ALLWIMER®

V316 spi 接口使用说明书

版权所有©珠海全志科技股份有限公司 2019。保留一切权利。

非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并 不得以任何形式传播。

商标声明

ALLWIMER 全志和其他全志商标均为珠海全志科技股份有限公司的商标。

本文档提及的其他所有商标或注册商标, 由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受全志公司商业合同和条款的约束,本文档中描述的全部 或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定,全志 司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因,本文档内容会不定期进行更新。除非另有约定,本文档仅 作为使用指导,本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

、任何与

全志科技版权所有, 侵权必冤 Copyright © 2019 by Allwinner. All rights reserved

ALLWIMER	V316 spi 接口使用说明书			
	Y. A.	文档履	历 ^{·通}	- Jak
版本号	日期	制/修订人	内容描述	
V1.0	2019-03-30	Allwinner	V316 初始化版本	

The state of the s

全志科技版权所有, 侵权必氮 Copyright © 2019 by Allwinner. All rights reserved



目 录

1	概述															1
	1.1	编写目的	内		1 sensialo							303/30	 •		•	1
/1	1.1	适用范围	围	10							AIV CO					1
A XX	1.3	相关人员	元	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX					1115	XA.						1/
2	模块	介绍 .	- A A				,		,					٠.,	4. (A)	2
	2.1	模块功能	· *** ** ** ** ** ** ** ** ** ** ** ** *				\.\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\							-1/2 -1/2		2
	2.2	相关术证	吾介绍						.1		1.					3
	2.3	模块配置	置介绍			· ; 1	. 1	. R	Ţ				 •			3
	2.3.1	l sys	_config.fex	配置说	明											3
	2.3	2 ³⁵ mei	nuconfig 配	置说明	315/30							18/8/0				5
Х	2,4	源码结构	勾介绍	THE LIVE							SIV.				•	7
\$ ³	接口	描述 .		\$ ^{\$}					1115	XA					•	8
	3.1	设备注册	册接♥	<i>.</i>				4問題	*					٠.,	4	8
	3.1.1	l spi_	register_dri	iver() .			\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	 						7.18	·	8
	3.	1.1.1	函数原型												•	8
	3.	1.1.2	功能描述													8
	3.	1.1.3	返回值													8
	3.	P.1.4	参数说明		an and an							16/3D				8
X	3.1.2	2 spi_	参数说明 _unregister_ 函数原型	driver()	(\$) ^{55°}						SIV CO					9
A XX	3.	1.2.1	函数原型	A A A A A A A A A A A A A A A A A A A					1115	XA TO SERVICE AND ADDRESS OF THE PARTY OF TH						9/

	4=>	37			
_!Q	1				
V	A	ù	W	M	ER
•		7-	- 41		

****	V 316 Sp1 &	安口使用说明书				HX	2 米
A. Contraction of the Contractio	3.1.2.2	功能描述				- X-183	9
	3.1.2.3	参数说明					9
	3.1.2.4	返回值					9
	3.2 数据传	输接口					10
	3.2.1 sp	oi_message_ini	t()				10
	3.2.1.1	函数原型					10
	3.2.1.2	功能描述	×		À		11
A REPUBLICATION OF THE PARTY OF	3.2.1.3	参数说明					11
\$ ⁷ `	3.2.1.4	返回值】:	无			. 5	11
	3.2.2 sp	oi_message_ado	d_tail()				11
	3.2.2.1	函数原型		1. N.N.			11
	3.2.2.2	功能描述					11
	3.2.2.3	参数说明					11
	3.2.2.4	返回值	The second				11
	3.2.3 sp	oi_sync()	<i>\$</i>		À		12
A THE TANK	3.2.3.1	函数原型	./				12
\$7	3.2.3.2	功能描述				. 5	12
	3.2.3.3	参数说明					12
	3.2.3.4	返回值					12
4	demo						13
		/mtd/device/m2	25p80.c				13
5	Declaration				The Land		14
	PC/ /						"

表目录

A Hilling High Land State of the State of th



图目录

2-1	Linux SPI 体系结构图	2
2-2	Device Drivers 配置选项	5
2-3	SPI support 配置选项	6
	SUNXI SPI Controller 配置法辅	7

S. Lift Hill by the beautiful of the state o

A TO SEE LEEVE

CANAL PROPERTY OF THE PARTY OF

A STATE OF THE STA



概述

1.1 编写目的

介绍 Linux 内核中 SPI 子系统的接口及使用方法,为 SPI 设备驱动开发提供参考

适用范围

1.3 相关人员

一八贝 SPI设备驱动、SPI总线驱动的开发/维护人员。



2 模块介约

2.1 模块功能介绍

Linux 中 SPI 体系结构分为三个层次,如下图所示:

| SPI Device Driver | SPI Controller | SPI device |

- 1. 用户空间,包括所有使用 SPI 设备的应用程序;
- 2. 内核,也就是驱动部分;
- 3. 硬件, 指实际物理设备, 包括了 SPI 控制器和 SPI 外设。

其中,Linux 内核中的 SPI 驱动程序仅支持主设备,逻辑上又可以分为 3 个部分: 1. SPI 核心 (SPI Core): 实现对 SPI 总线驱动及 SPI 设备驱动的管理;

2. SPI 总线驱动 (SPI Master Driver): 针对不同类型的 SPI 控制器,实现对 SPI 总线访问的具体方法;



3. SPI 设备驱动(SPI Device Driver): 针对特定的 SPI 设备,实现具体的功能,包括 read, write 以及 ioctl 等对用户层操作的接口。SPI 总线驱动主要实现了适用于特定 SPI 控制器的总线读写方法,并注册到 Linux 内核的 SPI 架构,SPI 外设就可以通过 SPI 架构完成设备和总线的适配。但是总线驱动本身并不会进行任何的通讯,它只是提供通讯的实现,等待设备驱动来调用其函数。SPI Core 的管理正好屏蔽了 SPI 总线驱动的差异,使得 SPI 设备驱动可以忽略各种总线控制器的不同,不用考虑其如何与硬件设备通讯的细节。

2.2 相关术语介绍

	<i>₹//</i> = 1	V// ~ '
术语	解释说明	A A A A A A A A A A A A A A A A A A A
Sunxi	指 Allwinner 的一系列 SC	C 硬件平台
SPI Y	Serial Peripheral Interface,	同步串行外设接口
SPI Master	SPI 主设备	160
SPI Device	指 SPI 外部设备	MU

2.3 模块配置介绍

2.3.1 sys_config.fex 配置说明

在不同的 Sunxi 硬件平台中, SPI 控制器的数目也不同, 但对于每一个 SPI 控制器来说, 在 sys config.fex 中配置参数相似, 如下:

```
[spi0]
[spi0]
spi0 used
spi0 cs number = 2
spi0 cs bitmap = 3
spi0_cs0
           = port:PC01<4><1><default><default>
           = port:PC10<4><1><default><default>
spi0 cs1
spi0 sclk
           = port:PC00<4><default><default>
spi0 mosi
           = port:PC02<4><default><default>
spi0 miso
            = port:PC03<4><default><default>
spi0 hold
           = port:PC14<4><default><default>
spi0 wp
           = port:PC13<4><default><default><default>
```



```
[spi0_suspend]

spi0_cs0 = port:PC01<7><default><default><
spi0_cs1 = port:PC10<7><default><default><default>
spi0_sclk = port:PC00<7><default><default><default>
spi0_mosi = port:PC02<7><default><default><default><
spi0_miso = port:PC03<7><default><default><default><
spi0_miso = port:PC14<7><default><default><default><
spi0_hold = port:PC13<7><default><default><default><
spi0_wp = port:PC13<7><default><default><default><
```

其中:

spi0_used: 为 1 表示使能, 0 表示不使能; spi0_cs_number: SPI 控制器支持的片选数; spi0_cs_bitmap: 由于 SPI 控制器支持多个 CS, 这一个参数表示 CS 的掩码; spi0_cs0、spi0_sclk、spi0_mosi 和 spi0_miso 用于配置相应的 GPIO。 spi0_suspend: 休眠时 PIN 脚的状态。

对于 SPI 设备,还需要通过以下参数配置 SPI board info,这些信息会通过 SPI 子系统的接口 spi register board info()在 SPI 总线驱动初始化前就注册到内核中。

```
[spi0/spi_board0]
compatible = "m25p80"
spi-max-frequency = 100000000
reg = 0
spi-rx-bus-width = 4
spi-tx-bus-width = 4
;spi-cpha
;spi-cpol
;spi-cs-high
```

其中:

spi0/spi_board0: 表示挂在总线 spi0 下的设备 spi_board0; compatible: SPI 设备的名字,在做总线适配时会用到; spi-max-frequency: SPI 设备的最大传输速度,单位是 Hz; reg: SPI 设备的片选线,取决于硬件的 CS 连线; spi-rx-bus-width: nor 收数据模式,2 为双线,4 为四线,不定义或 1 为单线; spi-tx-bus-width: nor 写数据模式,2 为双线,4 为四线,不定义或 1 为单线; 其它可选配置参数: spi-cpha, spi-cpol, spi-cs-high。



2.3.2 menuconfig 配置说明

在命令行中进入内核 linux 目录, 执行 make ARCH=arm menuconfig(64 位平台执行 make ARCH=arm64 menuconfig)进入配置主界面,并按以下步骤操作。

选择 Device Drivers 选项进入下一级配置,如下图所示:

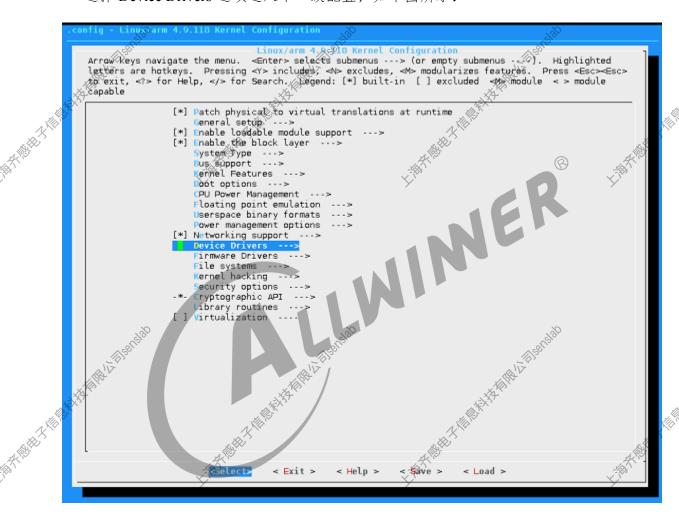


图 2-2: Device Drivers 配置选项

选择 SPI support 选项,进入下一级配置,如下图所示:



```
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <N> modularizes features. Press <Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <N> module <> module
                                                                                                                    Press «Esc»«Esc»
                           Generic Driver Options --->
                           Bus devices
                      < > Connector - unified userspace <-> kernelspace linker ----
                      <*> Memory Technology Device (MTD) support --->
-*- Device Tree and Open Firmware support --->
                      Parallel port support ----
                      [*] Block devices
                      < > NVMe Target support
                           Misc devices
                           SCSI device support
                      < > Serial ATA and Parallel &FA drivers (libata) ----
[ ] Multiple devices driven support (RAID and LVM) ---
[*] Network device support --->
                      [ ] Open-Channel SSD target support
                           Input device support --->
                           Character devices
                           I2C support
                      < > SPMI support
                      < > HSI support
                           PPS support --->
                           PTR clack support --->
                           Pin controllers --->
                      [*] GPIO Support --->
                     < > Dallas's 1-wire support ---
[ ] Adaptive Voltage Scaling class support
[ ] Board level reset or power off ---
[*] Power supply class support --->
                      < > Hardware Monitoring support
                      < > Generic Thermal sysfs driver
                      [*] Watchdog Timer Support
Sonies Silicon Backplane
                            Broadcom specific AMBA
```

图 2-3: SPI support 配置选项

选择 SUNXI SPI Controller 选项,可选择直接编译进内核,也可编译成模块。如下图所

示:

上海大樓上上

A THE PARTY OF THE



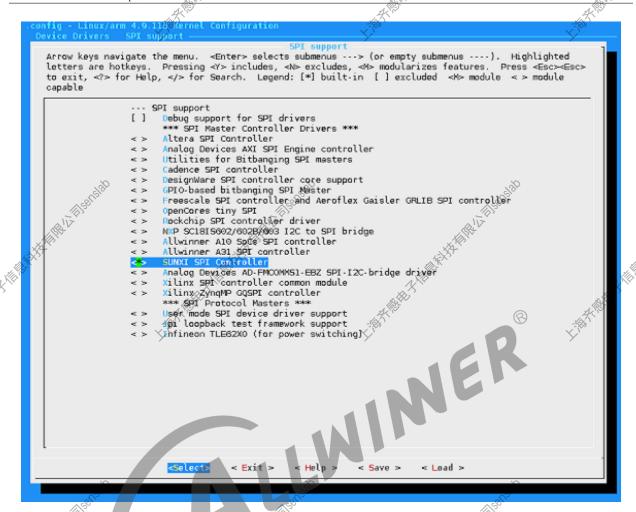


图 2-4: SUNXI SPI Controller 配置选项

2.4 源码结构介绍

SPI 总线驱动的源代码位于内核在 drivers/spi 目录下:

全志科技版权所有,侵权必究



3 接口描述

3.1 设备注册接口

接口定义在 include/linux/spi/spi.h, 主要包含 spi_register_driver 与 spi_unregister_driver 接口, 其中给出了快速注册的 I2C 设备驱动的宏 module spi driver(), 定义如下:

MINER

#define module_spi_driver(__spi_driver)\
module_driver(__spi_driver, spi_register_driver,\
spi_unregister_driver)

3.1.1 spi_register_driver()

3.1.1.1 函数原型

int spi_register_driver(struct spi_driver *sdrv)

3.1.1.2 功能描述

注册一个 SPI 设备驱动。

3.1.1.3 返回值

返回0表示成功,返回其他值表示失败。

3.1.1.4 参数说明

sdrv, spi_driver 类型的指针, 其中包含了 SPI 设备的名称、probe 等接口信息。SPI 设备驱动可能支持多种型号的设备。可以在.id_table 中给出所有支持的设备信息。其中, 结构 spi_driver 的定义如下:



```
struct spi_device_id {
  char name[SPI NAME SIZE];
  kernel_ulong_t driver_data /* Data private to the driver */
       attribute ((aligned(sizeof(kernel ulong t))));
};
struct spi driver {
  const struct spi_device_id *id_table;
         (*probe)(struct spi device *spi);
        (*remove)(struct spi_device *spi);
  int
  void
         (*shutdown)(struct spi_device *spi)
 int
         (*suspend)(struct spi device *spi, pm message t mesg);
         (*resume)(struct spi device *spi);
  int
  struct device driver driver;
```

3.1.2 spi_unregister_driver()

函数原型 3.1.2.1

void spi_unregister_driver(struct spi_driver *sdrv)

3.1.2.2 功能描述

注销一个 SPI 设备驱动。

参数说明 3.1.2.3

sdrv, spi driver 类型的指针, 其中包含了 SPI 设备的名称、probe 等接口信息。



3.2 数据传输接口

SPI 设备驱动使用 "struct spi_message" 向 SPI 总线请求读写 I/O。一个 spi_message 中包含了一个操作序列,每一个操作称作 spi_transfer,这样方便 SPI 总线驱动中串行的执行一个个原子的序列。

spi message 和 spi transfer 的定义也在 spi.h 中:

```
struct spi_transfer {
  const void *tx buf;
  void
          *rx buf;
  unsigned len;
                                         dma_addr_t tx_dma;
  dma_addr_t rx_dma;
  unsigned cs change:1;
       bits per word;
  u16
       delay usecs;
        speed hz;
  u32
  struct list head transfer list;
};
             *spi;
is_dma_mapped:1;
complete)(void *
struct spi\message {
  struct list head transfers;
 struct spi device *spi;
  unsigned
            (*complete)(void *context)
  void
  void
            *context;
  unsigned
              actual_length;
         status;
  struct list head queue;
  void
            *state;
};
```

3.2.1 spi_message_init()

3.2.1.1 函数原型



void spi_message_init(struct spi_message *m)

3.2.1.2 功能描述

初始化一个 SPI message 结构,主要是清零和初始化 transfer 队列。

- - m, spi message 类型的指针
- 返回值】: 无 3.2.1.4
- 3.2.2 spi_message_add_tail()
- 3.2.2.1 函数原型

void spi message add_tail(struct spi_transfer *t, struct spi_message *m)

3.2.2.2 功能描述

向 SPI message 中添加一个 transfer。

参数说明 3.2.2.3

- t,指向待添加到 SPI transfer 结构; m, spi message 类型的指针。
- 3.2.2.4 返回值

无



3.2.3 spi_sync()

3.2.3.1 函数原型

nt spi_sync(struct spi_device *spi, struct spi_message *message)

3.2.3.2 功能描述

启动、并等待 SPI 总线处理完指定的 SPI message。

3.2.3.3 参数说明

spi, 指向当前的 SPI 设备; m, spi_message 类型的指针, 其中有待处理的 SPI transfer 队列。

3.2.3.4 返回值

0. 成功: 小于0. 失败

全志科技版权所有, 侵权必氮 Copyright © 2019 by Allwinner. All rights reserved



4 demo

4.1 drivers/mtd/device/m25p80.c

此 SPI 设备是一个 NorFlash, 需要 MTD 架构的支持。其中调用调用 spi_register_driver() 注册 SPI 驱动, 用户主要工作是实现 probe 函数和要通过 SPI message 实现数据的读写。

```
static struct platform_driver sunxi_spi_driver = {
  .probe = sunxi spi probe, 4
  .remove = sunxi spi remove,
  .driver = {
    .name = SUNXI SPI DEV NAME,
    .owner = THIS MODULE,
    .pm = SUNXI SPI DEV PM OPS,
    .of match table = sunxi spi match,
 },
};
static int __init sunxi_spi_init(void)
  return platform_driver_register(&sunxi_spi_driver);
static void __exit sunxi_spi_exit(void)
  platform_driver_unregister(&sunxi_spi_driver);
subsys initcall(sunxi spi init);
module exit(sunxi spi exit);
```



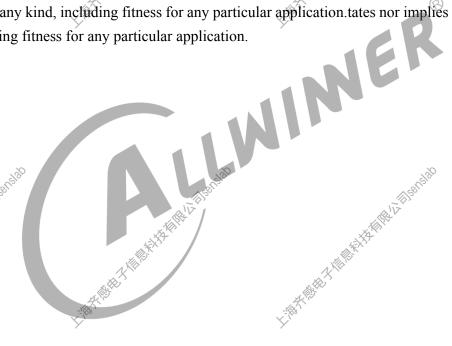


Declaration

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 datasheet 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.



·特尔·

A THE VIEW OF THE PARTY OF THE

AF HARING TO SEE STON