RICO-3288 I2C

Edit by Jason Wang

I2C

1. Introduction¶

The Inter-Integrated Circuit (I2C) is a two wired (SCL and SDA), bi-directional serial bus that provides an efficient and simple method of information exchange between devices. This I2C bus controller supports master mode acting as a bridge between AMBA protocol and generic I2C bus system.

I2C Controller supports the following features:

- Item Compatible with I2C-bus
- AMBA APB slave interface
- Supports master mode of I2C bus
- Software programmable clock frequency and transfer rate up to 400Kbit/sec
- Supports 7 bits and 10 bits addressing modes
- Interrupt or polling driven multiple bytes data transfer
- Clock stretching and wait state generation
- There are two I2C controller in bus sub-system:I2C PMU, and I2C AUDIO. There are four I2C controller in peripheral sub-system: I2C SENSOR, I2C CAM, I2C_TP, and I2C_HDMI.

2. How to Use¶

Define and register the I2C device¶

The structure i2c_client is required to describe the I2C device when registering the I2C device. However, the user only needs to provide the appropriate I2C device information in standard Linux, and Linux will construct the i2c_client structure based on the information provided.

The I2C device information provided by the user is written to the dts file as a node, and it is shown as follows table 2.1.

```
&i2c4 {
        status = "okay";
       egalax_i2c@2a {
                compatible = "eeti,egalax_i2c";
                reg = <0x2a>;
                int-gpios = <&gpio2 GPIO_C1 IRQ_TYPE_EDGE_FALLING>;
                rst-gpios = <&gpio2 GPIO_C0 GPIO_ACTIVE_HIGH>;//for shipment
                };
};
```

Table 2.1

Define the I2C driver¶

Before defining I2C drivers, the user first defines the variables of_device_id and i2c_device_id.The of _ device _ id is used to call the device information defined in the dts file in the driver, it is defined as follows table 2.2.

```
#if LINUX_VERSION_CODE >= KERNEL_VERSION(3,1,0)
static const struct of_device_id egalax_i2c_dt_ids[] = {
        { .compatible = "eeti,egalax_i2c" },
        { }
};
#endif
```

Table 2.2

Define the variable i2c_device_id, it is defined as follows table 2.3.

```
static const struct i2c_device_id egalax_i2c_idtable[] = {
        { "egalax_i2c", 0 },
        { }
};
```

Table 2.3

i2c _driver is shown as follows table 2.4.

```
static struct i2c_driver egalax_i2c_driver = {
        .driver = {
                .name       = "egalax_i2c",
                .owner      = THIS_MODULE,
        #if LINUX_VERSION_CODE >= KERNEL_VERSION(3,1,0)
                .of_match_table = egalax_i2c_dt_ids,
        #endif
        },
        .id_table  = egalax_i2c_idtable,
        .probe             = egalax_i2c_probe,
        .remove           = __devexit_p(egalax_i2c_remove),
#ifndef CONFIG_HAS_EARLYSUSPEND
        .suspend  = egalax_i2c_pm_suspend,
        .resume   = egalax_i2c_pm_resume,
#endif
};
```

Table 2.4.