIO debounce time in us.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIntConfig( )

NAME

vxbGpioIntConfig( ) - config interrupt trigger mode and polarity

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioIntConfig
    (
    UINT32        id,     /* GPIO ID */
    INTR_TRIGER   trig,   /* trigger mode */
    INTR_POLARITY pol     /* polarity */
    )
```

DESCRIPTION

This routine configures interrupt trigger mode and polarity.

The *trig* argument specifies the way of trigger. this argument could be any of the following values: **INTR_TRIGGER_CONFORM** :0 - do not change the way of trigger. **INTR_TRIGGER_EDGE** :1 - edge-triggered interrupt. **INTR_TRIGGER_LEVEL** :2 - level-triggered intrrupt.

The *pol* argument specifies a interrupt polarity. this argument could be any of the following values: **INTR_POLARITY_CONFORM** :0 - do not change interrupt polarity. **INTR_POLARITY_HIGH** :1 - high logical level/rising edge. **INTR_POLARITY_LOW** :2 - low logic level/falling edge. **INTR_POLARITY_BOTH** :3 - valid for edge only, e.g. double edge trigger.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIntConnect( )

NAME

vxbGpioIntConnect( ) - connect GPIO interrupt

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioIntConnect
    (
    UINT32      id,     /* GPIO ID */
    VOIDFUNCPTR pIsr,   /* ISR */
    void *      pArg    /* parameter */
    )
```

DESCRIPTION

This routine connects the given interrupt service routine to the interrupt signal for the specified GPIO pin.

The interrupt number should be presented in DTS for each GPIO controller so there's no need for us to know it. GPIO driver will get it and connect to correct vector. Note the interrupt property like trigger mode, polarity should also be handled in GPIO driver (by reading the DTS info).

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIntDisconnect( )

NAME

vxbGpioIntDisconnect( ) - disconnect device's interrupt

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioIntDisconnect
    (
    UINT32      id,     /* GPIO ID */
    VOIDFUNCPTR pIsr,   /* ISR */
    void *      pArg    /* parameter */
    )
```

DESCRIPTION

This routine disconnects the specified ISR from the interrupt source. And then free the GPIO pin.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIntEnable( )

NAME

vxbGpioIntEnable( ) - enable device's interrupt

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioIntEnable
    (
    UINT32      id,     /* GPIO ID */
    VOIDFUNCPTR pIsr,   /* ISR */
    void *      pArg    /* parameter */
    )
```

DESCRIPTION

This routine enables the specified interrupt on any interrupt controller intervening between the processor and the device. It affects neither the interrupt source nor the processor.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

Kernel API Reference: VXBUS Routines

Top

vxbGpioIntDisable( )

NAME

vxbGpioIntDisable( ) - disable device's interrupt

LAYER

VXBUS

SYNOPSIS

```
STATUS vxbGpioIntDisable
    (
    UINT32      id,     /* GPIO ID */
    VOIDFUNCPTR pIsr,   /* ISR */
    void *      pArg    /* parameter */
    )
```

DESCRIPTION

This routine disables the specified interrupt on the lowest-level interrupt controller between the processor and the device. It does not affect the interrupt source nor the processor.

RETURNS

OK or **ERROR**

ERRNO

N/A

SEE ALSO

vxbGpioLib

---

vxbGpioShow( )

NAME

vxbGpioShow( ) - show GPIO controller information

LAYER

VXBUS

SYNOPSIS

```
VOID vxbGpioShow
    (
    UINT32 verbose
    )
```

DESCRIPTION

This routine will show GPIO controller information. If verbose equals 0, then it just print out pin usage information, else it will call driver's print routine.

RETURNS

N/A

ERRNO

N/A

SEE ALSO

vxbGpioLib