Application Notes

I2C User Guide

Revision 0.1

Amlogic, Inc.
3930 Freedom Circle
Santa Clara, CA 95054
U.S.A.
www.amlogic.com

Table of Contents

Revision history

Revision	Date	Owner	Changes
0.1	March 12, 2013	Sunny Luo	Draft

1. Overview

There are three I2C masters on M6:

- master A
- master B
- master AO

Every I2C master above has its own controller independently. We can use either one or more or even all of them if needed. This guide describes how to add the I2C master device and how to add a slave device to an I2C bus.

2. Software Operation

2.1 Add I2C master device

To use a I2C adapter, we should add an I2C master device in “platform_devs[ ]” in BSP. In the platform device structure “platform_device”, it includes: name, id, resource and platform data.

2.2 Register I2C slave device

If we use an I2C slave device, we should add it into “i2c_board_info”, where the terms inside will be registered to I2C manager through “i2c_register_board_info()”. It includes device name and slave address. It can also include the irq number and platform data used by the slave driver if needed.

3. Sample

For example, there are two slave devices on master A: GC0308, GSL1680.

3.1 Step 1: Add I2C master device for master A

```
static pinmux_item_t aml_i2c_a_pinmux_item[] = {
    {
        .reg = 5,
        //.clrmask = (3<<24)|(3<<30),
        .setmask = 3<<26
    },
    PINMUX_END_ITEM
};

static struct aml_i2c_platform aml_i2c_plat_a = {
    .wait_count          = 50000, /* ACK wait count */
    .wait_ack_interval   = 5,  /* the polling period for ACK,unit usec, the max ACK waiting time= ....................
                                Wait_count * wait_ack_interval(us) */
    .wait_read_interval  = 5, /* the waiting time after read , unit usec. */
    .wait_xfer_interval  = 5, /* the waiting time after start transfer */
    .master_no = AML_I2C_MASTER_A, /* master number, range:0~ 2 */
    .use_pio = 0,
    .master_i2c_speed   = AML_I2C_SPPED_300K, /* i2c bus speed, range: 10000~400000 */
    .master_pinmux      = {
        .chip_select    = pinmux_dummy_share,
        .pinmux         = &aml_i2c_a_pinmux_item[0]
    }
};

static struct platform_device aml_i2c_device_a = {
    .name  = "aml-i2c", /* device name same with hw i2c driver, do not change it */
    .id  = 0,  /* I2C bus number, range 0~2, */
    .num_resources    = ARRAY_SIZE(aml_i2c_resource_a),
    .resource     = aml_i2c_resource_a,   /* master A register address */
    .dev = {
        .platform_data = &aml_i2c_plat_a,
    },
};

static struct platform_device  *platform_devs[] = {
    ......
    aml_i2c_device_a,
    ......
}
```

3.2 Step 2: Register I2C slave devices to bus 0

```
static struct i2c_board_info __initdata aml_i2c_bus_info_a[] = {
#ifdef CONFIG_VIDEO_AMLOGIC_CAPTURE_GC0308
   {
      I2C_BOARD_INFO("gc0308_i2c",  0x21), /* "gc0308_i2c": device name, same with its drive
                                                0x21: 7bit i2c slave address,  */
      .platform_data = (void *)&video_gc0308_data,  /* gc0308 platform data */
   },
#endif

#ifdef CONFIG_GSL1680_CAPACITIVE_TOUCHSCREEN
   {
      I2C_BOARD_INFO("gsl1680", 0x40), /* "gsl1680": device name.
                                          0x40: 7bit slave address */
   },
#endif
}

static __init void meson_init_machine(void)
{
  ......
  i2c_register_board_info(0, aml_i2c_bus_info_a, ARRAY_SIZE(aml_i2c_bus_info_a));
  ......
}
```