V459 melis

TWI 模块使用说明

1.0

2020.03.12

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.03.12		version 1.0

目录

1. 概述

1.1 编写目的

介绍 melis-RTOS 中 TWI 模块驱动的接口以及使用方法，为 TWI 接口驱动的使用者提供参考

1.2 适用范围

适用于 V459 配套的 melis-RTOS 系统平台

1.3 相关人员

melis-RTOS 驱动层/应用层的开发/使用/维护人员

2. 模块介绍

2.1 模块功能介绍

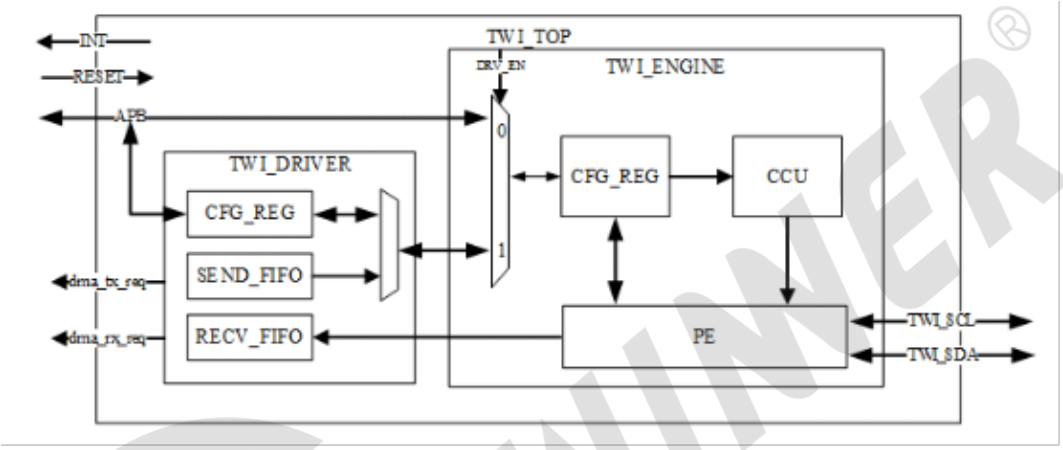

图 1: TWI 控制器

TWI 控制器的框图如上所示。该控制器支持的标准通信速率为 100Kbps，最高通信速率可以达到 400K bps。其中 CPUX 域的 TWI 控制器时钟源来自于 APB2，CPUS 域的 R-TWI 时钟源来自于 APBS。TWI 传输数据的方式包括包传输和 DMA 运输。

V459 平台支持 5 路 TWI，包含 5 路 TWI 与 1 路 S_TWI，其中 S_TWI 作为与 PMU 通信使用。

2.2 相关术语介绍

相关术语	解释说明
TWI	Normal Two Wire Interface，全志平台兼容 I2C 标准协议的总线控制器

2.3 模块配置介绍

TWI 模块寄存器的基本配置位于 source/ekernel/drivers/include/hal/sunxi_hal_twi.h 文件里面，其中包括每个 TWI 的寄存器地址和中断号，部分配置如下：

```
#define SUNXI_TWI0_PBASE 0x05002000 //寄存器地址
#define SUNXI_TWI1_PBASE 0x05002400
#define SUNXI_TWI2_PBASE 0x05002800
#define SUNXI_TWI3_PBASE 0x05002c00
#define SUNXI_S_TWI0_PBASE 0x07081400

//TWI中断
#define SUNXI_GIC_START 32
#define SUNXI_IRQ_TWI0 (SUNXI_GIC_START + 41)
#define SUNXI_IRQ_TWI1 (SUNXI_GIC_START + 42)
#define SUNXI_IRQ_TWI2 (SUNXI_GIC_START + 43)
#define SUNXI_IRQ_TWI3 (SUNXI_GIC_START + 44)
#define SUNXI_IRQ_S_TWI0 (SUNXI_GIC_START + 107)

//TWI的引脚配置
#define TWI_PIN_NUM 2 /*SCK,SDA
#define TWI0_PIN_MUXSEL 5
#define TWI1_PIN_MUXSEL 5
#define TWI2_PIN_MUXSEL 4
#define TWI3_PIN_MUXSEL 5
#define S_TWI0_PIN_MUXSEL 3
#define TWI_DISABLE_PIN_MUXSEL 7
#define TWI_PULL_STATE 1
#define TWI_DRIVE_STATE 0

#define TWI0_SCK GPIOI(3)
#define TWI0_SDA GPIOI(4)
#define TWI1_SCK GPIOI(1)
#define TWI1_SDA GPIOI(2)
#define TWI2_SCK GPIOH(5)
#define TWI2_SDA GPIOH(6)
#define TWI3_SCK GPIOH(13)
#define TWI3_SDA GPIOH(14)
#define S_TWI0_SCK GPIOL(0)
#define S_TWI0_SDA GPIOI(1)
```

2.4 模块源码结构

TWI 模块的源代码位于 ekernel 的 driver 目录下，其中 drv 层和 hal 层分别存放着 TWI 模块的应用层代码和驱动层代码，而 test 目录则存放着 TWI 模块的测试用例代码，具体如下所示：

```
source/ekernel/drivers/hal/source/twi/
|---- hal_twi.c  //驱动层源码
source/ekernel/drivers/drv/source/twi/
|---- drv_twi.c  //应用层源码
source/ekernel/drivers/include/
|---- drv/sunxi_drv_twi.h //应用层头文件
|---- hal/sunxi_hal_twi.h //驱动层头文件
source/ekernel/drivers/test/
|---- test_twi.c  //TWI测试文件
```

2.5 模块结构说明

TWI 模块提供给用户配置的主要包括 TWI 的端口号，TWI 模块的一些控制用法以及发送的消息，所以重点讲解这几个数据结构，想要了解更多的数据结构可以查看 source/ekernel/drivers/include/hal/sunxi_hal_twi.h 文件。

2.5.1 TWI 端口号 twi_port_t

该数据结构主要用来表明 TWI 的编号，用户可以用来调用 TWI 的控制器。具体定义如下：

```
typedef enum
{
    TWI_MASTER_0,          /**< TWI master 0. */
    TWI_MASTER_1,          /**< TWI master 1. */
    TWI_MASTER_2,          /**< TWI master 2. */
    TWI_MASTER_3,          /**< TWI master 3. */
    S_TWI_MASTER_0,        /**< S_TWI master 0. */
    TWI_MASTER_MAX         /**< max TWI master number, \<invalid\> */
} twi_port_t;
```

2.5.2 TWI 消息结构体 twi_msg_t

该数据结构是 TWI 通信时的消息结构，定义每个通信数据的格式，

```
typedef struct twi_msg
{
   uint16_t addr;      //从设备地址
   uint16_t flags;     //flag，可以定义的看下面两个宏
#define TWI_M_RD       0x0001  /* read data, from slave to master（读）
                        * TWI_M_RD is guaranteed to be 0x0001!（写）
                        **/
#define TWI_M_TEN      0x0010  /* this is a ten bit chip address （10位地址）*/
   uint16_t len;         /* msg length */
   uint8_t *buf;       /* pointer to msg data */
} twi_msg_t;
```

2.5.3 TWI 控制结构体 hal_twi_transfer_cmd_t

该数据接口储存了一些用户在调用 twi_control 的时候可以用到的一些参数，具体如下：

```
typedef enum
{
   I2C_SLAVE = 0,  //设置从机地址
   I2C_SLAVE_FORCE = 1, //强制设置从机地址
   I2C_TENBIT = 2, //支持10位地址
   I2C_RDWR = 3    //读写支持
} hal_twi_transfer_cmd_t;
```

其他的关于 TWI 的时钟频率，使用的引脚都已经在软件上面配置了，如果需要更改，可以参照《模块配置介绍》章节进行修改

2.6 驱动层接口

驱动层提供的接口可以通过查看 sunxi_hal_twi.h 获取，下面介绍一些比较重要的接口。

2.6.1 TWI 初始化接口

由于软件上没有自动初始化 TWI，用户可以调用 TWI 初始化接口初始化一个 TWI 控制器，里面包括初始化时钟，中断以及引脚配置等等。

```
twi_status_t hal_twi_init(twi_port_t port);
//port：TWI端口号。成功返回0，失败返回负数
```

2.6.2 TWI 控制接口

用户可以通过该接口来更改 TWI 的一些配置，包括从设备地址以及读写数据等等。

```
twi_status_t hal_twi_control(twi_port_t port, hal_twi_transfer_cmd_t cmd, void *args);
//port:端口号
//cmd：控制参数，包括hal_twi_transfer_cmd_t里面的几个
//args:传入的数据
//如果成功返回0，失败返回负数
```

2.6.3 TWI 发送数据接口

用户可以调用该接口来发送数据，注意在调用这个接口之前最好使用 hal_twi_control 来配置 TWI 的从机地址。

```
twi_status_t hal_twi_write(twi_port_t port, unsigned long pos,const void *buf, uint32_t size);
//port：TWI端口号，pos：偏移量（目前支持1个字节大小），buf：发送的数据，sise：发送数据大小，不包括偏移量
//成功返回0，失败返回负数
```

2.6.4 TWI 接收数据接口

用户可以调用该接口来接收数据，注意在调用这个接口之前最好使用 hal_twi_control 来配置 TWI 的从机地址。

```
twi_status_t hal_twi_read(twi_port_t port, unsigned long pos, void *buf, uint32_t size);
//port：TWI端口号，pos：偏移量（目前支持1个字节大小），buf：接收的数据，sise：接收数据大小，不包括偏移量
//成功返回0，失败返回负数
```

2.6.5 TWI 失能接口

用户可以调用该接口用以失能该 TWI。

```
twi_status_t hal_twi_uninit(twi_port_t port);
//port：TWI端口号。成功返回0，失败返回负数
```

2.7 应用层接口

TWI 应用层使用 rt_device 框架，所以可以通过调用 rt_device 接口来使用 TWI，具体如下。

2.7.1 打开/关闭 TWI 端口

使用标准的文件接口打开/关闭 TWI 接口

```
fd = rt_device_find("twi0");//其他twi类似
rt_device_open(fd, RT_DEVICE_OFLAG_RDWR | RT_DEVICE_FLAG_STREAM); //打开TWI，会自动初始化TWI
rt_device_close((rt_device_t)fd);
```

2.7.2 TWI 控制接口

可以调用 control 函数更改 TWI 配置，如下：

```
rt_err_t rt_device_control(rt_device_t dev, int cmd, void *arg)
//cmd：控制命令，可以参照驱动层控制接口，arg:传入的数据
```

2.7.3 TWI 读写接口

可以调用 read/write 接口用以读写从设备数据, 注意在调用这些接口之前先调用 TWI 控制接口配置从设备地址

```
rt_device_write((rt_device_t)fd, pos, tbuf, len);
rt_device_read((rt_device_t)fd, pos, rbuf, len);
//pos：偏移量（目前支持1个字节大小），rbuf：发送/接收的数据，len：发送数据大小，不包括偏移量
```

3. 模块使用 DEMO

TWI 模块的使用 DEMO 位于 source/ekernel/drivers/test/test_twi.c，如果想要了解可以查看该测试用例。

该测试用例对应用层和驱动层的 TWI 都提供了相应的使用参考。

当系统进入控制台后，可以使用 drvtwi 调用应用层的接口

用户可以使用以下命令调试驱动层的 twi 的接口

```
twi 4(端口号) 0x50(从机地址) 0x01(偏移地址) -r（读）
twi 4(端口号) 0x50(从机地址) 0x01(偏移地址) -w（写） 0x50（写数据）
```

4. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.