GK8602X 芯片系列

芯片手册

深圳市国科微半导体有限公司

2015-06-09

v0.1

1 产品概述

1.1 介绍

GK8602A 是针对高清行车记录仪产品开发的一款低功耗的 SOC 芯片，支持 1080@30p 图像的多码流 H.264 格式编码，及高质量的 ISP 处理，支持主流的多种 sensor，内置 DDR2，内置 Audio Codec 以及 Efuse 加密模块。

1.2 系统功能

处理器内核

- 嵌入式 ARM1176ZJFS 处理器
- 频率最高可达 600MHz
- 集成 MMU 单元
- 16K ICache 和 16KDCache

ISP

- 支持 3A（AE/AWB/AF）
- 坏点校正
- 图像增强
- 暗电流校正
- 数字宽动态对比增强（WDR）
- 3D 降噪功能
- 支持镜头畸变校正
- 提供 ISP tuning tool

视频编码能力

- 支持 Baseline/Main/High profile L4.1 H.264 编码
- 支持 JPEG Baseline 格式编码
- 支持多路码流输出，最高支持 1080p@30
- CBR/VBR 码率控制
- 支持对感兴趣区域（ROI）编码
- 可配置 GOP 长度，支持 I/P/B 帧

音频编解码

- 支持多种音频编解码协议 G.711、G.726、AAC、MP3
- 支持回波抵消功能

视频接口

- 支持 8/10/12 bit RGB Bayer 输入，时钟频率最高 81MHz
- 支持与 Sony、Aptina、OmniVision、Panasonic 等主流 CMOS Sensor 对接
- 兼容多种 sensor 电平
- 支持 1 路 CVBS 输出接口
- 支持 OSD 图像叠加

音频接口

- 内部集成 Audio Codec
- 24bit 数字音频 I2S 输入输出接口

外围接口

- 集成高精度 32.768K RTC
- 3 个 UART 接口，其中 1 个可转换 RS485
- 2 个主从 SPI 接口
- 2 个 I2C 接口，其中 1 个自适应 Sensor 电平
- 4 个 PWM 接口
- 集成 2 通道 10bit SAR-ADC
- 1 个 USB 2.0 OTG 接口
- 1 个 SDIO 2.0 接口
- 28 个 GPIO，可根据需求任意配置

外部存储接口

- 内部集成 1Gb DDR2，16bit DDR2@800MHz
- SPI Nor/Nand Flash 接口，支持 1/2/4 线模式
- 可选择从 SPI Nor 或者 SPI Nand 启动

加解密

- 内部集成 128KB Efuse
- 支持 DES/3DES/AES 加密算法
- 支持客户自定义 Key 的加密启动

芯片规格

- 芯片典型功耗 900mW（包括 DDR 功耗）
- 内核电压 1.1V，IO3.3V，DDR2 1.8V
- 工作温度 -20 ~ +85
- TFBGA 228pin 封装，0.65 管脚间距，
- 11mm x 11mm

1.3 应用场景

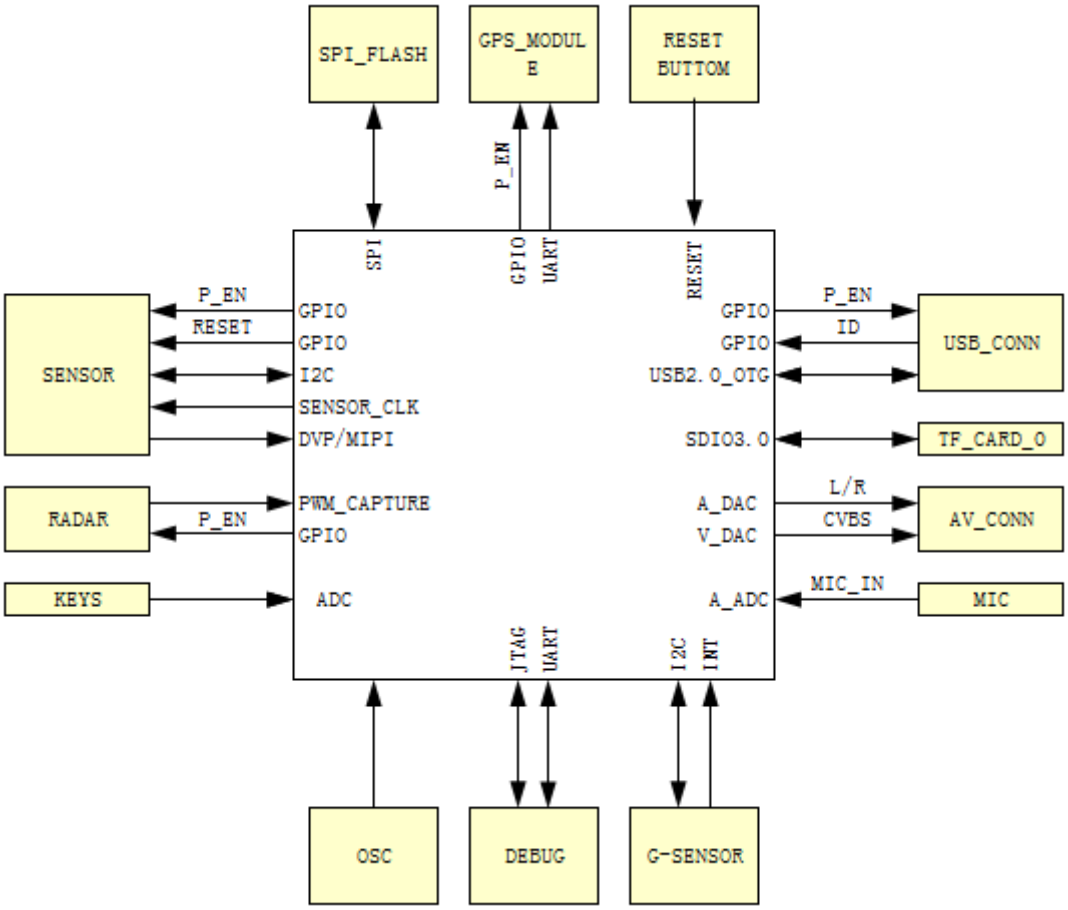


Figure 1.3.1 应用场景示意图

1.4 SOC 框图

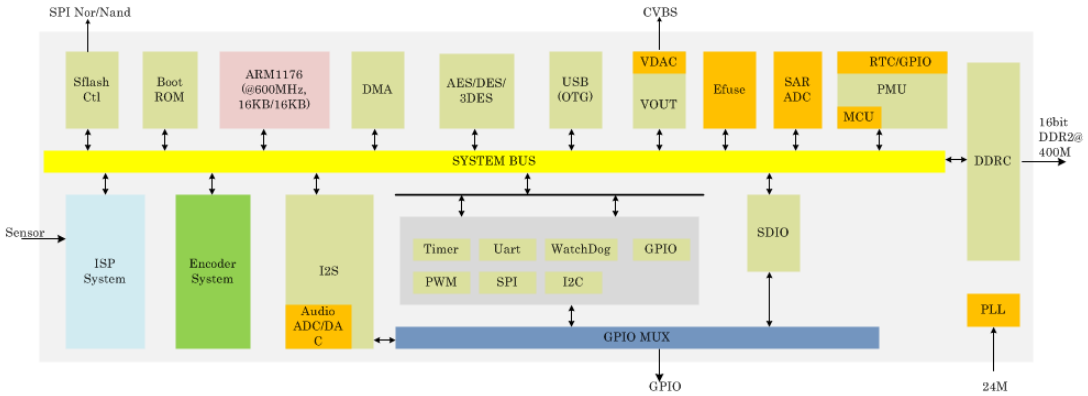


Figure 1.4.1 SOC 系统框架图

2 硬件特性

2.1 封装与管脚

2.1.1 封装

GK8602A 芯片采用 TFBGA 228 封装，封装尺寸为 11mm*11mm ，管脚间距为 0.65mm，详细封装参见图。

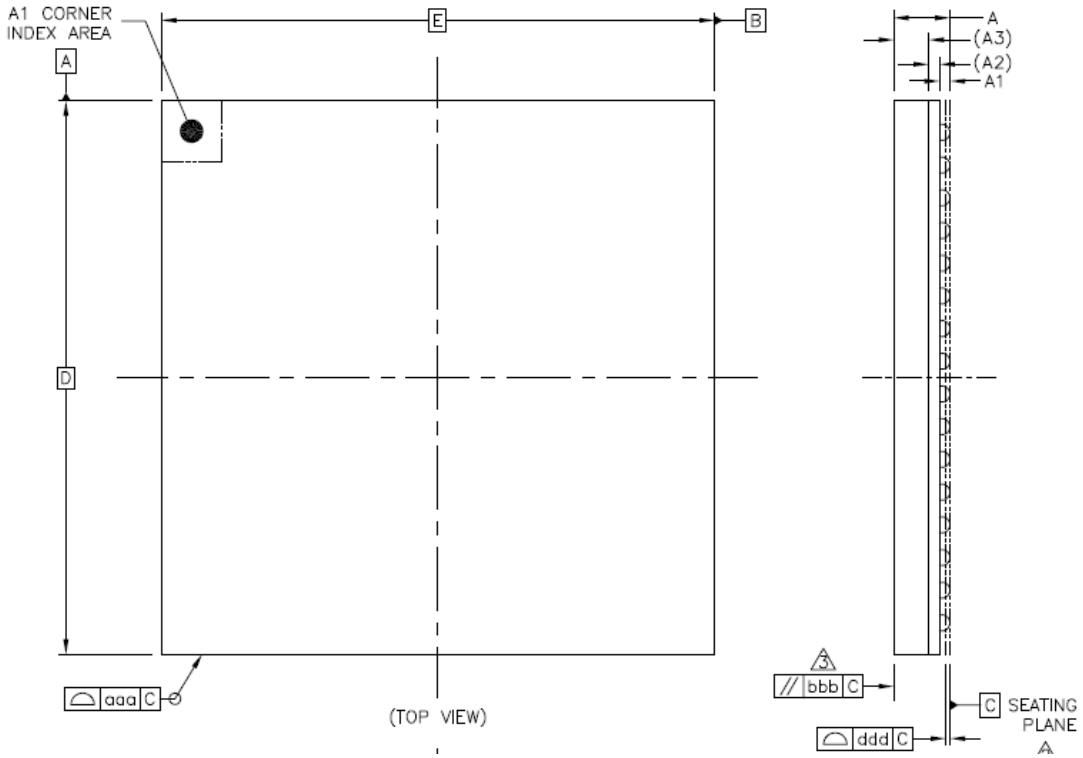


Figure 2.1.1 芯片封装顶视图

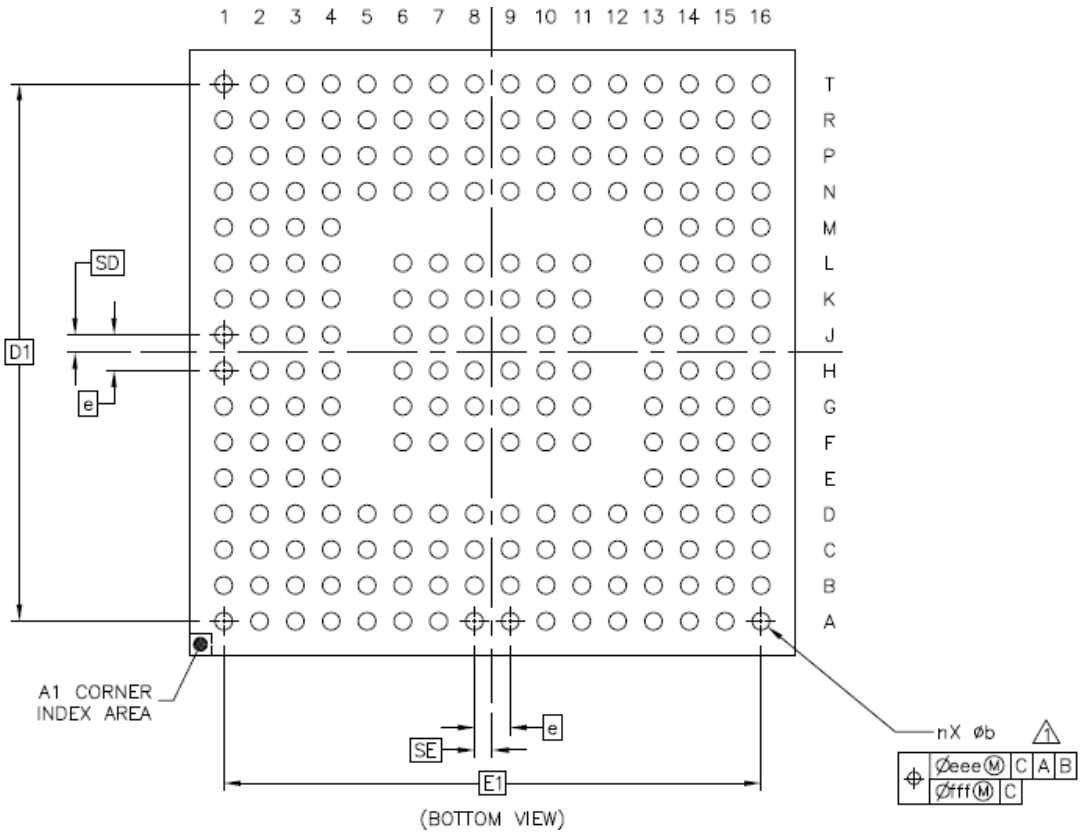


Figure 2.1.2 芯片封装底视图

	SYMBOL	COMMON DIMENSIONS		
		MIN.	NOR.	MAX.
TOTAL THICKNESS	A	---	---	1.2
STAND OFF	A1	0.16	---	0.26
SUBSTRATE THICKNESS	A2		0.21	REF
MOLD THICKNESS	A3		0.7	REF
BODY SIZE	D		11	BSC
	E		11	BSC
BALL DIAMETER			0.3	
BALL OPENING			0.275	
BALL WIDTH	b	0.27	---	0.37
BALL PITCH	e		0.65	BSC
BALL COUNT	n		228	
EDGE BALL CENTER TO CENTER	D1		9.75	BSC
	E1		9.75	BSC
BODY CENTER TO CONTACT BALL	SD		0.325	BSC
	SE		0.325	BSC
PACKAGE EDGE TOLERANCE	aaa		0.1	
MOLD FLATNESS	bbb		0.1	
COPLANARITY	ddd		0.08	
BALL OFFSET (PACKAGE)	eee		0.15	
BALL OFFSET (BALL)	fff		0.08	

Table 2.1.1 封装参数说明表

2.1.2 管脚分布

GK8602A 的管脚有 228 个，管脚分布见图

	1	2	3	4	5	6	7	8
A	AVDD3P2_AUX-425	VSS	NC	NC	NC	NC	NC	FSOURCE-382
B	PAD_SAR_KEY2-4	PAD_SAR_KEY1-3	NC	NC	NC	NC	NC	VSS
C	PAD_SPCLK-13	PAD_CLK_SO-12	PAD_SFIELD-11	NC	NC	NC	NC	VSS
D	VDDIO33_SR	VDDIO33_SR	PAD_SVSYNC-14	VSS	VDDIO33	VDDIO33	VDD11	VDD11
E	PAD_SDATA[11]-21	PAD_SDATA[10]-22	PAD_SHSYNC-20	VSS				
F	PAD_SDATA[9]-28	PAD_SDATA[8]-29	VSS	VDD11		VSS	VSS	VSS
G	PAD_SDATA[7]-30	PAD_SDATA[6]-31	VSS	VDD11		VSS	VSS	VSS
H	PAD_SDATA[5]-36	PAD_SDATA[4]-37	NC	VDD11		VSS	VSS	VSS

Figure 2.1.3 管脚分布图 Part1

9	10	11	12	13	14	15	16	
NC	AVDD33_USB	PAD_DP-370	AVDD33	PADA_IDAC_OUT-365	NC	NC	VSS	A
NC	AVDD_MPLL	PAD_DM-371	AVDD11_USB	NC	NC	PAD_SF_HOLD-343	PAD_SF_WP-336	B
VSS	VSS	VSS	VSS	NC	NC	PAD_SF_CLK-342	PAD_SF_SO-335	C
VSS	VSS	VDD11	VDD11	VDDIO33	NC	PAD_SF_SI-337	PAD_GPIO[0]-332	D
				VDDIO33	PAD_GPIO[1]-320	PAD_JTAG_TRSTN-331	PAD_JTAG_TDI-330	E
VSS	VSS	VSS		VDDIO33	PAD_JTAG_TMS-329	PAD_JTAG_TCK-328	PAD_JTAG_TDO-322	F
VSS	VSS	VSS		VSS	PAD_GPIO[2]-323	PAD_GPIO[3]-321	NC	G
VSS	VSS	VSS		VSS	NC	NC	NC	H

Figure 2.1.4 管脚分布图 Part2

J	PAD_SDATA[3]-38	PAD_SDATA[2]-39	NC	VSS		VSS	VSS	VSS
K	PAD_SDATA[1]-44	PAD_SDATA[0]-45	PAD_GPIO[27]-48	VSS		VSS	VSS	VSS
L	PAD_GPIO[26]-53	PAD_GPIO[25]-54	PAD_GPIO[24]-55	VDDIO33		VSS	VSS	VSS
M	PAD_GPIO[23]-56	PAD_GPIO[22]-60	PAD_GPIO[21]-61	VDDIO33				
N	PAD_GPIO[20]-62	PAD_SEL_18-66	PAD_GPIO[17]-67	PAD_GPIO[16]-68	VDD11	VDD11	DVDD11	DVDD11
P	PAD_GPIO[15]-69	PAD_GPIO[14]-70	PAD_GPIO[13]-75	PAD_GPIO[5]-87	PAD_GPIO[4]-88	VSS	VSS	VSS
R	PAD_GPIO[12]-76	PAD_GPIO[11]-77	PAD_GPIO[8]-82	PAD_GPIO[6]-86	VDDIO18	AVDD_PLL	VDDIO18	VDDIO18
T	VSS	PAD_GPIO[10]-78	PAD_GPIO[9]-81	PAD_GPIO[7]-83	VDDIO18	AVSS_PLL	**DDRVREF**	VDDIO18
	1	2	3	4	5	6	7	8

Figure 2.1.5 管脚分布图 Part3

VSS	VSS	VSS		VSS	PAD_GPIO[18]-313	PAD_GPIO[19]-312	VDD11_PLL	J
VSS	VSS	VSS		VDD11	VSS	VSS_PLL	VDD33_PLL	K
VSS	VSS	VSS		VDD11	VSS	NC	NC	L
				VSS	AVDD33E	NC	NC	M
VSS	VDD11	VDD11	VSS	VSS	VSS	PAD_MICBIAS-279	VABB33	N
VSS	PAD_TEST_MODE-240	PAD_PMU_GPIO[4]-245	PAD_PMU_GPIO[2]-247	PAD_PMU_GPIO[0]-249	PAD_EXT_RSTN-250	PAD_MICINN-272	PAD_AUDREF-277	P
rtc_XI-234	VSS	osc_XI-238	PAD_PMU_GPIO[3]-246	PAD_PMU_GPIO[1]-248	PAD_VOPR_LINEO-265	PAD_VOPL_LINEO-266	PAD_MICINP-270	R
rtc_XO-233	VSS	osc_XO-237	VDD11AON	VDDIO33AON	PAD_VOMR_LINEO-263	PAD_VOML_LINEO-264	VSS	T
9	10	11	12	13	14	15	16	

Figure 2.1.6 管脚分布图 Part4

2.1.3 管脚描述

2.1.3.1 全局及测试接口管脚

PIN	管脚名称	方向	电压(V)	描述
R11	OSC_XI	I	3.3	系统时钟（24M）
T11	OSC_XO	O	3.3	系统时钟输出
R9	RTC_XI	I	3.3	RTC 时钟（32.768K）
T9	RTC_XO	O	3.3	RTC 时钟输出
P10	PAD_TEST_MODE	I	3.3	1：测试模式　0：工作模式
P14	PAD_EXT_RSTN	I	3.3	全局系统复位（低有效）

2.1.3.2 Sensor 接口管脚

PIN	管脚名称	方向	电压(V)	描述
E1	PAD_SDATA[11]	I	3.3/2.5/1.8	Sensor 并口数据信号 11
E2	PAD_SDATA[10]	I	3.3/2.5/1.8	Sensor 并口数据信号 10
F1	PAD_SDATA[9]	I	3.3/2.5/1.8	Sensor 并口数据信号 9
F2	PAD_SDATA[8]	I	3.3/2.5/1.8	Sensor 并口数据信号 8
G1	PAD_SDATA[7]	I	3.3/2.5/1.8	Sensor 并口数据信号 7
G2	PAD_SDATA[6]	I	3.3/2.5/1.8	Sensor 并口数据信号 6
H1	PAD_SDATA[5]	I	3.3/2.5/1.8	Sensor 并口数据信号 5
H2	PAD_SDATA[4]	I	3.3/2.5/1.8	Sensor 并口数据信号 4
J1	PAD_SDATA[3]	I	3.3/2.5/1.8	Sensor 并口数据信号 3

J2	PAD_SDATA[2]	I	3.3/2.5/1.8	Sensor 并口数据信号 2
K1	PAD_SDATA[1]	I	3.3/2.5/1.8	Sensor 并口数据信号 1
K2	PAD_SDATA[0]	I	3.3/2.5/1.8	Sensor 并口数据信号 0
C3	PAD_SFIELD	I	3.3/2.5/1.8	
C2	PAD_CLK_SO	O	3.3/2.5/1.8	Sensor 工作参考时钟
C1	PAD_SPCLK	I	3.3/2.5/1.8	Sensor 并口数据同步时钟
D3	PAD_VSYNC	I	3.3/2.5/1.8	Sensor 并口数据场同步信号
E3	PAD_HSYNC	I	3.3/2.5/1.8	Sensor 并口数据行同步信号
N2	PAD_SEL_18	I	3.3	Sensor 工作电压选择 1：1.8V 0：2.5V/3.3V

2.1.3.3 JTAG 接口管脚

PIN	管脚名称	方向	电压(V)	描述
F16	PAD_JTAG_TDO	O	3.3	JTAG 数据输出
F15	PAD_JTAG_TCK	I	3.3	JTAG 时钟输入
F14	PAD_JTAG_TMS	I	3.3	JTAG 模式选择输入
E16	PAD_JTAG_TDI	I	3.3	JTAG 数据输入
E15	PAD_JTAG_TRSTN	I	3.3	JTAG 复位输入

2.1.3.4 Efuse 管脚

PIN	管脚名称	方向	电压(V)	描述
A8	FSOURCE	I	Analog 1.8	Efuse 烧写时接 1.8V，读取时接地

2.1.3.5 低速传感器接口管脚

PIN	管脚名称	方向	电压(V)	描述
B2	PAD_SAR_KEY1	I	Analog 3.3	传感器通道 1
B1	PAD_SAR_KEY2	I	Analog 3.3	传感器通道 2

2.1.3.6 Video DAC 管脚

PIN	管脚名称	方向	电压(V)	描述
A13	PADA_IDAC_OUT	O	Analog 3.3	CVBS 模拟信号视频输出

2.1.3.7 Audio Codec 管脚

PIN	管脚名称	方向	电压(V)	描述
R16	PAD_MICINP	O	Analog 3.3	麦克风差分输入
P15	PAD_MICINN	I	Analog 3.3	麦克风差分输入
N15	PAD_MICBIAS	O	Analog 3.3	麦克风偏置电压
T14	PAD_VOMR_LINE0	O	Analog 3.3	音频右声道差分输出
R14	PAD_VOPR_LINE0	O	Analog 3.3	音频右声道差分输出
T15	PAD_VOML_LINE0	O	Analog 3.3	音频左声道差分输出
R15	PAD_VOPL_LINE0	O	Analog 3.3	音频左声道差分输出
P16	PAD_AUDREF	I	Analog 3.3	音频参考电压，外接 2.2uF 电容

2.1.3.8 SFLASH 管脚

PIN	管脚名称	方向	电压(V)	描述
C16	PAD_SF_SO	I/O	3.3	数据输出信号，两线/四线模式下为数据双向信号
B16	PAD_SF_WP	I/O	3.3	写保护信号，四线模式下为数据双向信号

D15	PAD_SF_SI	I/O	3.3	数据输入信号，两线/四线模式下为数据双向信号
C15	PAD_SF_CLK	O	3.3	SFLASH 时钟信号
B15	PAD_SF_HOLD	I/O	3.3	HOLD 信号，四线模式下为数据双向信号

注：SF_CSN0/CSN1 固定为 GPIO0/1

2.1.3.9 系统 GPIO 复用管脚

PIN	管脚名称	方向	电压(V)	描述
	PAD_GPIO[27:0]	I/O	3.3	通用双向管脚，可通过软件任意配置输入、输出、硬件模式、软件模式
	PAD_PMU_GPIO[4:0]	I/O	3.3	通用双向管脚，可通过软件任意配置输入、输出、硬件模式、软件模式。在 Always_On 模块内，系统上电、红外、报警等功能

3 系统

3.1 时钟结构

GK8602A 系统外部晶振为 24M，有 6 个 PLL。

- PLL_SYS 为主 PLL，用于产生系统总线及外设的工作时钟
- PLL_DDR 为 DDR 相关电路提供时钟。（此 PLL 在 DDR PHY 内部）
- PLL_ARM 为 ARM 相关电路提供时钟
- PLL_SENSOR 为 SENSOR 相关时钟域
- PLL_AUDIO 为音频相关电路提供相应时钟
- PLL_VIDEO 为视频相关电路提供相应时钟

3.1.1 时钟配置说明

GK8602A PLL 模块有两种工作模式：整数模式和小数模式

REFDIV[5:0]:输入参考时钟分频系数（1~63）
FBDIV[11:0]：VCO 分频系数（整数模式：16~2400　　小数模式：20~240）
POSTDIV1[2:0]：输出时钟分频系数 1（1~7）
POSTDIV2[2:0]：输出时钟分频系数 2（1~7）
FRAC[23:0]：VCO 小数部分分频系数

3.1.2 输出时钟频率计算公式

整数模式：Fout=(Fin/ REFDIV)*FBDIV/ POSTDIV1/ POSTDIV2
小数模式：Fout=(Fin/ REFDIV)*(FBDIV+ FRAC/2^{24})/ POSTDIV1/ POSTDIV2

3.2 CPU

GK8602A 内部采用 ARM1176ZJFS 处理器，主频最高 600MHz，16K ICache 以及 16K DCache。

3.2.1 CPU 特点

1. 支持 TrustZone 保密性扩展。
2. 支持智能功耗管理。
3. 支持高速 AMBA 总线。
4. 支持 8 级流水结构。
5. 支持分支预测功能。
6. 支持两级中断延迟配置。
7. 内部使用 CP14 与 CP15 协处理器。
8. 支持 VFP 协处理器。
9. 支持外部协处理器接口。
10. 支持指令与数据 MMU。
11. 支持指令与数据 Cache，包括一个具有 Hit-Under-Miss 功能的非阻塞的数据 Cache。
12. 支持虚拟序列与物理地址 Caches。
13. 64 位 Cache 接口位宽。
14. 支持一级缓存 TCM。
15. 支持 Trace 功能。
16. 支持基于 JTAG 的调试。

3.2.2 功能描述

ARM11 内部为 ARM1176 核，ARM1176 使用的 I/RW 接口支持 64bit AXI 总线协议，使用的 P/D 接口支持 32bit AXI 总线协议。在 ARM11 模块内部，使用了多个 64/32bit 的 AXI2AHB 桥，使其输出接口为支持 64bit AHB 总线协议的 I/RW 接口以及支持 32bit AHB 总线协议的 P 接口（D 接口不使用）。其中，I/RW 接口用于与指令、数据缓存连接；P 接口用于与系统 AHB 总线上的一个 MASTER 模块相连。

系统内部的高速模块采用 32bit AHB 总线互联，低速模块采用 32bit APB 总线连接，APB 总线通过 APB 桥与 AHB 总线连接。

由于系统使用 P 接口与系统 AHB 总线相连，因此在 ARM11 软件中需要将系统外设地址空间映射至 P 接口。

CPU 与总线接口时序请参见 AXI/AHB/APB 标准协议。

3.2.3 工作方式

请参见 ARM1176-JZFS 使用手册。

3.3 Boot 机制

3.3.1 概述

当系统启动后，CPU 从 Boot Rom 中读出指令。按照 BootCode 的指引，CPU 将 Sflash 里的 Image 拷贝至 DDR 中，然后运行。

根据 sflash 中存放的 Image 是明文还是密文，可将 Boot 机制分为普通启动和加密启动。即若为普通启动，CPU 直接将明文 Image 拷贝至 DDR 运行；若为加密启动，CPU 先将密文 Image 通过解密引擎解密成明文，在拷贝至 DDR，然后运行，Boot 完成。

3.3.2 特点

- Boot ROM 大小为 2.5KB （从 0x0 到 0xA00）
- 支持 SPI NOR Flash 或者 SPI NAND Flash 启动
- 支持从 EFUSE 读取不同模式来做不同模式的启动
- 支持 AES/DES 全数据加密启动，且解密过程在芯片内部进行，解密密钥从 EFUSE 中读取，软件无法得到 AES key

3.3.3 普通启动

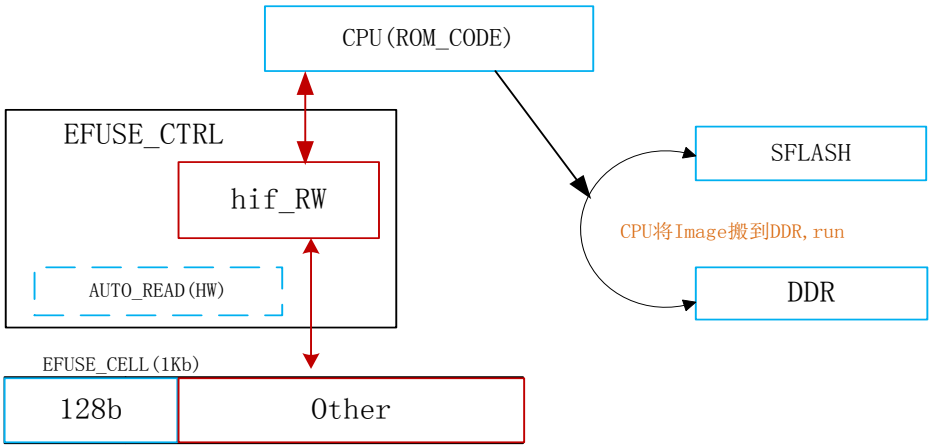


Figure 3.3.1 普通启动系统硬件框图

如上图所示，Boot_code 读取 Efuse 中的是否加密启动标志位。若标志为普通启动，则 CPU 直接将 Sflash 中的内容拷贝到 DDR，运行 Image，普通启动完成。

3.3.4 加密启动

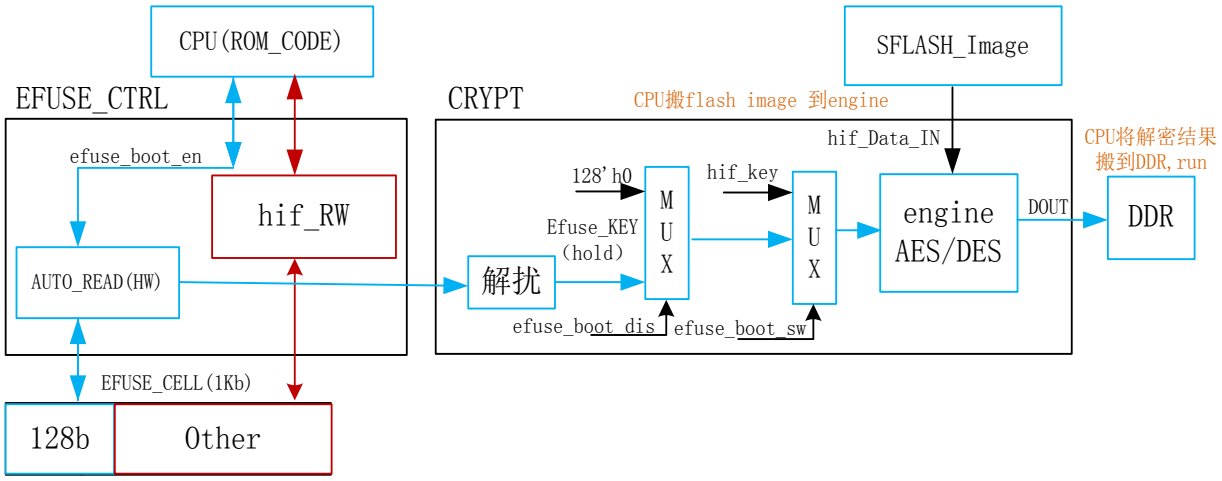


Figure 3.3.2 加密启动系统硬件框图

如上图所示，芯片加密启动过程中涉及两个硬件模块:EFUSE_CTRL 和 Crypt。EFUSE_CTRL 负责控制读取 EFUSE 区域中的信息。crypt 模块负责将 flash 中的加密数据解密。EFUSE 区域中前 128bits 为 flash image 解密密钥 key1，软件无法通过 CPU 读出此 128bits。解密时密钥 key1 由硬件输出至 Crypt 模块,并经过解扰生成 key0 作为解密密钥。crypt 模块支持 AES/DES 两种加解密方式，可以通过配置 EFUSE 中的 engine_sel_bit 配置决定使用哪种引擎加解密。

3.4 Efuse 控制器

3.4.1 概述

Efuse 控制模块接收 AHB 接口的读写控制信号，根据 efuse 芯片写入\读出信号的时序要求产生读写 efsuse 的控制信号。

3.4.2 特点

- Efuse 接口时序寄存器可调
- 开辟有 CPU 不可见的安全区
- 标准 AHB 接口，方便挂载

3.4.3 功能描述

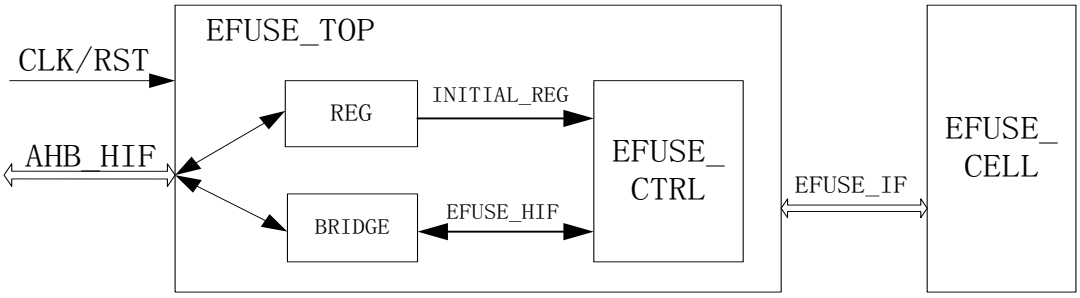


Figure 3.4.1 EFUSE_TOP 硬件框图

错误!未找到引用源。 描述了 EFUSE 控制器的硬件框图，Efuse 控制模块接收 HIF 接口的读写控制信号，根据 efuse 芯片写入\读出信号的时序要求产生读写 efsuse 的控制信号。下面各图为 HIF 端及 efuse 端接口时序图：

错误!未找到引用源。 描述了 HIF 接口读数据的时序图。

错误!未找到引用源。 描述了 HIF 接口写数据的时序图。

图 **错误!未找到引用源。** 描述了 efuse 数据写入的时序关系图，

图 3.6.5 描述了 efuse 数据读出的时序关系图，

表 3.6.1 说明了 efuse 数据写入时的信号的建立，保持时间的要求。

表 3.6.2 说明了 efuse 数据读出的信号的建立，保持时间的要求。

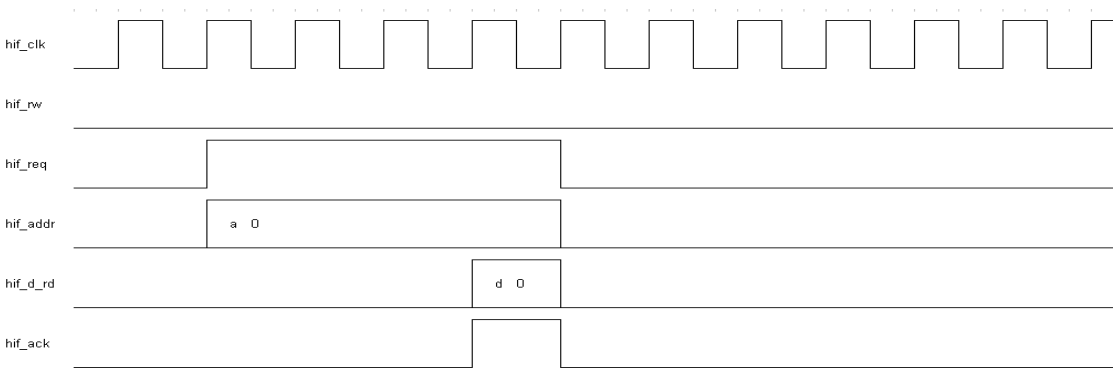


Figure 3.4.2 HIF 接口读数据时序图

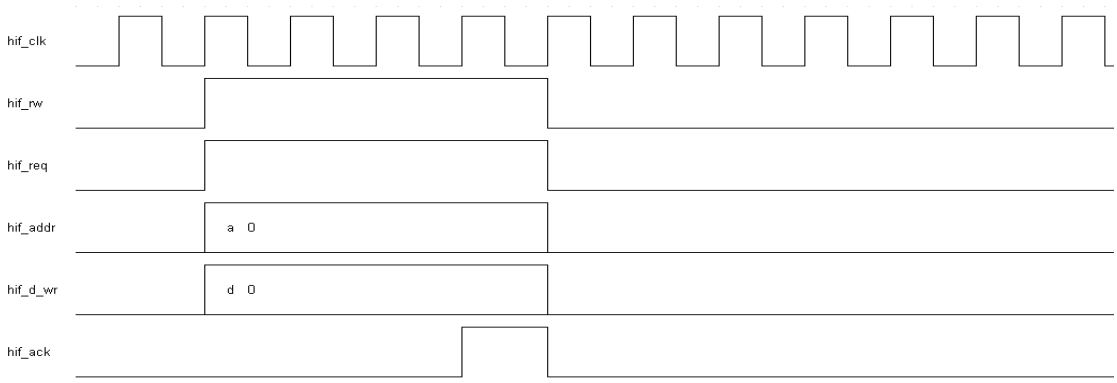


Figure 3.4.3 HIF 接口写数据时序图

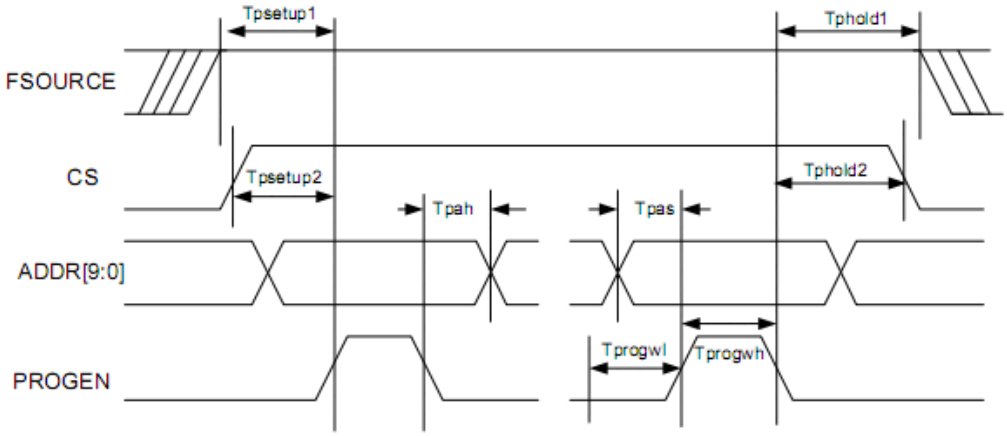


Figure 3.4.4 efuse 数据写入时序图

Parameter	Description	Min	Typ	Max	Units
Tpsetup1 (Note 1)	FSOURCE to PROGEN setup time	200	-	-	ns
Tphold1	FSOURCE to PROGEN hold time	200	-	-	ns
Tpsetup2	CS to PROGEN setup time	20	-	-	ns
Tphold2	CS to PROGEN hold time	20	-	-	ns
Tprogwh	PROGEN pulse width high	1.8	2	2.2	us
Tprogwl	PROGEN pulse width low	200	-	-	ns
Tpas	ADDR[9:0] to PROGEN setup time	20	-	-	ns
Tpah	ADDR[9:0] to PROGEN hold time	20	-	-	ns

Table 3.4.1 efuse 数据写入时序要求表

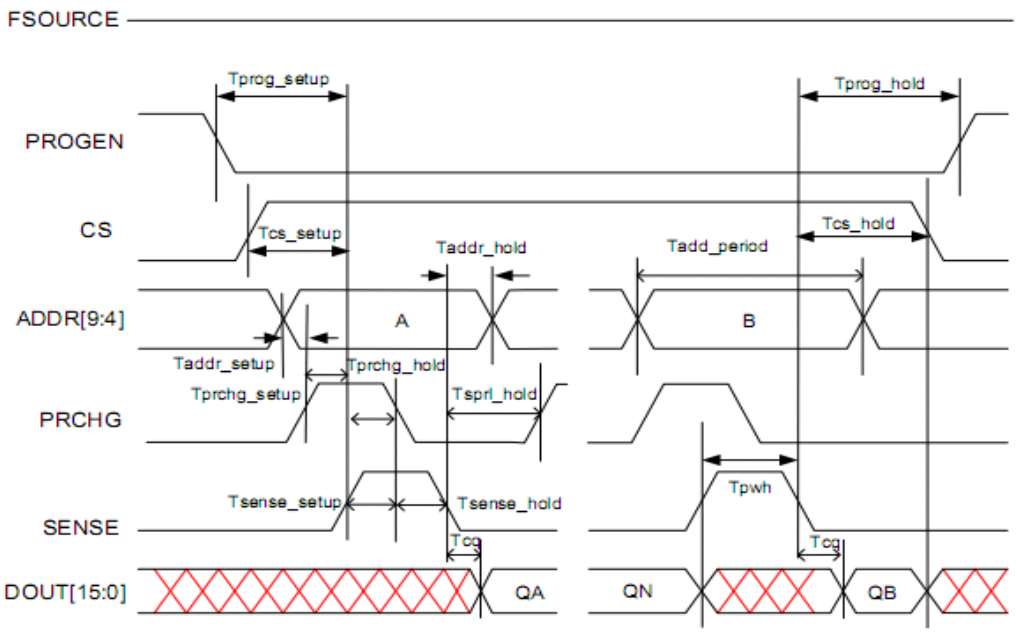


Figure 3.4.5 efuse 数据读出时序图

Parameter	Description	Min	Typ	Max	Units
Tcs_setup	CS to SENSE setup time	18	-	-	ns
Tcs_hold	CS to SENSE hold time	12	-	-	ns
Tprog_setup	PROGEN to SENSE setup time	18	-	-	ns
Tprog_hold	PROGEN to SENSE hold time	12	-	-	ns
Tprchg_setup	PRCHG to rising edge of SENSE setup time	6	-	-	ns
Tprchg_hold	PRCHG to rising edge of SENSE hold time	6	-	-	ns
Tsense_setup	SENSE PULSE to falling edge of PRCHG setup time	6	-	-	ns
Tsense_hold	SENSE PULSE to falling edge of PRCHG hold time	6	-	-	ns
Tsprl_hold	PRCHG to falling edge of SENSE hold time	6	-	-	ns
Taddr_setup	ADDRESS to the rising edge of PRCHG	12	-	-	ns
Taddr_hold	ADDRESS to the falling edge of SENSE	12	-	-	ns
Taddr_period	ADDRESS valid period in one read cycle	42	-	-	ns
Tpwh	Pulse width of SENSE signal	12	-	-	ns
Tcq	Delay time from the falling edge of SENSE to output data valid. [with output load: 0.6pf]	-	-	8	ns

Table 3.4.2 efuse 数据读出时序要求表

3.4.3.1 接口时序

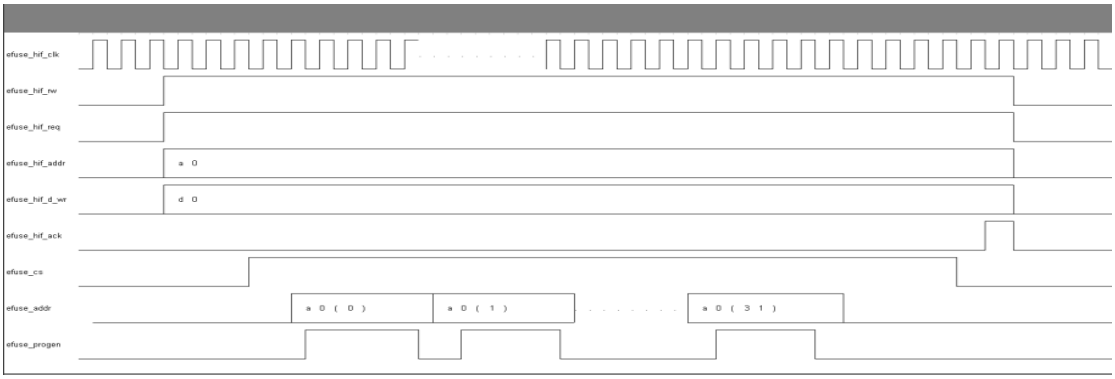


Figure 3.4.6 HIF 写 efuse 时序图

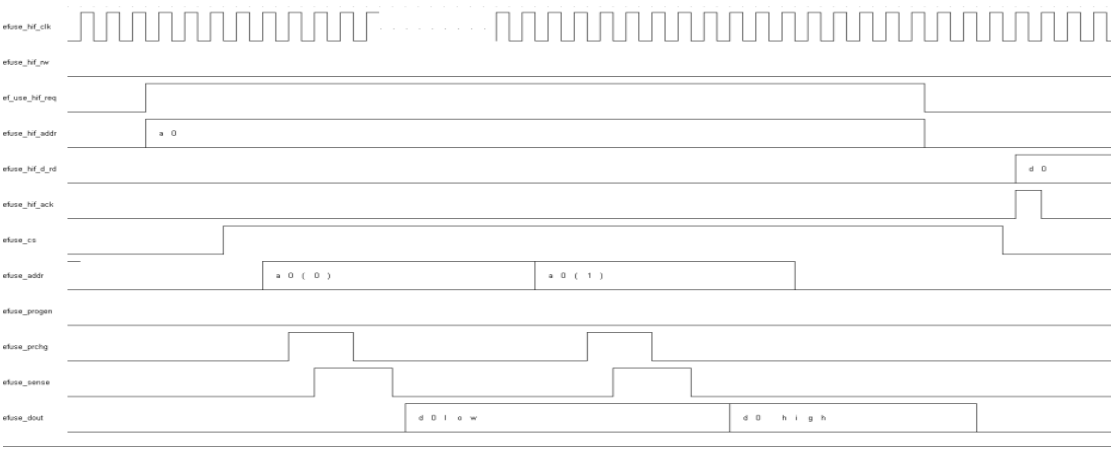


Figure 3.4.7 HIF 读 efuse 时序图

3.4.4 工作方式

3.4.4.1 初始化

在 Efuse_ctrl 工作之前，为了和 Efuse 器件要求的接口时序相对应，需要对其进行初始化，即配置寄存器 EFUSE_CTRL_REG,当 Efuse_ctrl 时钟频率为 138MHz 时，该寄存器值应为 0x4551C119。

3.4.4.2 硬件自动读取

为安全启动流程，我们将 Efuse 前 128 位开辟为秘钥区，只有硬件逻辑可以读取，CPU 不可读取。Efuse_ctrl 初始化后，置位寄存器 EFUSE_BOOT_EN,硬件逻辑开始读出前 128 位 Efuse 数据,传到加解密引擎中，并置 RD_BOOT_OK 为 1，表示 128 位读完成。

3.5 加密引擎

3.5.1 概述

此加密引擎挂载在 AHB 总线上，支持 AES 和 DES 加密解密通用引擎。AES 引擎支持 128/192/256bit 三种 KEY 选择，128 数据输入，128bit 结果输出。DES 引擎为标准的 64bit KEY，64bit 数据输入，64bit 数据输出。

3.5.2 特点

- 支持 AES128/AES192/AES256
- AES/DES 拥有独立的中断源
- AHB 总线接口，灵活配置

3.5.3 功能描述

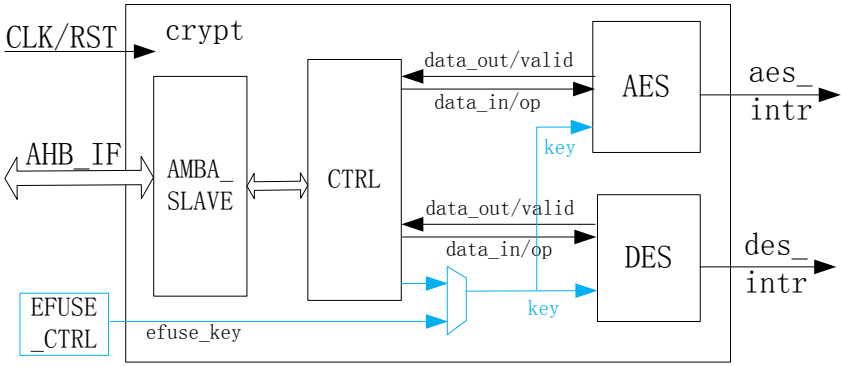


Figure 3.5.1 加密引擎模块框图

如上图所示，配置寄存器，写入数据及读出处理后的数据，都由 CPU 通过 AHB 总线完成。引擎的 KEY 可以选择由 AHB 写入或者 Efuse_ctrl 模块提供。引擎工作时，按照流程依次输入 KEY，FLAG，DATA 然后等待引擎 ready 信号置 1，就可以读取处理后的结果。如果中断使能为打开，当处理完成时会产生中断。

3.5.4 工作方式

1. 通过 AHB 总线写入 key（AES 128bit/192bit/256bit 或者 DES 64bit）
2. 写 FLAG（ 0：加密/ 1：解密）
3. 写需要加密或解密的数据（AES 128bit 或者 DES 64bit）
4. 等待 Output_Ready 置 1 或收到中断后，就可以从输出数据寄存器中读取处理后的数据
5. 当读取完成后，就能输入下一个数据。如果要更新 KEY 或 FLAG,要在更新数据之前更新。

3.6 GPIO

3.6.1 概述

GPIO 模块为通用接口模块，用户可以根据实际系统使用需求对芯片管脚与芯片内部功能所使用的功能接口进行配置，提高了芯片应用的灵活性。

3.6.2 特点

1. GPIO 模块采用了全路由方式设计，使 GPIO 管脚与内部模块功能接口可以任意配置，具有完全的配置灵活性。
2. 支持最多 64 个 GPIO 配置。
3. 支持外部中断源配置。

3.6.3 功能描述

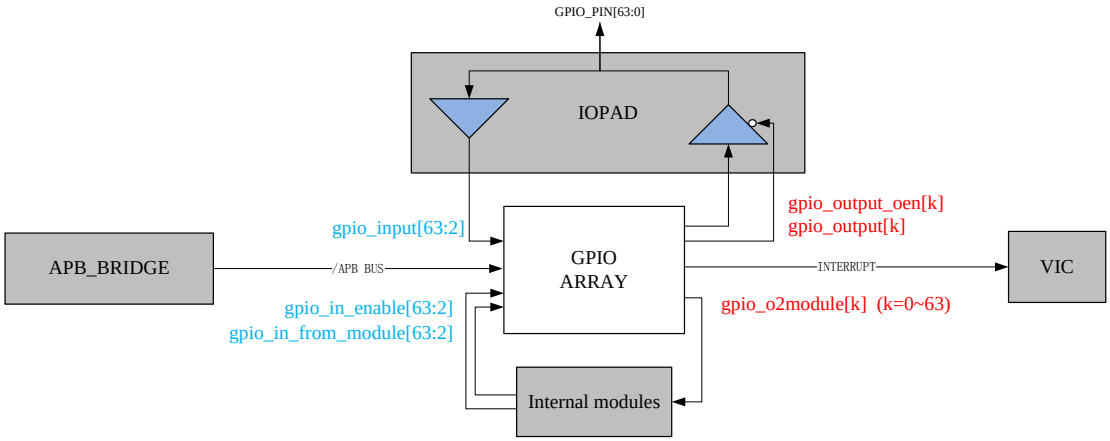


Figure 3.6.1 GPIO 结构图

GPIO 结构设计如上图所示。GPIO 模块通过 APB 总线进行配置，最多可以支持 37 个 GPIO 管脚(当前系统实际使用了 28 个 GPIO 管脚)。同时 GPIO 支持由芯片外部输入中断，通过 VIC 向系统 CPU 触发中断。

3.6.4 工作方式

3.6.4.1 GPIO 输入与中断

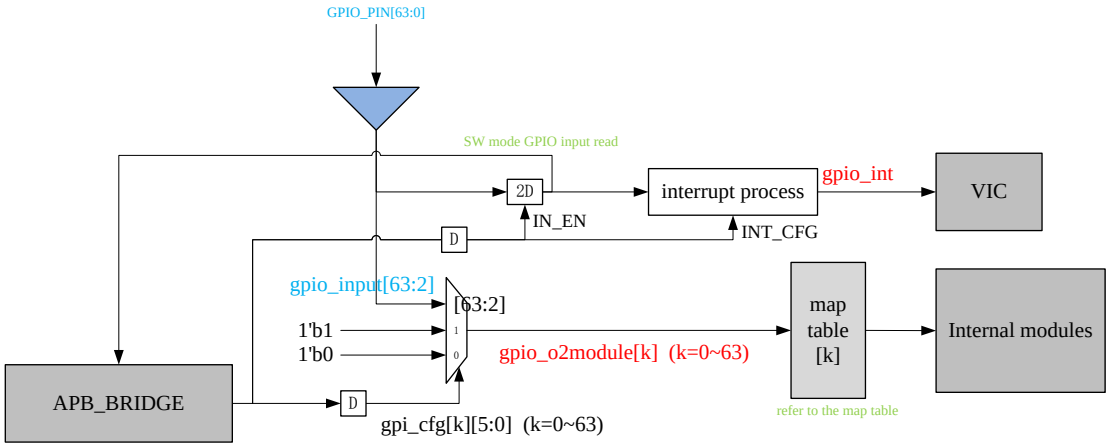


Figure 3.6.2 GPIO 输入与中断

GPIO 输入与中断设计如上图所示。gpio_o2module[k](k=0~63)与内部模块的连接关系，以及各个信号对应的配置地址映射表如下(map table)。配置 gpio_o2module[k]对应的gpi_cfg[k]寄存器，完成其与 GPIO_PIN[63：2]的 MUX 选择（GPIO_PIN[1:0]不用于输入模式，被 2’b10 占用）。

3.6.4.2 GPIO 输出

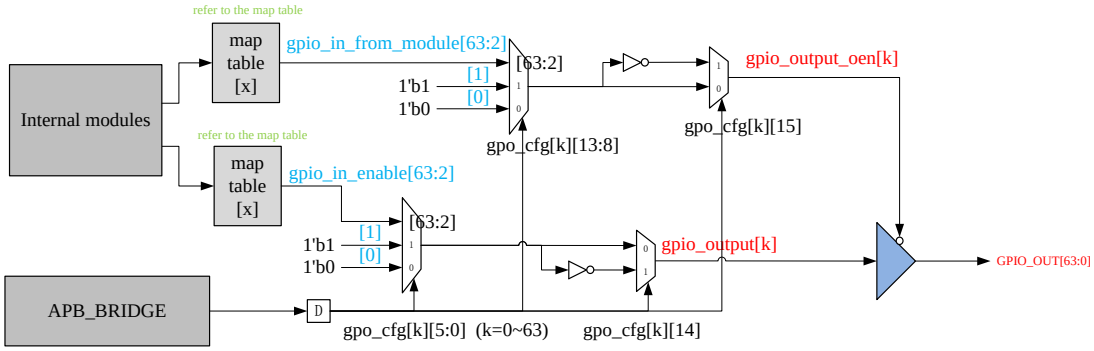


Figure 3.6.3 GPIO 输出设计

GPIO 输出设计如上图所示。内部模块信号与 gpio_in_from_module[x] 以及 gpio_in_enable[x] (x=0~63)的映射关系如下（map table）。

3.6.4.3 RTC 功能

RTC 模块为真实时间控制器，输入时钟为 24MHz/32.768KHz，它通过配置内部寄存器 PRE_SCALER 来控制计数器，使其输出 1Hz 计时时钟，同时输出对应的系统中断。系统 CPU 通过计时相关的寄存器获得当前时间信息，支持秒，分，时，天的计时信息输出。可以通过配置内部寄存器选择采用 24MHz 时钟或 32.768KHz 时钟进行计数。

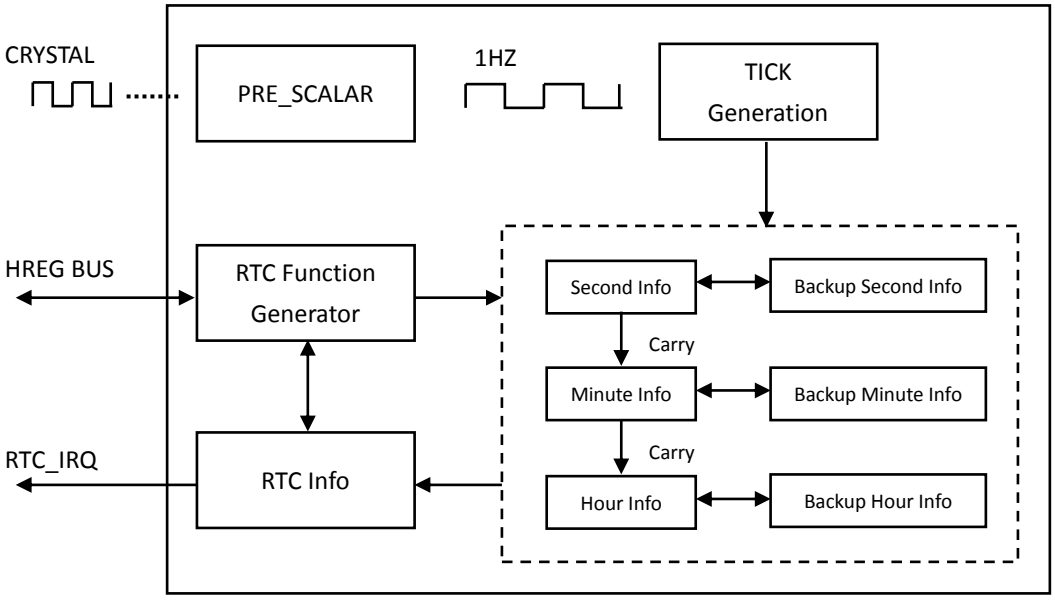


Figure 3.6.4 RTC 系统结构

RTC 系统结构如上图所示。

RTCREG_DIV0 – RTCREG_DIV3 组成 pre-scalar 寄存器，该寄存器可以通过 HREG 总线访问：

pre-scalar[31:0] = {RTCREG_DIV3, RTCREG_DIV2 ,RTCREG_DIV1, RTCREG_DIV0}

例如，使用 24MHz 输入，则 pre-scalar 应配置为 (0x16e3600-1)。

由于 RTC 支持 24MHz/32.768KHz 两个时钟输入，RTCREG_CFG[0](module_enable 寄存器)或 RTC 计数器软复位应先配置为“0”，当 RTCREG_DIV0 – RTCREG_DIV3 以及其它寄存器配置完成，如果使用 32KHz 时钟则需等待至少 5 个 32.768KHz 时钟周期的时间后（约为 30.52us），才可以配置模块使能或软复位为“1”。

注意：如果 module_enable 寄存器配置为 0，则 RTC 内所有逻辑单元会复位为初始值。

3.6.4.4 GPIO 功能

GPIO 模块有两个模式，软件模式与硬件模式。GPIO 初始化为硬件模式。GPIO 模块结构如下图所示。

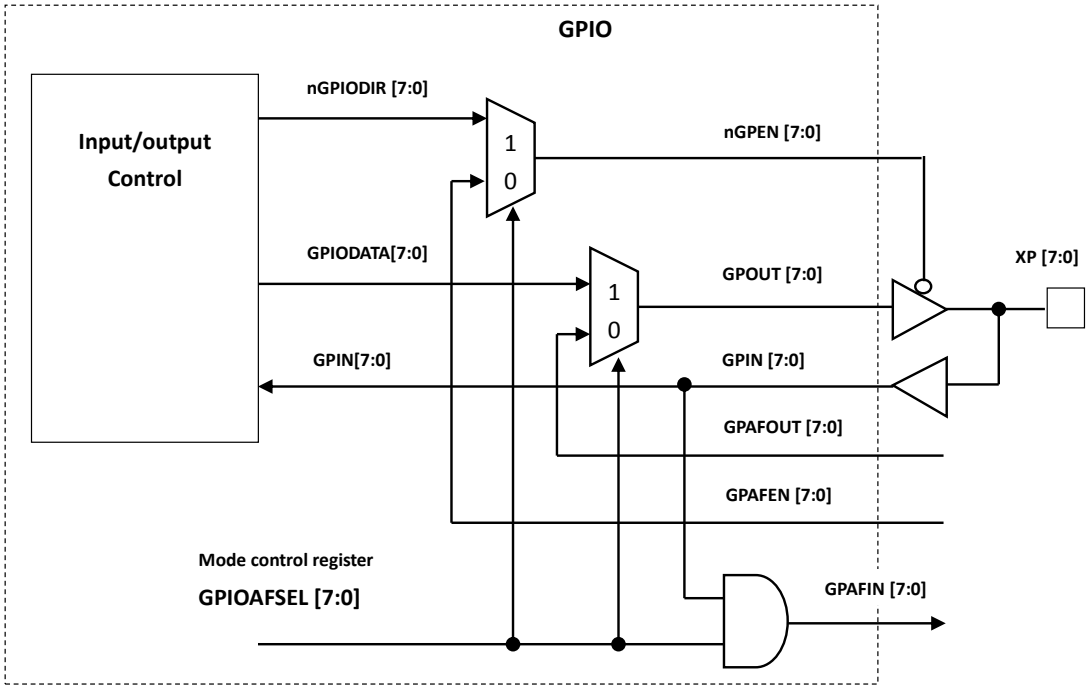


Figure 3.6.5 GPIO 模块结构图

3.6.4.5 中断功能

所有 PMU 外设模块的中断都被送至中断处理模块，经 IRQ mask 与 IRQ clear 处理后输出两组中断，一组为 XINT 用于 CPU 与 MUX；一组为 PHERI_INT[7:0]用于 MCU 处理。

3.6.5 工作方式

3.6.5.1 上电与工作流程

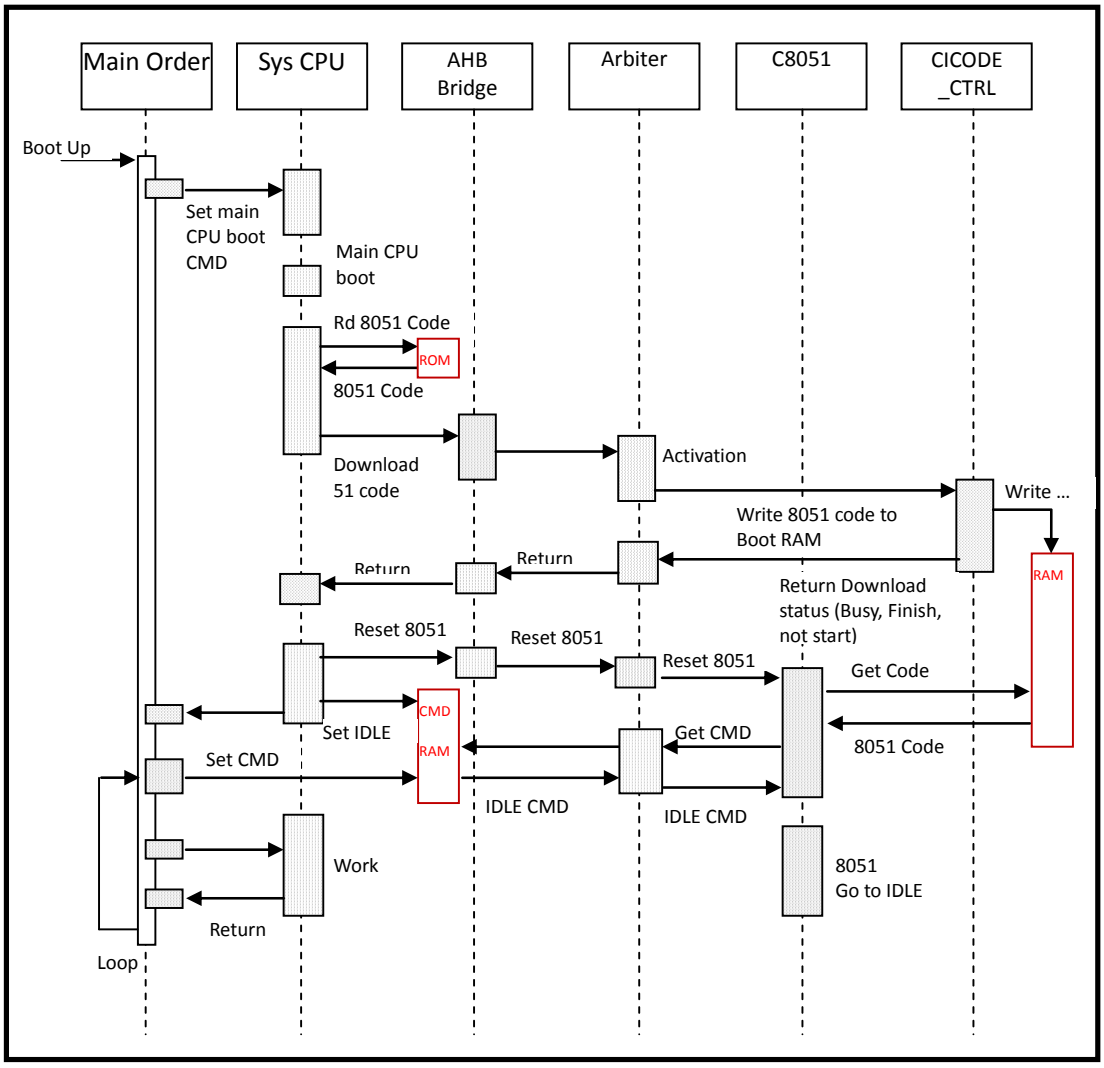


Figure 3.6.6 PMU 系统上电与工作流程

PMU 系统上电与工作流程如上图所示。

在上电过程中，系统 CPU 通过 ROM 启动，此时 PMU 内的 C8051 不工作。

系统 CPU 通过 AHB 总线将 C8051 的启动代码下载到 PMU 内的“Boot RAM”中，随后释放 C8051 的复位信号。

在系统 CPU 工作的过程中，C8051 不需要工作，处于空闲状态。

3.6.5.2 下电流程

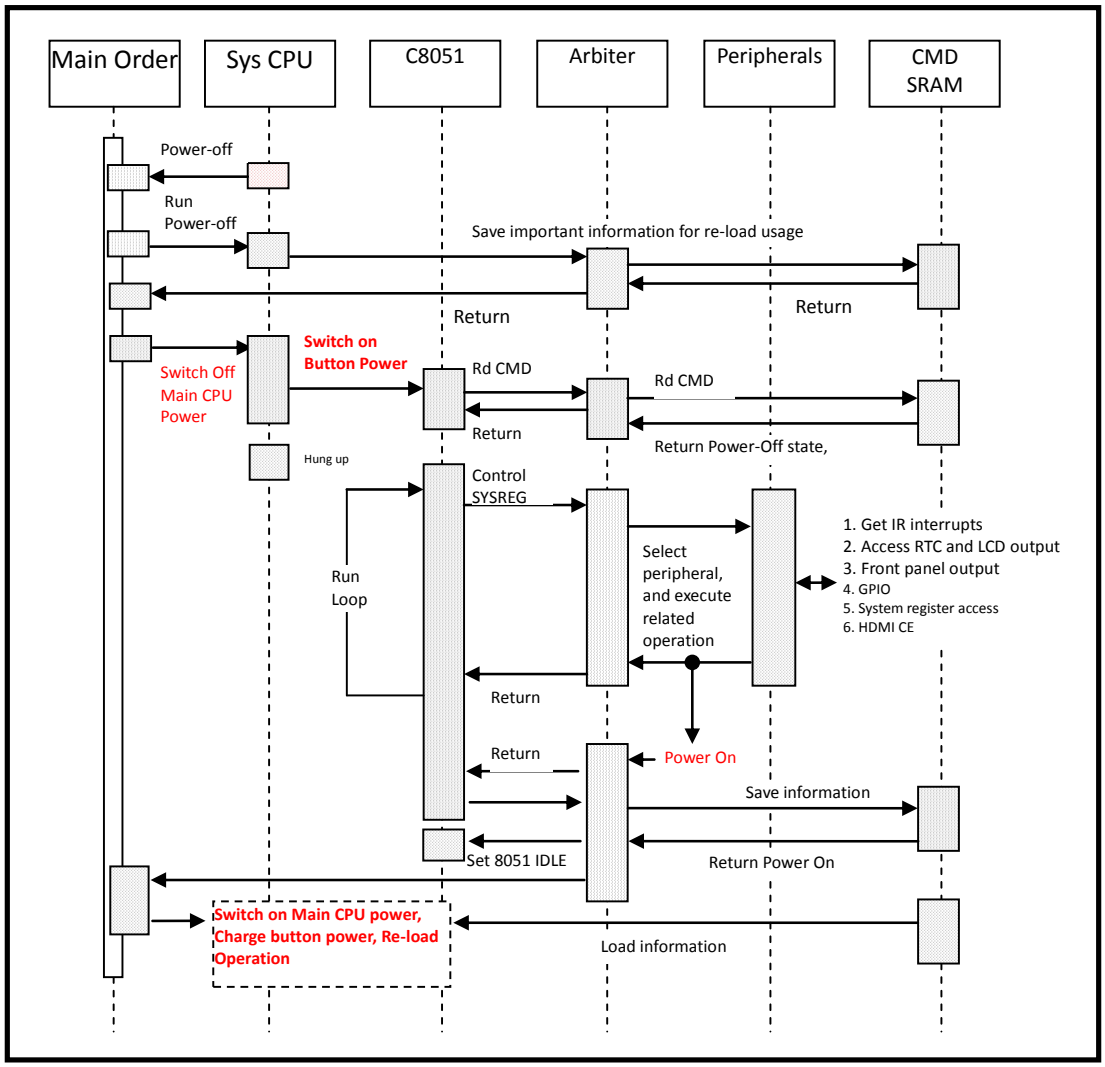


Figure 3.6.7 PMU 系统下端流程图

PMU 系统下端流程图如上图所示。

下电过程由主 CPU 发起，主系统下电时，C8051 控制 SYSREG 并关闭主系统供电，此时系统进入待机状态，功耗最低。在系统下电前，主 CPU 可以在 PMU 的 SRAM 中保存必要的信息，当系统从待机状态唤醒后，可以重新加载这些缓存的信息。

3.7 DMA

3.7.1 概述

系统支持 DMA，用于数字音频接口/模拟音频接口的数据搬移。支持内存至总线设备或总线设备至内存的两种 DMA 操作。

3.7.2 特点

1. 两个物理 DMA 通道，数字音频接口与模拟音频接口复用相同的 DMA 通道。
2. DMA 源地址可配置。
3. DMA 目的地址可配置。
4. DMA 可配置最大数据传输量为 4M 字节。
5. 可配置总线接口传输大小：8~1024 字节。
6. 可配置总线数据位宽：1,2,3,4 或 8 字节。
7. 软件直接可配置或通过内存描述符解析得到 DMA 控制信息。
8. 主处理器启动 DMA 操作。
9. 中断信号标明 DMA 操作结束。
10. 通过 Round Robin 仲裁方式共享 AMBA 总线主机接口。
11. 全流水结构，高数据流量。
12. 支持 AMBA 总线从接口读写寄存器。

3.7.3 功能描述

通过软件配置，DMA 将源地址数据搬移至目的地址。DMA 引擎由源地址的最低地址开始读取数据，写入目的地址的最低地址，其后的操作对地址进行累加，直至完成配置的数据量传输。

DMA 通道 1 用于音频外设接收数据至内存的数据搬移，DMA 通道 2 用于将内存数据搬移至音频外设缓存。

3.7.4 工作方式

数字音频接口与模拟音频接口两个外设复用相同的 DMA 通道，通过配置 AHB 通用寄存器中的 Audio_i2s_sel 寄存器完成外设选择。

一次完整的 DMA 包括如下步骤：

1. DMA 通道配置完成。
2. 使能 DMA 传输。这将启动数据的搬移过程。
3. 一系列的数据读写操作，完成由源地址到目的地址的数据搬移。
4. 当配置的数据量搬移完成时，一次数据搬移结束。
5. DMA 通道数据传输功能自动变为非使能。
6. DMA 传输状态被记录至通道状态寄存器。

DMA 通道必须在任何 DMA 操作前完成配置。每个 DMA 通道有三个寄存器用于保存 DMA 操作的配置信息：

1. 通道控制寄存器。
2. 通道源地址。
3. 通道目的地址。

一次 DMA 操作最多可以搬移 4M 字节数据。AMBA 总线传输大小可配置为 8~1024 字节，AMBA 总线位宽可配置为 1,2,4,或 8 字节。不支持内存到内存或设备到设备的数据搬移。

3.7.4.1 工作模式

在启动任何 DMA 操作前，系统软件先选择外设以及对应的 DMA 通道，并对相应的通道寄存器进行配置。

在软件配置期间，DAM 通道必须是非使能的。在工作模式下，数据传输使能控制标志是通道控制寄存器中的使能比特。当该比特写“1”后，DMA 传输启动。一次典型的数据传输，首先控制器会配置源与目的地址，然后配置通道控制寄存器其余的信息，最后向使能比特写“1”启动传输。当数据传输完成或传输出错时，DMA 通道会产生一个中断，该中断会告知主控器，DMA 传输已完成。

3.8 定时器

3.8.1 概述

定时器实现定时功能。原理是通过记录标准时钟的脉冲个数来达到计时功能，所有本质上定时器即为计数器。

3.8.2 特点

3 组独立 32 位可编程定时器，每组定时器拥有 2 个匹配数寄存器，且计数值可以随时读取。

三个计时器可以独立配置为使用内部时钟或外部时钟。使用内部时钟时，定时器的时钟与 APB 总线时钟相同。使用外部时钟时，时钟由相应的 GPIO 输入。此时定时器容易受到输入时钟的影响。

3.8.3 功能描述

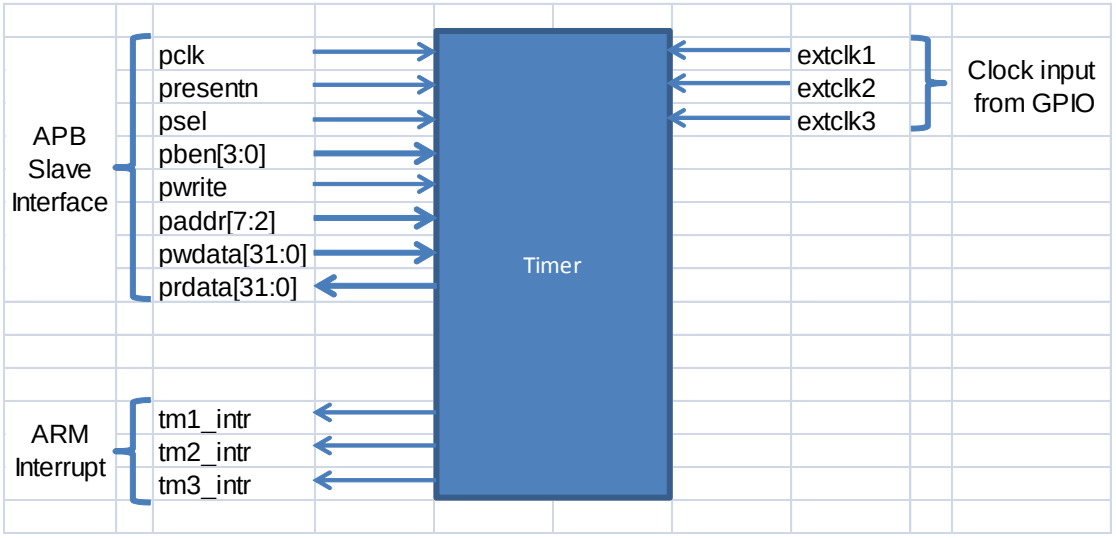


Figure 3.8.1 定时器功能框图

3.8.4 工作方式

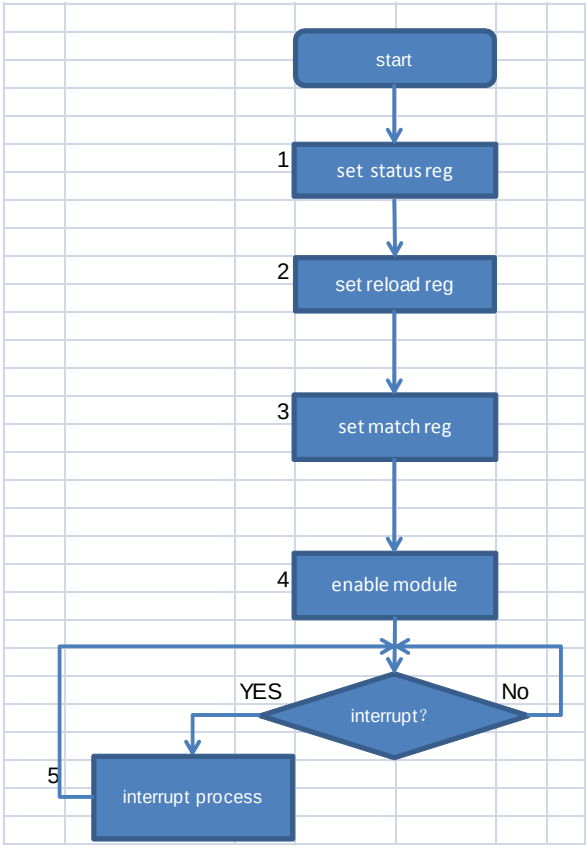


Figure 3.8.2 定时器配置流程图

配置流程：

1 设定计数值寄存器。当定时器开始工作后，此寄存器值将在每个时钟减 1。

1. 设定重载值寄存器。当计数值寄存器为 0 后，计数值寄存器将设置为此寄存器值。
2. 设定匹配寄存器。当计数值寄存器等于此寄存器时，产生中断信号。
3. 开启计数器
4. 中断处理

3.9 看门狗

3.9.1 概述

看门狗是一个特殊定时器（自减计数器），用于防止因为外界干扰或者程序错误等原因，造成系统锁死。原理为系统运行后也就启动了看门狗的计数器，看门狗就开始自减计数，如果到了一定的时间还未清除计数器，那么看门狗计数器就会溢出从而引起中断，产生系统复位信号。

3.9.2 特点

看门狗计数时钟与 APB 总线时钟相同，看门狗复位时将产生一个系统复位信号和一个中断信号。

3.9.3 功能描述

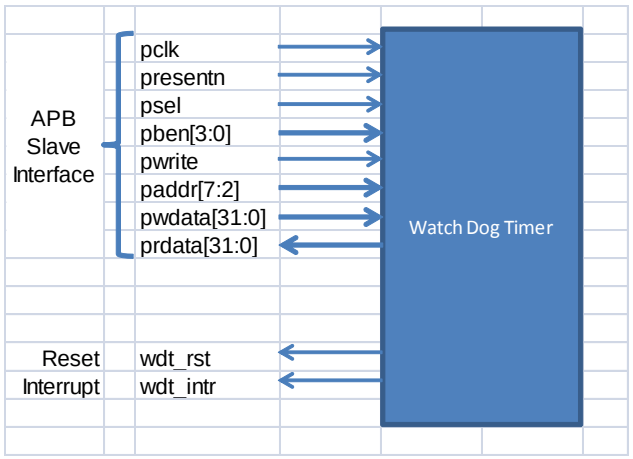


Figure 3.9.1 看门狗功能框图

3.9.4 工作方式

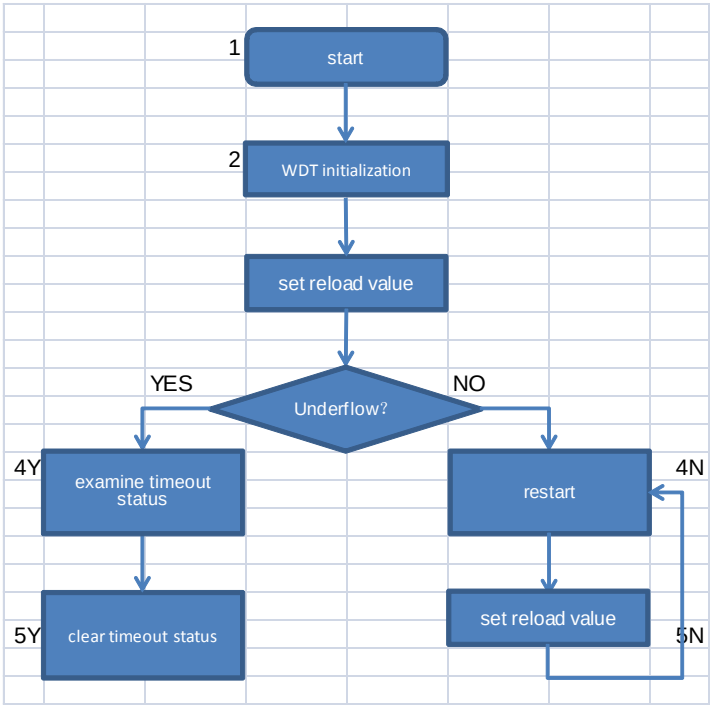


Figure 3.9.2 看门狗配置流程图

配置流程：

1. 看门狗初始化。配置控制寄存器，启用复位信号，配置复位信号脉宽，开启看门狗。
2. 设定重载入值寄存器。当看门狗计数器下溢出是，计数器将载入此寄存器值。
3. 溢出判定
4. Y 计数器已经下溢出。检验超时状态寄存器，判定系统复位由看门狗引起。
   N 没有溢出。重设寄存器，清除看门狗。
5. Y 清除超时状态寄存器。
   N 设定重载入值寄存器，或者直接定时清除看门狗。

4 外围接口

4.1 DDRC

4.1.1 概述

DDRC 是 DDR Controller 的简称，主要功能是实现 DDR 的时序控制以及系统 BUS 与 DDR 之间数据的交换等功能，是系统数据与内存交互的主要通道；该 DDRC 实现了高性能 DDR 控制器，总线数据位宽为 64bit@200MHz，采用流水线式的数据流处理，最多可接收 8 个总线请求，支持多种不同的 DDR SDRAM，例如：DDR2/DDR3 等；

4.1.2 特点

1.支持 DRAM 类型为 DDR2,DDR3,最大 DRAM 容量 8Gbit;

2.可配置外部数据总线为 32bit/64bit,内部总线为 64bit;

3.DDR CLOCK 可达 400MHz, data rate 可达 800MHz;

4.支持从 1byte 到 1Kbyte 任意突发长度，但是一次突发请求不能跨越 2KB 边界;

5.支持 INCR 和 WRAP 传输模式;

4.1.3 功能描述

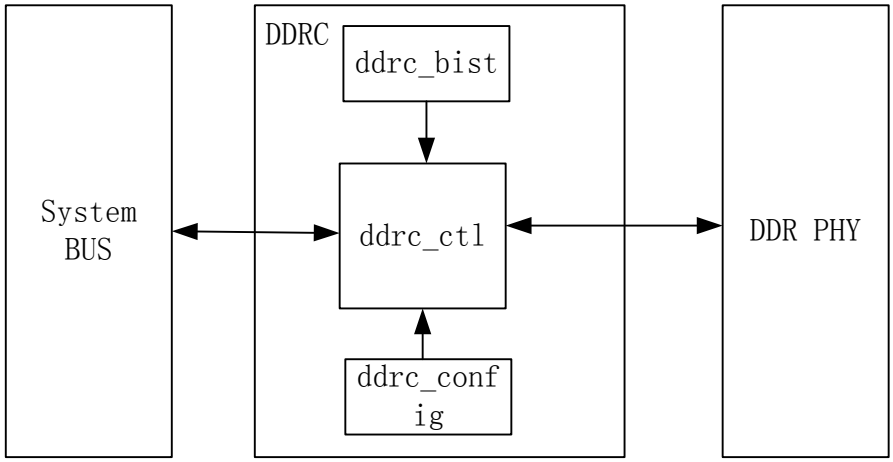


Figure 4.1.1 ddrc block diagram

ddrc_bist : 该模块是 ddrc build-in self test 的缩写，目的是在芯片 FT 阶段进行快速测试 ddrc 和 ddr phy 的功能，该模块通过芯片顶层端口控制内部所要测试的地址以及启动该模块工作，测试完成后，输出 test result 送给芯片顶层 pin 以便测量测试结果；

ddrc_ctl: 该模块主要功能为接收处理总线请求，数据预处理，system bus 时序控制等，是 ddr controller 主要功能模块；

ddrc_config: 该模块是 ddrc 寄存器配置模块，用于配置各种时序参数以及 ddrc_ctl 所需的功能寄存器配置；

ddrc 的数据接口是简单的握手协议，下图为 ddrc 接口时序图；

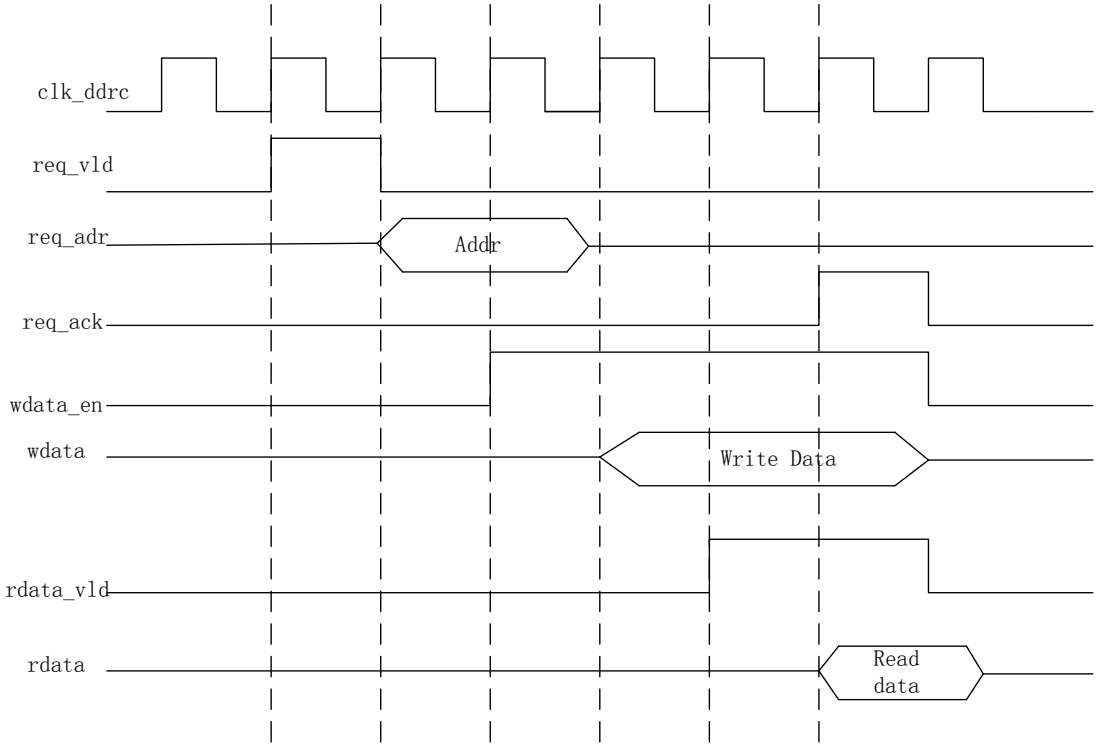


Figure 4.1.2 ddrc 接口时序图

4.1.4 工作方式

该 DDRC 工作前必须遵循规定的寄存器配置流程,如下：

1. 配置 DDR PHY 的 PLL 以及工作寄存器；
2. 配置 ddrc_reset 为高电平，以复位 ddrc 所有寄存器；
3. 配置 ddrc_cfg，启动 DRAM 标准初始化流程；
4. 使能 ddrc_ctl；
5. 正常访问数据；

4.2 USB

4.2.1 概述

USB 接口主要用于批量数据传输和保存。包括 USB Controller 和 USB PHY，符合 USB2.0 协议，支持高速（480Mbps）和全速（12 Mbps）传输，支持 OTG 功能，可以工作于 host 和 device 模式。USB 内置容量可配置的 FIFO，可以通过 DMA 进行数据传输。USB Controller 通过 32bit AHB 总线连接到 ARM，USB PHY 通过配置寄存器总线连接到 ARM。

4.2.2 特点

1. 支持 USB2.0 标准的高速和全速传输功能，兼容 USB1.1
2. 支持 USB OTG 功能
3. 支持 USB host 模式和 device 模式
4. 除了 EndPoint 0，最多可以支持 7 个 IN/OUT EndPoint
5. USB 内置 FIFO，动态可配，FIFO 支持 DMA 传输
6. USB PHY 支持 UTMI+level2 的功能

4.2.3 功能描述

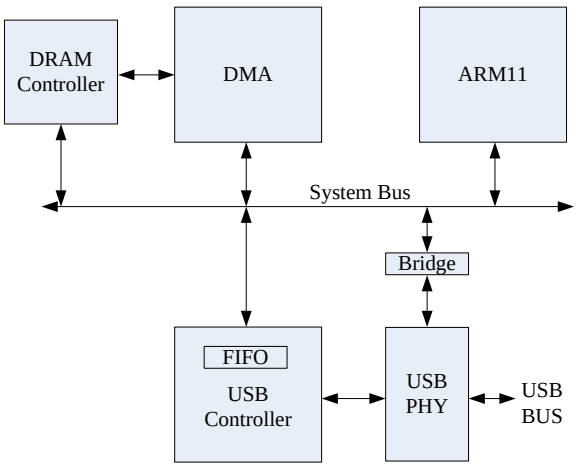


Figure 5.2.1 USB 系统功能框图

如上图所示，ARM11 通过 AHB 总线访问 USB Controller 和 USB PHY。DMA 通过 AHB 总线在 DDR 和 USB FIFO 之间进行数据传输。

USB Controller：USB Controller 支持 USB OTG 功能，主要有 host 模式和 device 模式，

USB PHY：USB PHY 主要是负责将 USB controller 发送过来的并行输出数据转换成高速的串行数据发送到 USB 总线上。同时还将 USB 总线上接收的高速串行数据转换成低速的并行数据传送给 USB controller。USB PHY 符合 USB 2.0 标准，支持 16bit UTMI 接口。

DMA Controller：DMA Controller 是通用 DMA。主要在 DDR 和 USB FIFO 之间进行数据传输。

4.2.4 工作方式

Host 模式下 DMA 操作流程如下：

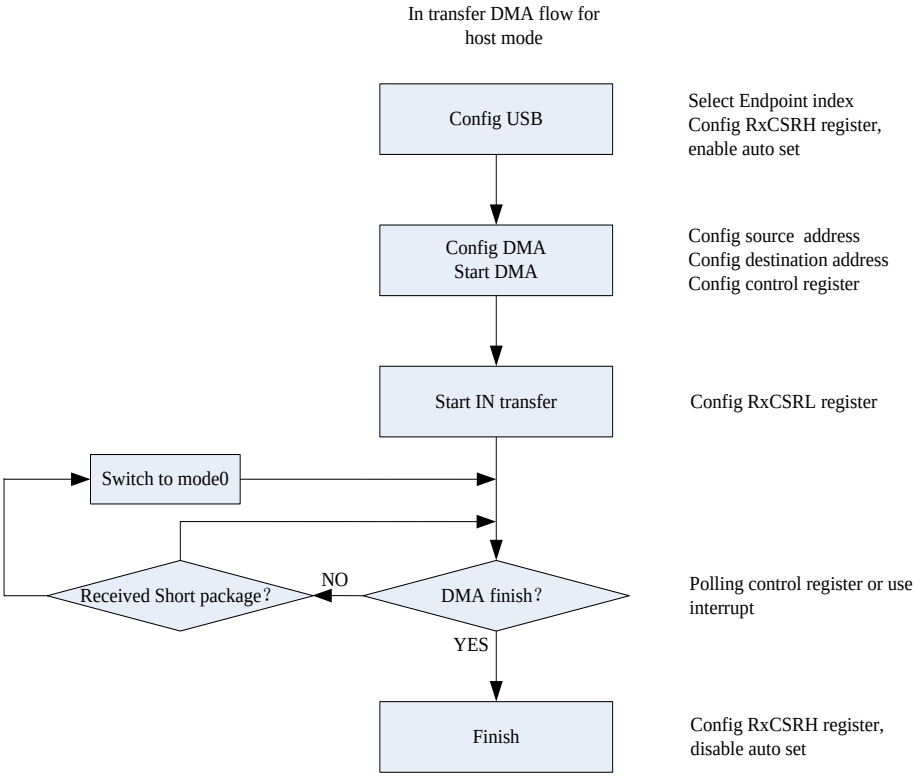


Figure 5.2.2 Host 模式 DMA In 传输

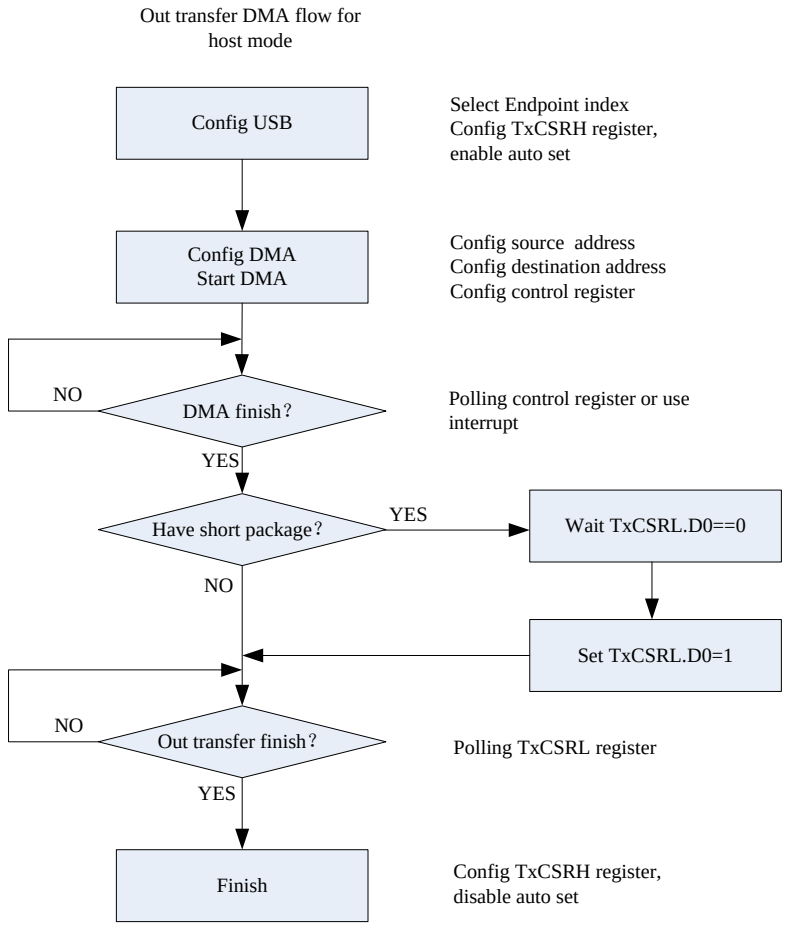


Figure 5.2.3 Host 模式 DMA Out 传输

Device 模式下 DMA 操作流程如下：

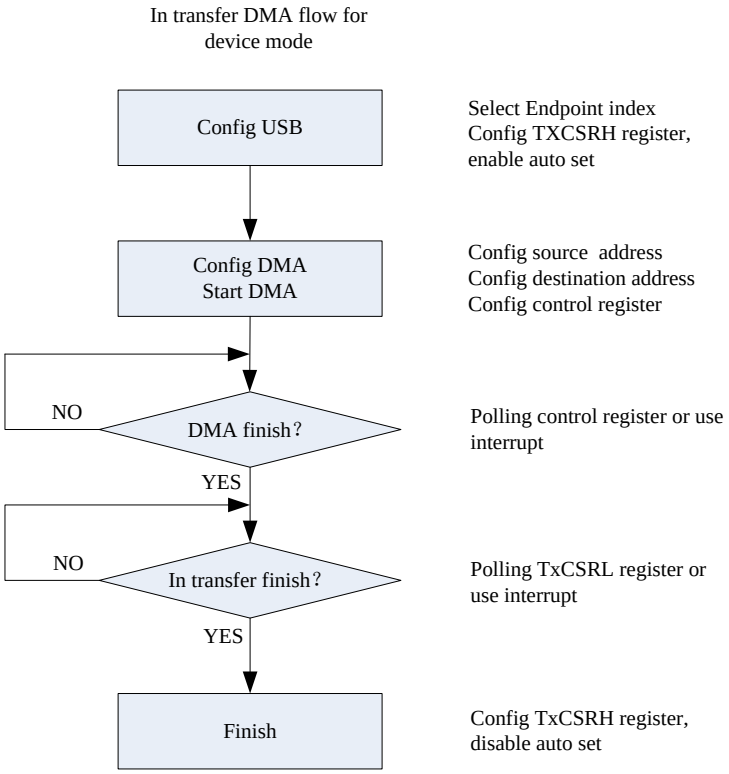


Figure 5.2.4 Device 模式 DMA In 传输

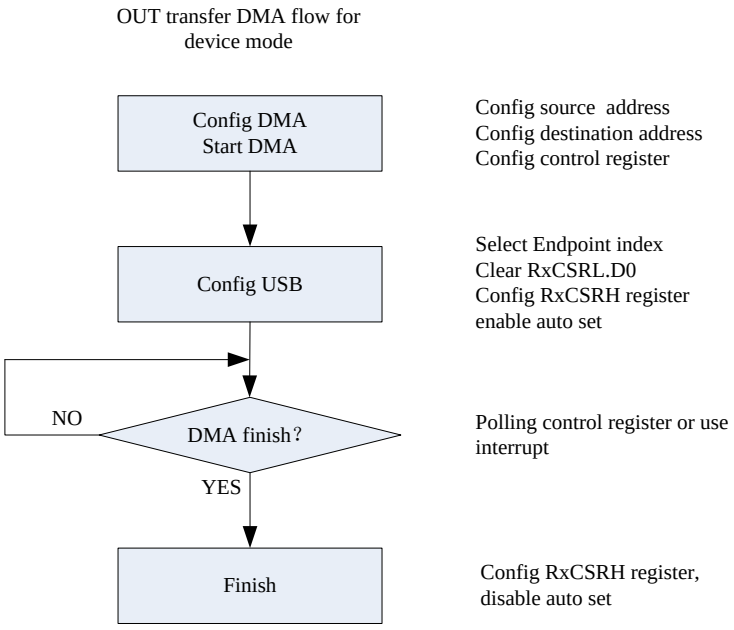


Figure 5.2.5 Device 模式 DMA Out 传输

USB 通过 DMA 在 ARM11 的控制下与 DDR 进行数据交互。USB 把 DDR 当作存储媒介实现 mass storage class device 的存储功能。

此外 ARM 程序负责控制 USB 控制器，实现 USB BUS 所必需的枚举操作，和各种通信握手。

Firmware 工作流程如下图所示，Mass storage class firmware 需要实现三大功能：首先解析 setup 数据包，进行枚举握手和其他功能握手；然后按照 mass storage class 所规定的协议

解析 SCSI command；最后启动 DMA，进行数据传输。

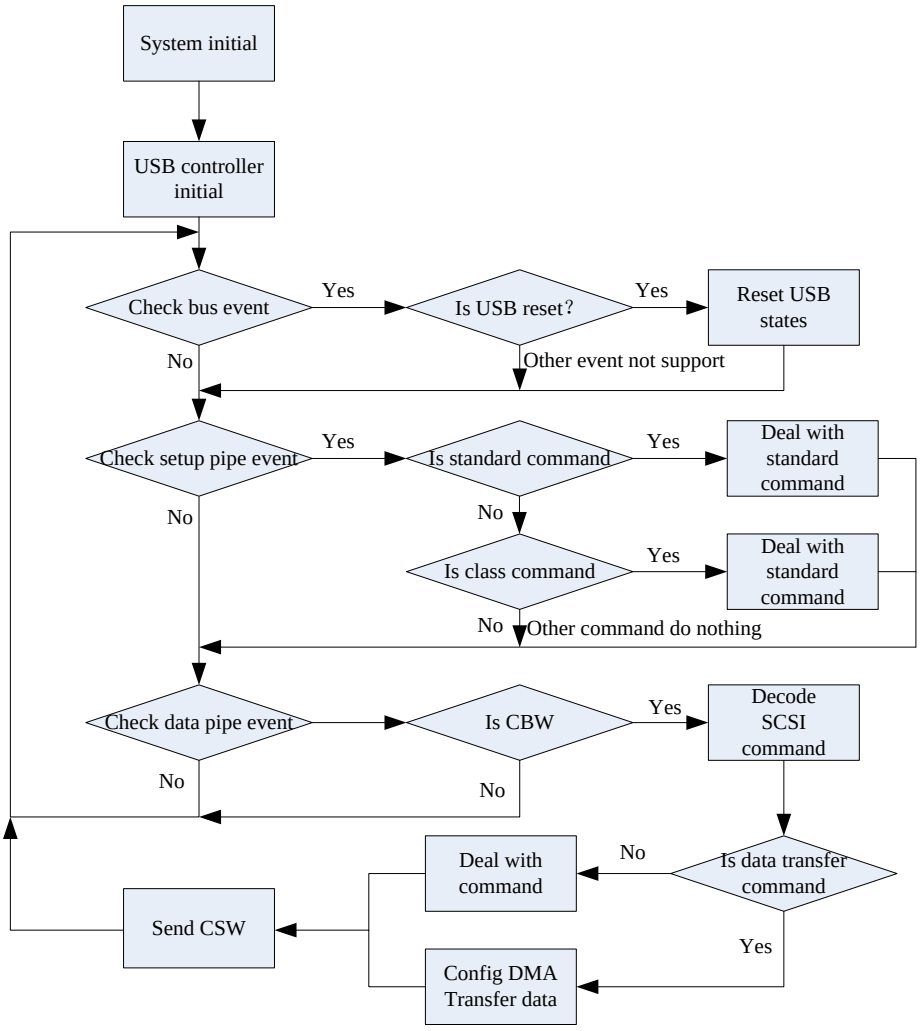


Figure 5.2.6　Firmware 工作流程如下图

完成初始化后，程序进入主循环。先后查询是否有 BUS event，setup event 和 data event 产生。

当 BUS event 产生后，判断是否是 USB BUS reset 事件。如果是则 reset 程序中各标志位，并等待 reset 结束。其他 BUS event 暂不支持。

收到 setup event 后按照 setup command 的格式对 setup package 进行解析。如果是 standard command，则按照标准的枚举协议进行处理。如果是 class command，则按照 mass storage class 中的 setup 协议进行处理。其他 command 不做处理。

收到 data event 后先按照 mass storage class 中的协议解析是否收到 CBW。不是则不进行处理。是则解析 CBW 中的 SCSI command。如果 SCSI command 是数据请求 command，则启动 DMA 进行数据传输。其他 command 按照 SCSI 协议进行处理。处理完毕后发送 CSW，报告处理结果。

4.3 数字音频接口

4.3.1 概述

全双工传输/接收的数字音频接口，通过 AHB 接口对模式寄存器进行相应的配置，可提供连续的数据格式。支持双通道 I2S 协议。支持 I2S、左对齐、右对齐等 5 种音频数据格式。

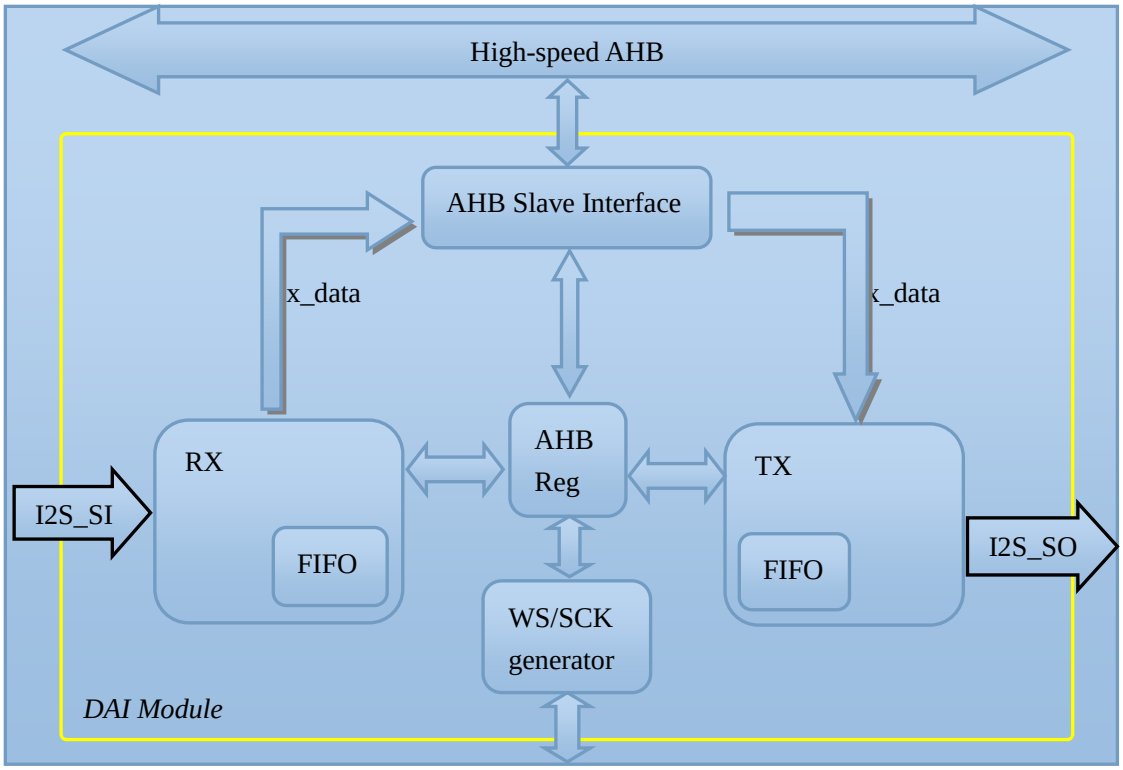
4.3.2 特点

- 全双工通信
- 双通道
- 支持采样率 24KHZ，32KHZ，44.1KHZ，48KHZ

4.3.3 功能描述

4.3.3.1 模块框图

主要包含 RX、TX 及时钟生成等模块，其内部结构及外围如下：



4.3.3.2 工作原理

在 I2S 模式下，信号 SCK 可作为输入或输出。在其他模式下，芯片作为 master 的情况下，SCK 只能作为输出。提供 2 个独立的数据通道，分别用于数据的传输和接收。每个数据通道都包含一个 FIFO，一个 FIFO 门限寄存器和 FIFO 标志寄存器，中断信号，DMA 请求及应答信号。到 ARM 核的中断则通过中断向量控制器(VIC)到达。DMA 的传输则由 DMA 引擎来控制。提供两个中断信号给 ARM，分别用于传输和接收。

传输控制寄存器和接收控制寄存器在复位后默认为 slave 模式，初始化时需要将这些寄存器设置为 master 模式。设置为 master 模式后，SCK 信号将为输出，并且依据

I2S_CLOCK 寄存器的 clk_div 分频 audio clk 生成。

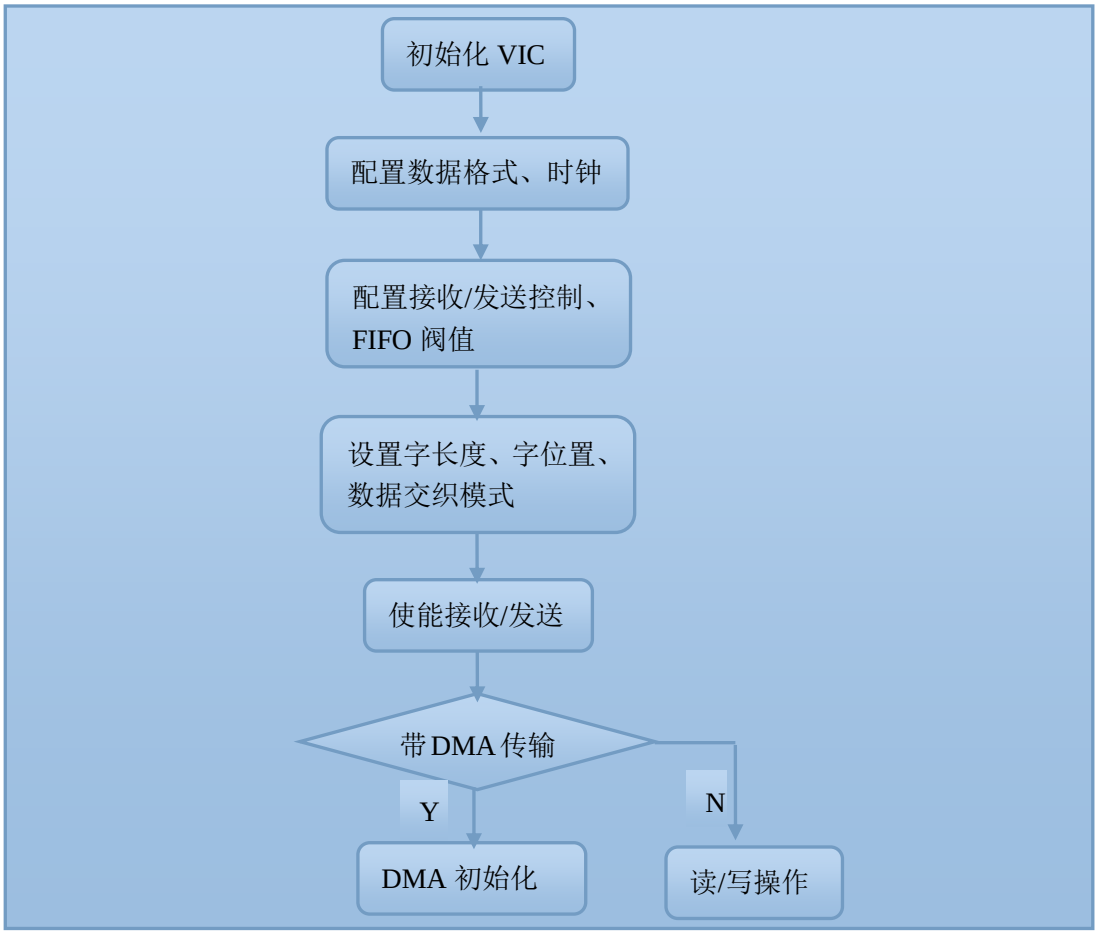
有两种复位方式，分别为上电复位和 FIFO 复位。上电复位后，所有寄存器将会复位。当初始化控制寄存器的 frst 位设置为 1 时，只有发送器和接收器 FIFO 复位，而其他寄存器将保持原有的值。当 txfrst 位设置为 1 时，只有发送器 FIFO 复位，其他寄存器包括接收器 FIFO 不受影响；反之，当 rxfrst 位设置为 1 时，只有接收器 FIFO 复位，其他寄存器包括发送器 FIFO 不受影响。

提供如下寄存器以确定音频数据格式：

- 模式寄存器：选择 5 种音频数据格式
- 字长度寄存器：选择音频数据的精度
- 字位置寄存器：选择音频数据的位置
- 时槽计数寄存器：DSP 模式下选择数据时间槽

4.3.4 工作方式

数字音频接口有两种工作方式，一种直接由 cpu 传输数据，另一种由 DMA 传输数据。在初始化前，须对 VIC 进行相应的初始化。D 接口配置流程为，设置模式寄存器选择模式，设置时钟控制寄存器以配置时钟的工作模式，再通过对发送控制寄存器和接收控制寄存器的设置，然后对字长度、字位置及数据交错模式等进行相应的设置。如果采用的是 DMA 传输数据，在进行输入/输出使能后，还需要对 DMA 进行相应的初始化。配置流程图如下：



4.4 模拟音频接口

4.4.1 概述

模拟音频接口用于完成模拟音频信号到数字音频信号的转换以及数字音频信号向模拟音频信号的转换。模拟音频接口与数字音频接口复用相同的 DMA 通道，两个音频接口不可同时工作，通过配置 AHB 通用寄存器中的 Audio_i2s_sel 寄存器完成外设选择。

4.4.2 特点

1. 支持 1 组 MIC 差分输入，采用 sigma-delta 结构 ADC。
2. 支持 1 组左右声道，差分输出，采用 sigma-delta 结构 DAC。
3. ADC 采样率支持最高 32KHz。
4. DAC 采样率支持最高 48KHz。
5. 支持 DMA 完成接收/发送数字音频信号的搬移。
6. 模拟音频支持 ADC 到 DAC 的数据流测试。
7. 支持内部产生 sine 波，完成对 DAC 的测试。

4.4.3 功能描述

模拟音频设备将芯片接口输入的 MIC 模拟差分信号转换为数字信号并存储于内存中；将需要发送的数字音频信号搬移至内部缓存，并通过 DAC 发送输出为差分的左右声道信号。

4.4.4 工作方式

模拟音频接口与数字音频接口复用相同的 DMA 通道，两个音频接口不可同时工作，通过配置 AHB 通用寄存器中的 Audio_i2s_sel 寄存器完成外设选择。

模拟音频接口工作时，需要先完成相关的模拟/数字部分的寄存器配置，软件配置 ADC 时，需要配置其为 MIC 处理模式。其后根据应用需要，配置 DMA 完成数据搬移工作。

模拟音频模块内部的 FIFO 数据位宽为 32 比特，深度为 64，左、右声道数据分别占用低、高 16bit 数据位置。

4.5 ADC

4.5.1 概述

ADC 为两路 10bit 逐次逼近型（SAR）模数转换器，实际应用中只用了两路模拟输入通道，通过多路选择器选择其中一路进行模数转换，输出 10bit 数字信号，主要用于低速监控应用，例如温度和光线传感器等。ADC 的工作频率可以通过寄存器配置时钟分频系数。

如果采样电压超过高阈值或小于低阈值，ADC 会产生中断信号，软件可以采用通过中断处理函数来做相关处理。

4.5.2 特点

1. 2 路模拟输入
2. 采样频率可配置，默认工作频率 6 Mhz
3. 通过总线配置寄存器
4. 支持 freerun 和 oneshot 模式
5. 高阈值和低阈值可配置
6. 采样数据超过阈值会产生软件中断

4.5.3 功能描述

ADC 分为模拟部分和数字部分，可以通过寄存器配置为 freerun 或 oneshot 模式，oneshot 模式只执行一次模数转换，freerun 模式不断重复执行模数转换。两路模拟输入通过内部的多路选择器选择一路进行模数转换，产生 10bit 的数字信号。

ADC 的数字部分，主要用于采集到的数字信号阈值比较和中断信号的产生。如果模数转换结果大于高阈值或者小于低阈值产生中断信号

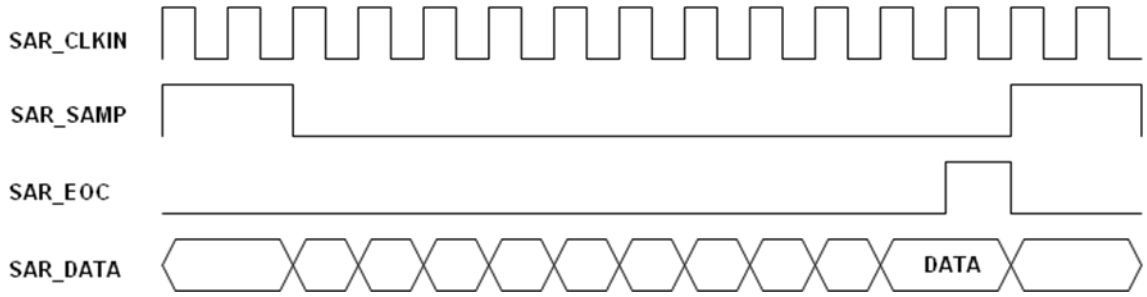


Figure 5.6.2 ADC 模数转换时序图

4.5.4 工作方式

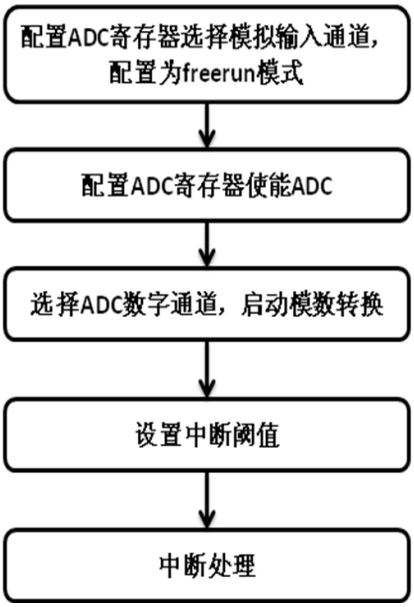


Figure 5.6.3　ADC 配置流程图

4.5.5 模拟 IP

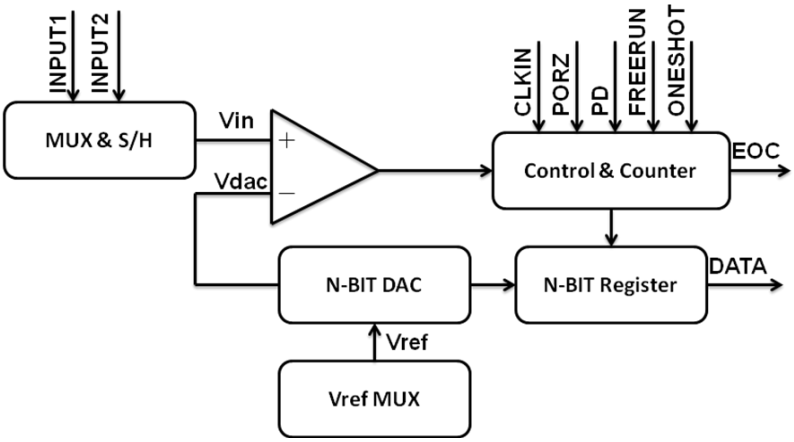


Figure 5.6.3　ADC 模拟 IP 功能框图

SAR ADC 实质上是实现二进制搜索算法。其基本结构如上图所示，模拟输入电压 Vin 由多路选择器选择模拟输入通道，采样/保持电路保持，为实现二进制搜索算法，N 位寄存器首先设置中间刻度(即:100… …00，最高位为 1)。这样强行数字/模拟转换器(DAC)输出 Vdac 为 Vref/2，其中 Vref 是提供给 ADC 的基准电压。然后，比较确定 Vin 是小于还是大于 Vdac。如果 Vin > Vdac，则比较器输出逻辑高电平或 1，N 位寄存器的最高有效位(MSB)保持 1；相反,如果 Vin < Vdac，比较器输出逻辑低电平或 0，N 位寄存器的 MSB 清为 0。随后，SAR 控制逻辑移至下一位，并将该位置为高电平同时使该过程一直持续到最低有效位(LSB)。上述操作结束后，也就完成了转换，N 位转换结果储存在寄存器内，以并行的方式输出。N 位 SAR ADC 需要 N 个比较周期，同时在当前一位转换完成之前不得进入下一次转换。

4.6 UART

4.6.1 概述

UART 通用异步发送器模块包含 uart0,uart1,uart2 三个端口。每个端口都支持中断访问的模式，支持 5，6，7，8bits 的数据长度，奇偶校验，停止位，并包括一个波特率控制器。uart2 单独支持 RS485 协议输出。

4.6.2 特点

UART 模块配合波特率控制器支持对外部端口上的异步信号进行收发的同步处理。UART 支持的波特率通过寄存器 DLL，DLH 可配。收发单个数据的位数通过线控制寄存器 LCR 可配置，支持 5~8bits 四个长度。uart2 同时支持 RS485 模式输出。

每个 UART 模块包含收发数据进行缓冲的 FIFO 功能，帧校验功能（支持 break 收发模式）。支持对校验结果输出中断，校验结果根据线状态寄存器 IIR 可查询。支持根据输入输出 FIFO 水线产生中断，中断水线 FIFO 控制寄存器 FCR 可配置。所有中断合成一个中断信号输出，中断源通过寄存器 IIR 进行查询。并且三个中断的使能通过寄存器 IER 可配置。

UART 寄存器支持通过 APB 桥进行读写访问。所有寄存器利用 8 组地址通过 latch register 的方式对 11 个寄存器进行访问。

4.6.3 功能描述

UART 模块功能框图如下：

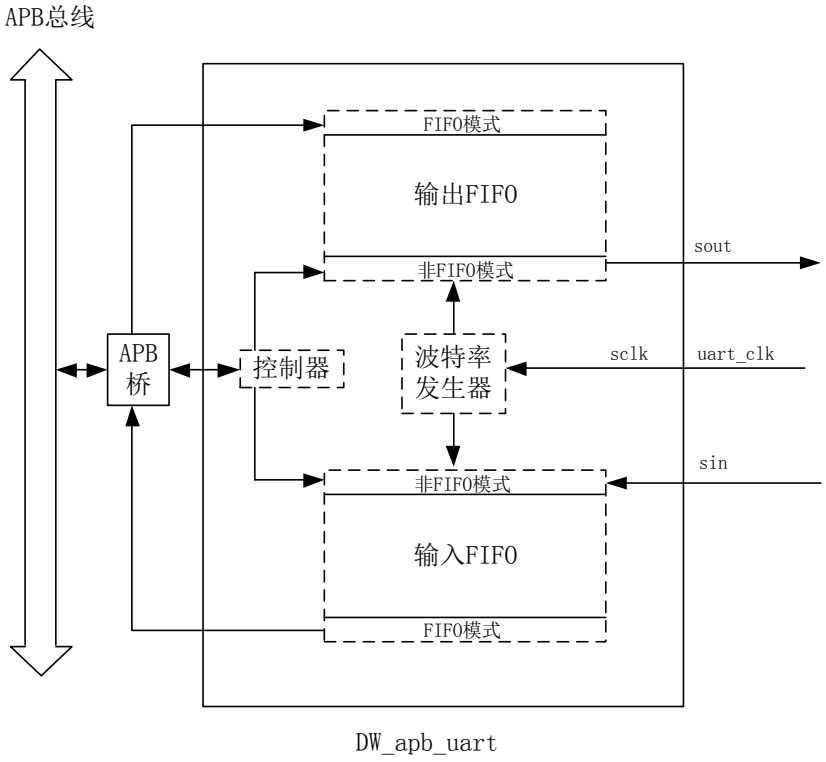


Figure 4.6.1 UART 模块功能框图

- 支持异步收发波特率可配置。

  uart 模块采用输入输出独立的两个数据线 sin 和 sout 进行异步信号输入输出。

  uart 模块包含 pclk,sclk 两个输入时钟信号，pclk 为 APB 时钟控制所有内部寄存器，sclk

由 rct 模块产生的 uart_clk 输入。波特率发生器根据 DLL,DLH 配置的 divisor 参数对 sclk 进行分频产生 boaud_clk 输出给输入输出 FIFO 控制 sin 采样和 sout 输出的数据的时钟周期，计算公式如下：

boaud_clk=sclk/(16*divisor)

输出和采样数据时序如下图，字符数据的长度根据控制器可配。输出 N 位字符数据后，根据前面输出数据 bit 1 的数量再输出 1 位奇偶校验位，以保证字符数据加奇偶校验位共输出奇个数或偶个数的 bit 1。这之后输出为高电平的 stop bit。两个字符之间为高电平的空闲位。

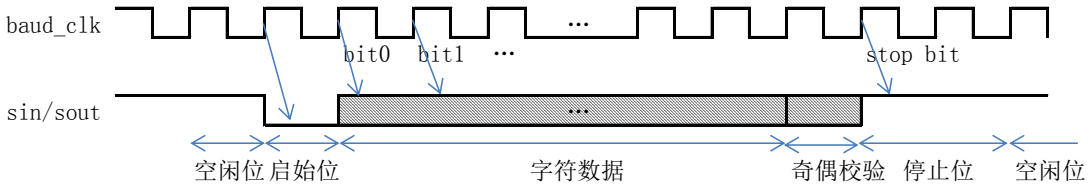


Figure 4.6.2 uart 外部接口收发时序图

同时支持发送奇偶校验位，以及按照奇/偶校验模式检查接收到数据的奇偶校验是否正确。通过配置 LCR 寄存器，收数据的奇偶校验功能可关闭，同时 sout 输出时不输出奇偶校验位。可配置选择奇或者偶数的校验。可配置 stop bit 的长度为 1~2bit。

- 输入输出 FIFO

  uart 包含一个输入 FIFO 和一个输出 FIFO。使得可以批量输入输出数据，减少对 CPU 的占用次数。FIFO 深度为 16x8bits。Uart FIFO 可以根据水线产生中断，CPU 可以通过 APB 总线从 RBR 寄存器读走数据，或者向 THR 寄存器写入要发送的数据。通过配置寄存器 FCR，可以控制：

  1.输入中断水线为 1 个字符，1/2 满，1/4 满，差 2 个字符满 4 种情况；

  2.输出中断水线为空，2 个字符，1/4 满，1/2 满 4 种情况；

  3. TX，RX FIFO 单独复位

  4. FIFO 为 FIFO 模式或非 FIFO 模式。FIFO 模式切换时，RX/TX 的 FIFO 都被复位。

- 数据收发检查和告警。

  uart 模块对数据的收发进行检测以保证数据正确性，从而通过寄存器 LSR 产生 3 个告警：1.stop bit 错误告警，如果一帧最后没有检查到 stop bit 将对应告警寄存器置 1；2. 接收到的数据奇偶校验错误，产生告警；3. 在非 FIFO 模式下，如果一个数据还没有被输出或者读走，就收到新的数据，产生覆盖告警，告警产生时，旧的数据将被覆盖丢失。

  另外 uart 在 RBR 至少收到一个数据时，会产生 DR 告警，该告警被读清。

  支持 break 收发模式，当配置 LCR bit6 为 logic 1 时，sout 输出保持为低，进入发数据 break 状态，直到 LCR bit6 清除。如果输入的 sin 信号保持为低的时间超过 start_time+data_bits+parity+stop_bits 的长度，uart 模块进入收数据 break 状态，LSR 的 bit 4 置 1 产生 break interrupt。该告警读清。

- LOOP BACK 测试模式。

  uart 模块支持 Loop Back 模式对模块内部数据通道进行检查。配置 MCR 寄存器第 4 个 bit 为 1 后配置为 Loop Back 模式。该模式下 sout 输出被固定为 1，串行的输出数据被从内部输入到 sin。该模式下所有的中断功能都可以使用。该模式是为了检查 uart 内部数据通道是否正确。

- 支持 RS485 模式。

  uart2 支持在非 auto_flow 模式下（默认模式），rts_n, dtr_n 保持输出高电平。

4.7 SPI

4.7.1 概述

SPI 是串行外设接口（Serial Peripheral Interface）的缩写，是一种高速，全双工，同步的通信总线。其由四根线组成，分别是 SDI（数据输入）、SDO（数据输出）、SCLK（时钟）、CS（片选）。

（1）SDO – 主设备数据输出，从设备数据输入；

（2）SDI – 主设备数据输入，从设备数据输出；

（3）SCLK – 时钟信号，由主设备产生；

（4）CS – 从设备使能信号，由主设备控制。

4.7.2 特点

拥有两个 SPI 控制器。第一个控制器为主设备模式，最多支持 6 个 SPI 从设备。第二个控制器也是主设备模式，支持 1 个 SPI 从设备。另外，第二个控制器支持自动 SPI（TSSI）传输模式。

在 TSSI 模式传输时，首先在关闭 SPI 的情况下，将需要传输的数据填入 FIFO，然后通过配置 TSSI 映射到 SPI 的寄存器，开启 SPI 控制器进行自动数据传输。

4.7.3 工作方式

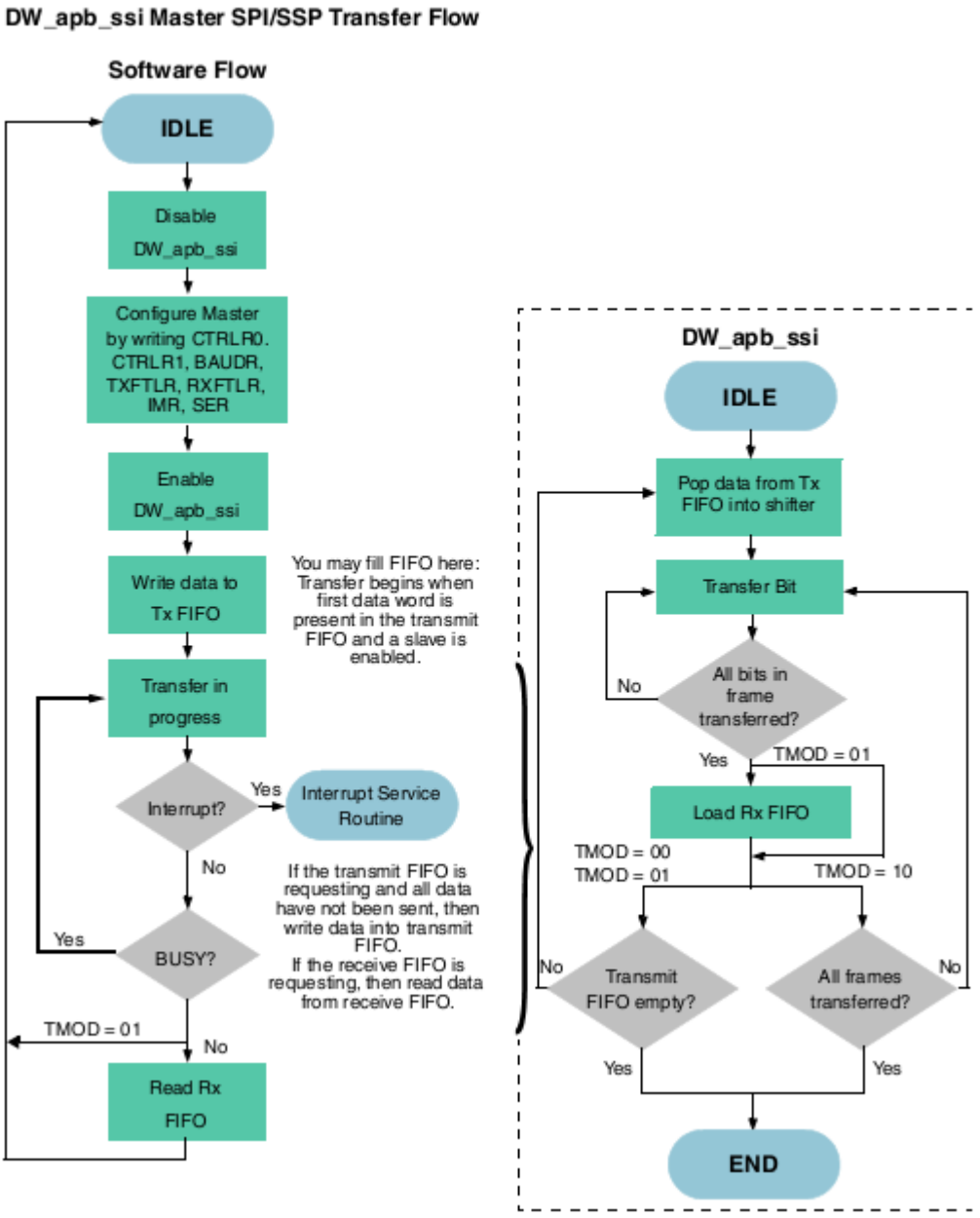


Figure 4.7.1 SPI 配置流程图

配置流程：

1. 关闭 SPI
2. 配置 CTRLR0，CTRLR1，BAUDR，TXFTLR，RXFTLR，IMR，SER 寄存器
3. 启用 SPI
4. 将数据写入发送 FIFO
5. 传输数据
6. 中断状态查询，没有中断产生执行步骤 7，产生中断执行中断处理流程
7. 传输数据忙状态查询，传输忙执行步骤 5，传输完成执行步骤 8
8. 传输模式判定。单传输模式，完成。其他模式，读取接收 FIFO 内数据

4.8 SFLASH

4.8.1 概述

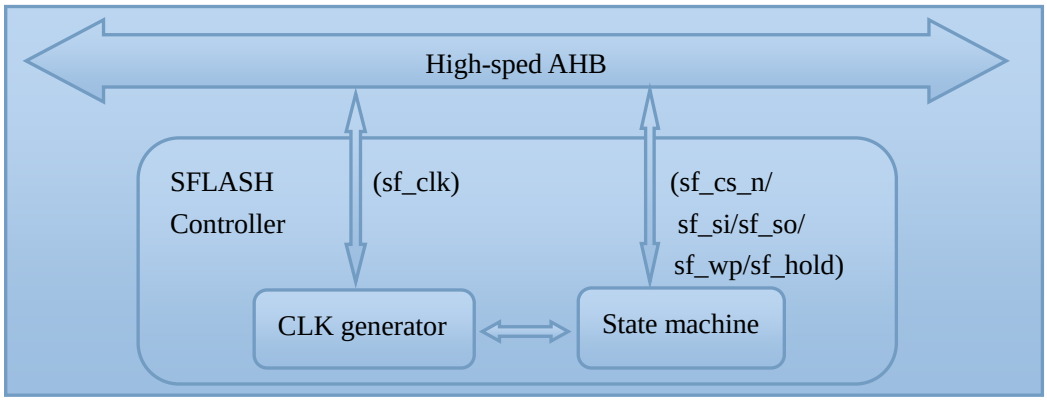
SFLASH 控制接口用于 SFLASH 存储器的存取。可以通过将命令写入命令寄存器从而对该接口进行控制，也可在数据端口进行数据的读写操作。该控制接口采用串行通信的方式，实现处理器与 SFLASH 存储器间的通信。该接口运行在 138MHZ 频率下，同时产生串行时钟，以控制 SFLASH 存储器。在默认情况下串行时钟速率为 13.8MHZ，并且可切换最大速率为 69MHZ。

4.8.2 特点

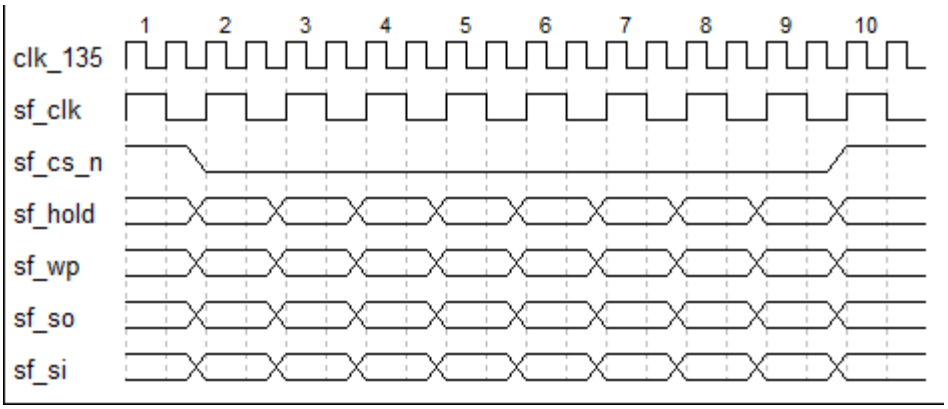
- SPI Flash x1（支持 SPI-Nor 及 SPI-Nand，CS 片选）
- 支持 64Mbit SFLASH
- 命令模式
- 支持单线/双线/四线 I/O 模式
- 可配置的串行时钟频率 69M/34.5M/23M/17.25M/13.8M

4.8.3 功能描述

模块功能框图如下：



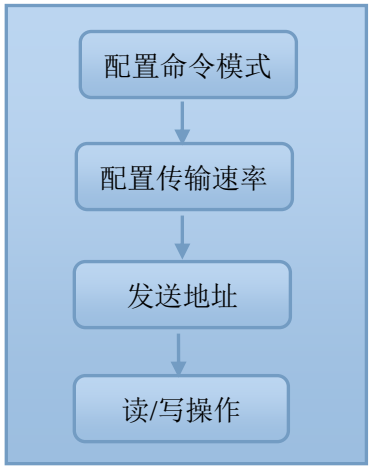
每次往数据端口进行读写操作，都需要传输一个字长度的数据，如果 4 个字节的数据不是全部需要的，取低字节有效。在 dummy 周期下，数据只取最高字节有效。命令模式下，写入数据端口的数据是以字节为单位。



4.8.4 工作方式

SFLASH 运行的标准流程为先执行 SFLASH 命令，命令由 SFLASH_COMMAND 寄存器写入。然后通过 SFLASH_PORT 寄存器进行数据读写操作（依据 SFLASH 命令）。如果写入的命令需要更多的地址类的数据，这些数据必须通过写入 SFLASH_PORT 寄存器来提供。每次对 SFLASH_PORT 寄存器的读写必须由 SFLASH_HIF_ack 信号应答，写操作情况下，应答信号表示本次写操作完成；读操作情况下，表示数据已读取完成。

（右图为配置流程图）



4.9 SDIO

4.9.1 概述

SDIO 模块不仅可以连接 SD 存储卡，同时也能连接支持 SDIO 接口的蓝牙适配器，GPS 接收器，无线网卡等设备。

4.9.2 特点

支持标准容量卡（SDSC，不超过 2GB ），高容量卡（SDHC，大于 2GB 又不超过 32GB），和扩展容量卡（SDXC，大于 32GB 又不超过 2TB ）。支持 1 位，4 位和 8 位传输模式。支持 MMC 4.0 协议。

4.9.3 功能描述

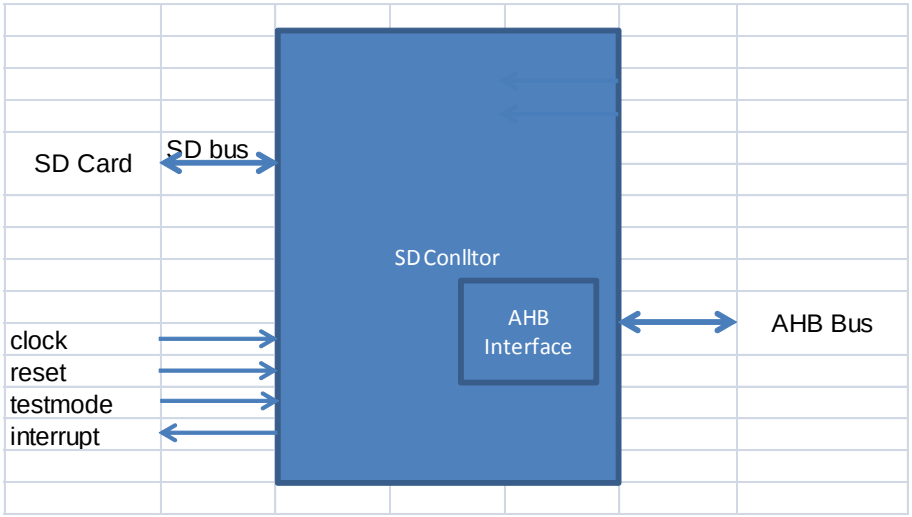


Figure 4.9.1 SDIO 功能框图

4.9.4 工作方式

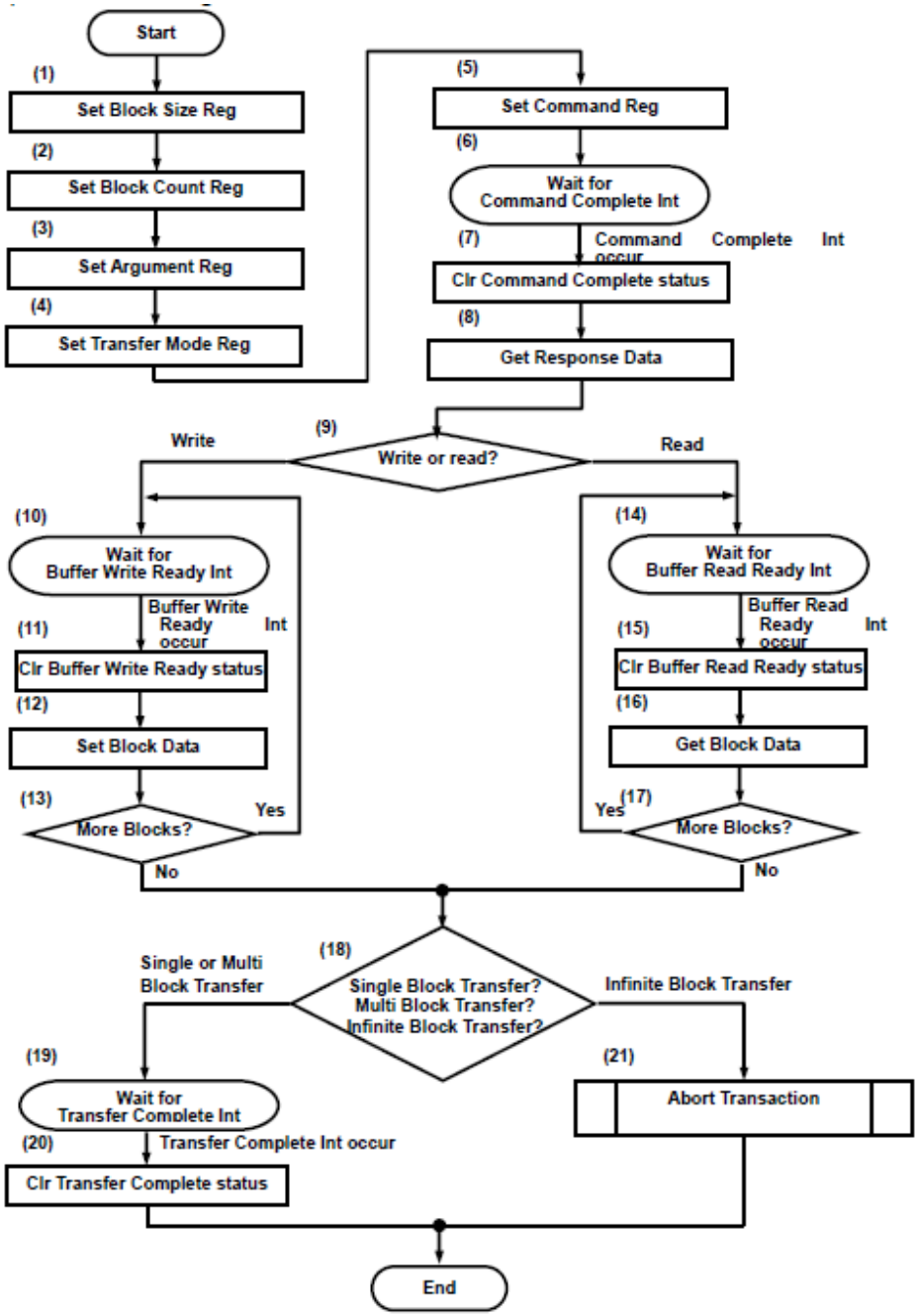


Figure 4.9.2 SDIO 普通模式配置流程图

普通模式（非 DMA）配置流程：

1. 配置数据块格式寄存器
2. 配置数据块计数寄存器
3. 配置参数寄存器
4. 配置传输模式寄存器
5. 配置命令寄存器
6. 等待命令传输完成中断
7. 清除普通中断状态寄存器的命令传输完成中断

8. 读取响应寄存器
9. 写入 SD 数据或者从 SD 卡读取数据判定
10. W 等待普通中断状态寄存器的写入 buffer 就绪中断产生
11. W 清除 buffer 就绪中断
12. W 将块数据写入 buffer 寄存器。块数据大小应该符合步骤 1 的配置
13. W 如果为多块数据传输，重复步骤 10W-12W，直到所有数据都被写入
10. R 等待普通中断状态寄存器的读取 buffer 就绪中断产生
11. R 清除 Buffer 就绪中断
12. R 从 buffer 寄存器读取数据。块数据大小应该符合步骤 1 的配置
13. R 如果为多块数据传输，重复步骤 10R-12R，直到所有数据都被读取
14. 如果为单/多块数据传输，执行步骤 15，如果是为持续传输模式，执行步骤 17
15. 等待普通中断状态寄存器的传输完成中断产生
16. 清除传输完成中断
17. 中断传输，SD 卡用 CMD12 命令，SDIO 用 CMD52

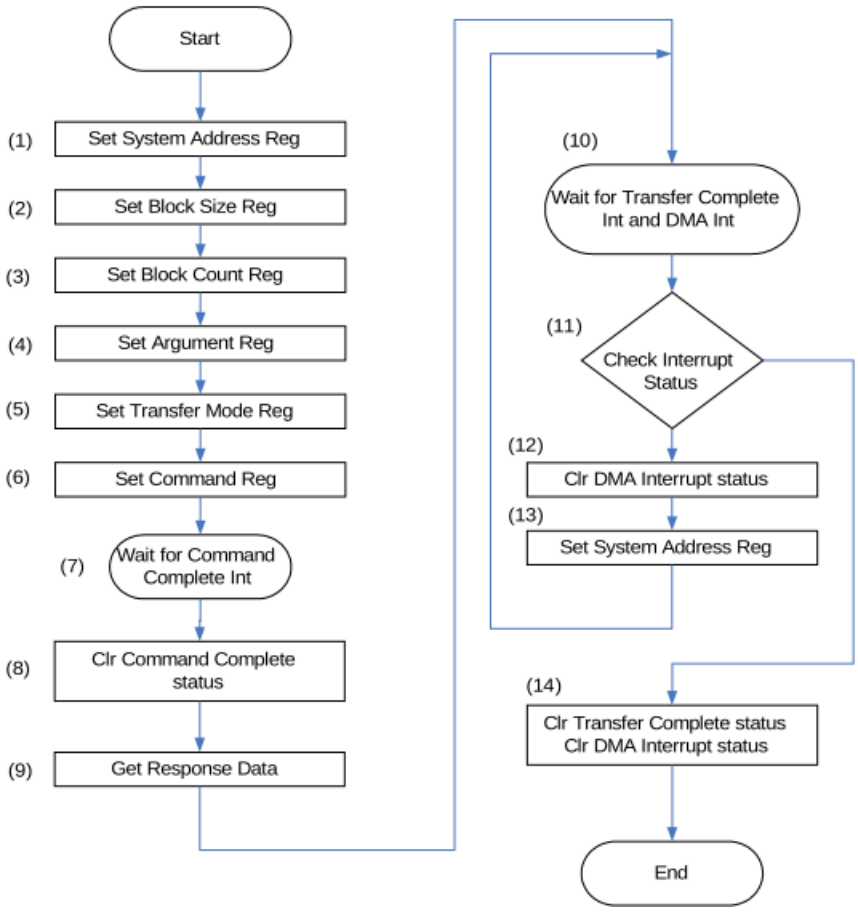


Figure 4.9.3 SDIO DMA 模式配置流程图

DMA 传输模式配置流程：

1. 设定 DMA 地址
2. 配置数据块格式寄存器
3. 配置数据块计数寄存器
4. 配置参数寄存器

5. 配置传输模式寄存器
6. 配置命令寄存器
7. 等待命令传输完成中断
8. 清除普通中断状态寄存器的命令传输完成中断
9. 读取响应寄存器

   DMA 方式写入 SD 卡数据：命令响应接收完成后，SD 控制器将对 AHB 总线发送请求。得到 AHB 的授权后，SD 控制器开始从系统内存中读取一块数据，并将数据填入 FIFO 前半部分。当第一块数据就绪后，此块数据开始写入 SD 卡。向 SD 卡写入第一块数据的同时 SD 控制器开始读取第二块数据并填入 FIFO 后半部分。一旦 FIFO 空出，SD 控制器就将从系统内存读回数据，直到读完所有数据。

   DMA 方式读取 SD 卡数据：从 SD 卡读取的第一块数据填入 FIFO 前半部分。FIFO 内的数据就绪后，控制器将对 AHB 总线发送请求。得到 AHB 的授权后，SD 控制器开始将 FIFO 内数据写入系统内存。向系统内存内写入数据的同时，SD 控制器接收第二块数据并填入 FIFO 的后半部分。FIFO 内就绪的数据不断被写入系统内存，直到所有数据传输完成。当 FIFO 被填满的时候，SD 控制器通过读等待功能或暂停 SD 总线时钟的方式暂停接受数据。
10. 等待传输完成中断和 DMA 中断
11. 如果传输完成中断产生，执行步骤 14，如果 DMA 中断产生，执行步骤 12
12. 清除 DMA 中断
13. 设定其他数据的 DMA 地址后执行步骤 10，进行其他数据的 DMA 传输
14. 清除传输完成中断和 DMA 中断

4.10 PWM

4.10.1 概述

- 4 个 PWM 输出
- 可编程的占空比和周期

4.10.2 特点

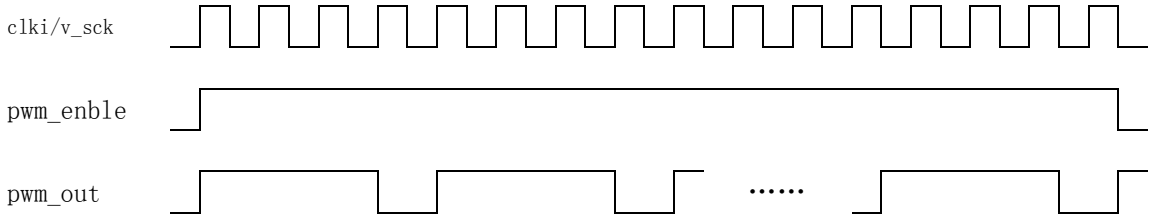
这里包含 4 个 PWM 输出，PWM 的输出通过 GPIO 配置选择 pin 脚（具体参考 GPIO 配置章节）。每个 PWM 内的寄存器组地址都是独立的。当使能时，PWM 将输出一个重复序列的逻辑高电平和低电平。每个逻辑高或逻辑低的持续时间是可编程的。

PWM 模块挂在 APB 总线上，通过 APB 对 PWM 进行读写操作，以配置 PWM 的输出。PWM 的输出，通过分频 gclk_pwm（常规模式）产生。

在一些应用里，PWM 被用于镜头模组里，控制直流电机。并且满足 PWM 波形的改变与视频捕捉接口时钟间的同步。为避免 PWM 波形的不连续，以使电机噪音最小化，软件的执行需要参考下面的指南说明。

- 编程 PWM_dat xon/xoff 需满足（xon + xoff）* n = vsync 周期，n 为整数。这将保证 vsync 周期内的 PWM 脉冲周期个数为整数。
- 通过改变 xon：xoff 以控制占空比
- 通过控制 xon 与 xoff 的和以控制 PWM 频率（pwm freq = 1/time_equivalent(xon+xoff)）
- 按上述方式设置 PWM freq 以产生可听见的频率（大于 20KHZ）

PWM 模块（单个 PWM 输出）接口简要时序图如下：

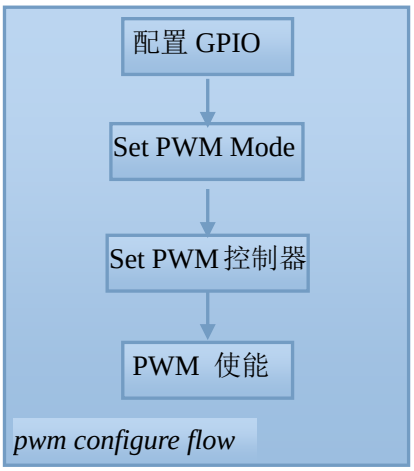


PWM 接口时序图

4.10.3 工作方式

在启动 PWM 输出前，需要通过 APB 总线配置 PWM 模式寄存器选择模式，并设置 PWM 控制寄存器配置 PWM 的脉冲周期及占空比。

PWM 启动前配置流程如右图所示：



4.11 I2C

4.11.1 概述

I2C 总线是一种两线式串行总线，用于连接控制器和外围设备。其只有两条总线线路：串行数据线 SDA，串行时钟线 SCL。每个连接到总线的器件都可以通过唯一的地址和主控制器进行通信。

4.11.2 特点

拥有两个 I2C 控制器。双向数据传输位速率最高支持 400kbit/s。支持 7 位地址寻址和 10 位地址寻址。

4.11.3 功能描述

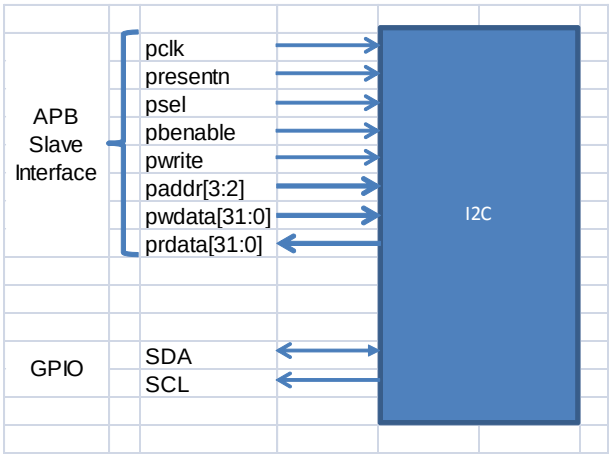


Figure 4.11.1 I2C 功能框图

4.11.4 工作方式

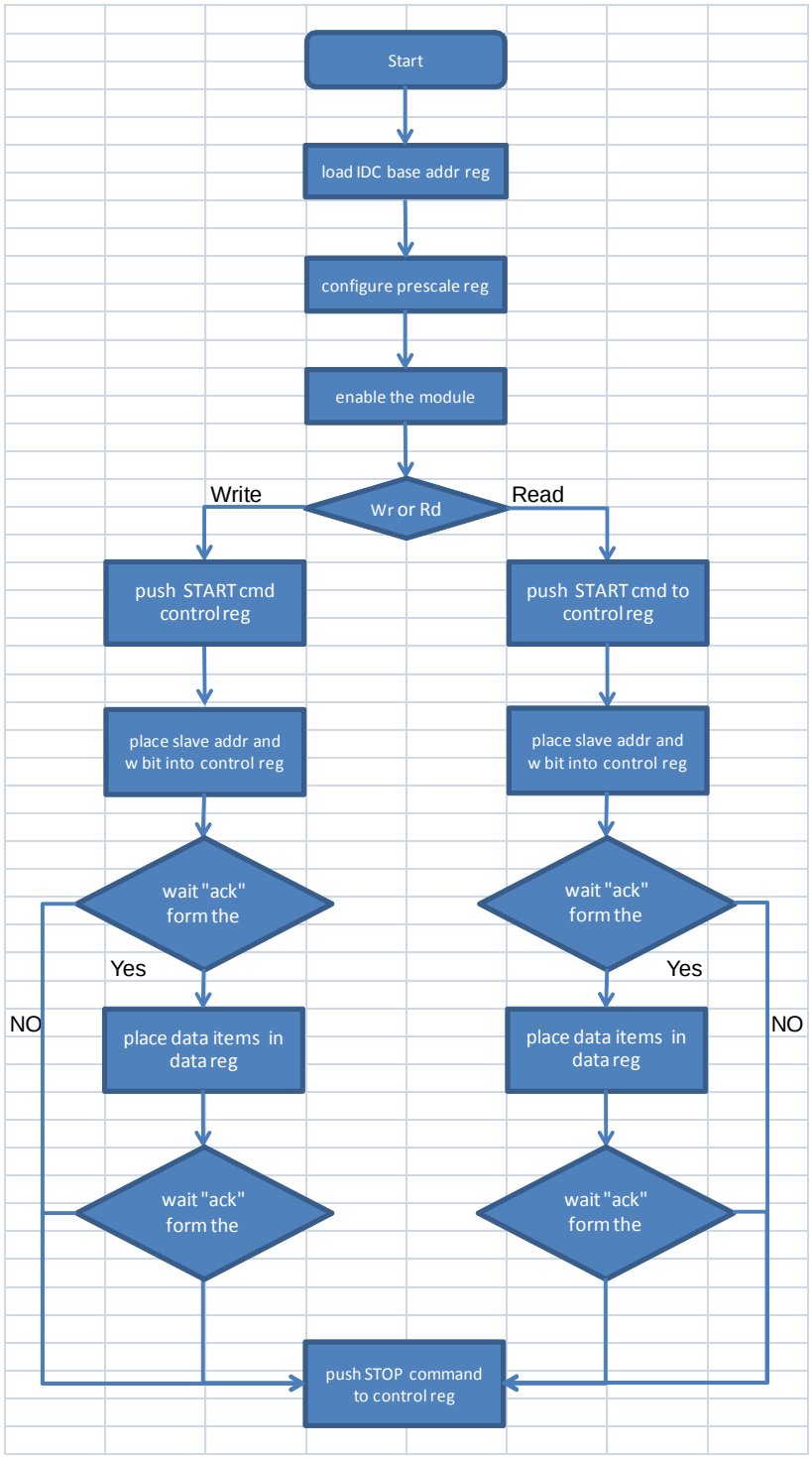


Figure 4.11.2 I2C 普通模式配置流程图

普通模式配置流程：

1. 选中 I2C 控制器
2. 配置 I2C 时钟，APB bus clock/ (4*(prescale[15:0]+1)
3. 启用 I2C
4. W 读取操作，发送开始信号

R 写入操作，发送开始信号

5. W 发送从设备地址，并设置为写入数据模式
   R 发送从设备地址，并设置为读取数据模式
6. 等待从设备 ACK 信号响应
7. W 发送数据
   R 接收数据
8. W 等待 ACK 信号
   R 发送 ACK 信号
9. 传输完成，发送停止信号

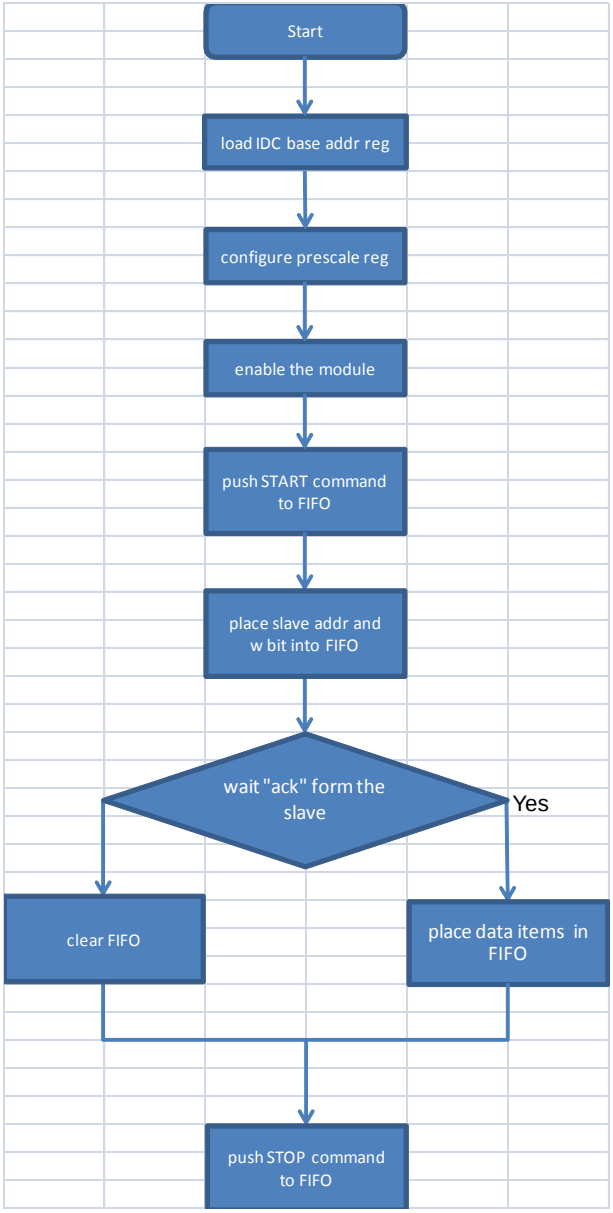


Figure 4.11.3 I2C 自动模式配置流程图

自动模式只能用于写入从设备数据。自动模式与普通模式配置流程基本相同。区别在于自动模式数据寄存器拥有 63 个字节深度的 FIFO。所以可以一次输出多个地址和数据到从设备，而不需要每写入一个地址或数据后等待 ack 信号。

4.12 VOUT

4.12.1 概述

VOUT 模块挂在 AHB 总线上，通过 AHB 和 CPU 实现对其寄存器配置。

支持一路标清数据输出，VOUT 通过合理的寄存器配置向 CPU 发起数据请求，并对请求回来的图像数据和 OSD 数据处理并 blending 后输出显示。

包含了 OSD 和图像数据的读取，OSD 数据格式转换，OSD 的 scaler，及 CVBS 编码等功能。

VOUT 模块主要实现以下功能：

1. Video 和 OSD 图像数据的读取和处理；
2. Video 和 OSD 图像的 mix；
3. CVBS 编码；
4. OSD 的 scaler；
5. Mix 后图像显示。

4.12.2 特点

- VOUT 模块支持 YUV422 的图像数据格式；
- 一路标清数据输出；
- 图像数据的窗口大小及水平垂直位置偏移可配置；
- 可配置的背景颜色；
- 支持 OSD mapped 的数据格式，查表大小 256x32bit，查找表数据寄存器可配置；
- 支持直接的 OSD 数据；
- PAL/NTSC 编码器

4.12.3 Video DAC

VDAC 是一个纯模拟 IP，实现 10bit 数字信号到模拟的输出。通过寄存器接口对其配置。

4.13 Sensor 接口

4.13.1 概述

GK8602A 支持与 SONY、Aptina、OV、Panasonic 等主流高清 CMOS 无缝对接，目前只支持并口数据输入，不支持 MIPI 和 LVDS。

GK8602A 的 Sensor 接口分为三部分，一部分是芯片输出给 sensor 的参考时钟，系统会根据不同 Sensor 的需求提供 24M/27M/37.125M/74.25M 的时钟；一部分是 I2C 模块来实现 Sensor 芯片的控制；还有一部分则是接收 Sensor 输出的视频数据，随路时钟以及行、场同步信号，这部分接口电路的功能由 Video Interface 模块来完成。

4.13.2 特点

Video Interface 模块能处理 8bit/10bit/12bit/14bit 的 sensor Bayer Pattern 的 RAW 输入数据，数据采集窗口可调。支持外部同步和内部同步两种同步方式，同步信号的极性可编程控制。支持内部 Test Pattern 产生模式。

4.13.3 功能描述

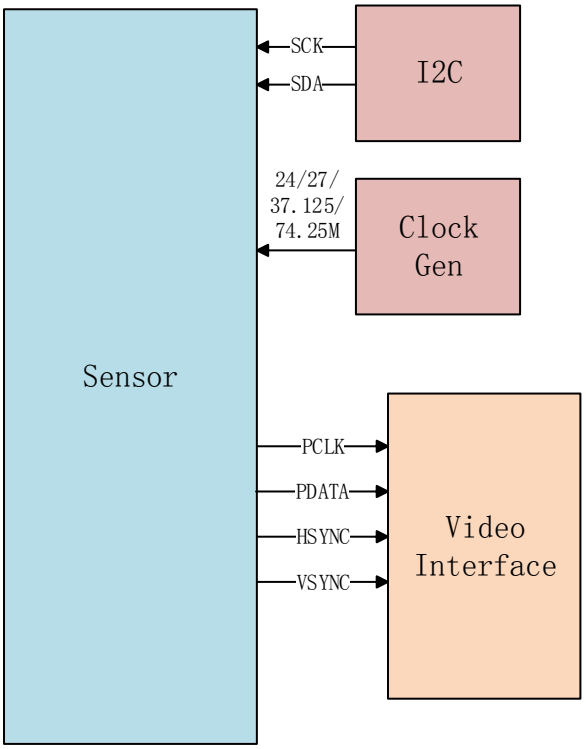


Figure 4.13.1 Sensor 接口模块示意图

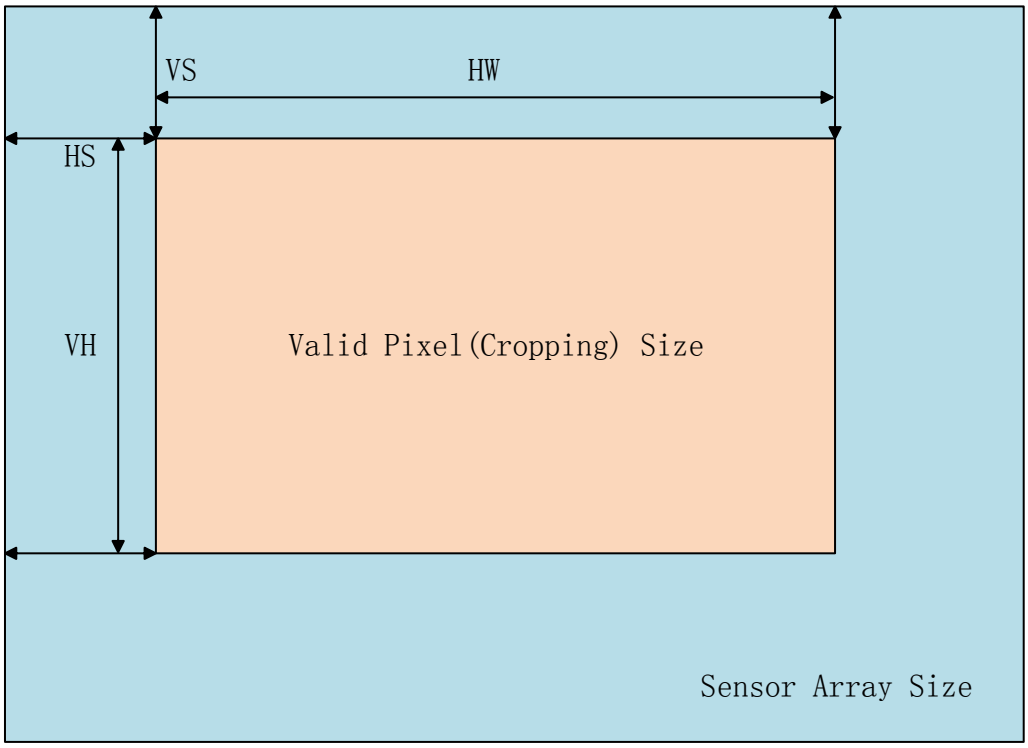


Figure 4.13.2 Sensor 图像窗口示意图

4.13.4 工作方式

通过 I2C 对 Sensor 进行相关配置，不同厂家的 Sensor 会提供不同的寄存器对 Sensor 输出图像的尺寸，同步方式，数据位宽，旋转，增益，降噪等进行设置。有的 Sensor 自带 PLL，我们只需要给一个 24/27MHz 的参考时钟，Sensor 就会根据寄存器的配置自动生成相关分辨率所需的时钟，有的 Sensor 不带 PLL，需要我们根据不同需求给 Sensor 合适的时钟。

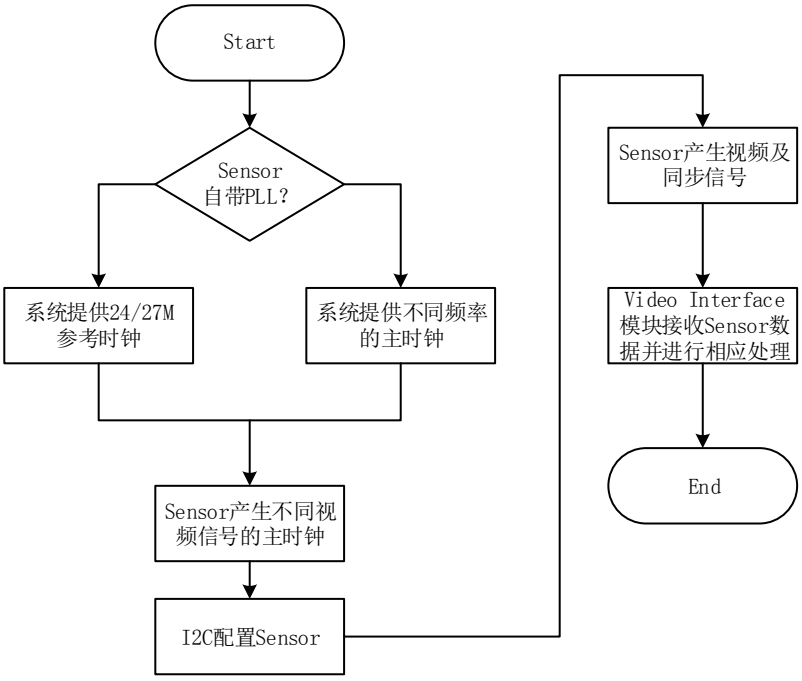


Figure 4.13.3 Sensor 接口模块工作流程图