|  |
| --- |
| **Software Unit Design Specification**  GEM BCM LFDriver  PATAC Current Logo PANTONE 1788U2X |
| **REVISION:**1.0.0 |
| **DATE:**2014-12-09 |
|  |
| **Author** songchenhong  **Approval** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Change Description** | **Revised By** |
| 1.0.0 | 2014-12-09 | Initial release | songchenhong |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 目的 4](#_Toc406592760)

[2. 适用范围 4](#_Toc406592761)

[3. 参考文档 4](#_Toc406592762)

[4. 定义及术语 4](#_Toc406592763)

[5. 概要设计 4](#_Toc406592764)

[5.1. 运行环境 4](#_Toc406592765)

[5.2. 设计概述 4](#_Toc406592766)

[5.3. 通信流程 5](#_Toc406592767)

[5.4. 结构关系 6](#_Toc406592768)

[5.4.1. 文件结构 6](#_Toc406592769)

[5.4.2. 分层结构 6](#_Toc406592770)

[5.5. 接口设计 6](#_Toc406592771)

[5.5.1. 外部接口 6](#_Toc406592772)

[5.5.2. 内部接口 6](#_Toc406592773)

[6. 详细设计 7](#_Toc406592774)

[6.1. 算法设计 7](#_Toc406592775)

[6.2. 函数设计 7](#_Toc406592776)

[6.3. 详细列表信息 7](#_Toc406592777)

[6.3.1. 数据列表 7](#_Toc406592778)

[6.3.2. 函数列表 8](#_Toc406592779)

[7. 附录 1 10](#_Toc406592780)

# 目的

本文档定义了GEM BCM 中LFDriver模块的软件设计说明，为该软件编码提供指导及约束。

# 适用范围

本文档适用于GEM BCM软件的设计开发过程,并限制于所有参与该项目软件开发的人员。

# 参考文档

*《LFDriver\_Requirement.doc》V1.0*

*《AUTOSAR\_SWS\_LFDriver.doc》V1.0*

# 定义及术语

*N/A*

# 概要设计

* 1. 运行环境

|  |  |
| --- | --- |
| *MCU型号及版本* |  |
| *编译器名称及版本* | Green Hills 6.1.4 |
| *编译器配置* | *详见编译器配置说明相关文档* |
| *相关外围芯片型号及版本* |  |

* 1. 设计概述

LF Driver需要实现以下功能：

* 通信驱动用的PWM模块、GPIO、Clock的Initialization和de-Initialization；
* 对通信内容进行分解重构使其符合曼彻斯特编码规则以及特殊Byte的发送需求；
* 配置DMA的功能，使其能够每次PWM的中断触发传送数据，改变PWM的周期和占空比以及脉冲极性（High to Low/Low to High）的设定；
* 通信开始以及通信停止的控制；
* 获取通信的状态（BUSY or IDLE）以及通信结束的回调函数。

*5.2.1*通信准备

1). 首先对模块初期化，为了符合125KHz的传输速率（8us 1Bit），生成PWM的TAU模块的Count Clock选择PLLCLK（PLL设为10MHz），PWM的周期设定channel的周期设为80；

2). 为了使发送的PWM符合符合曼彻斯特编码规则以及特殊Byte的发送需求，需对通信内容做如下处理：

* 将传入的通信内容，按每一Bit的需求设定的周期和占空比的数值存放入DMA传送的周期和占空比的Buffer中。
* 将通信内容的每一个Bit进行0或1的判断，为1则TAU的TOL寄存器的设定值设为1（Negative logic即Low to High），为0 则TAU的TOL寄存器的设定值设为0（Positive logic即High to Low），并存放入DMA传送的TOL的Buffer中。

3). 配置DMA，并启动DMA，等待TAU启动后中断触发DMA的数据传输。

*5.2.2*通信开始

启动TAU的Counter开始PWM的输出，DMA自动完成数据搬移，改变PWM的周期和占空比以及PWM的极性。通信开始后到通信结束完全由硬件执行，完成整个PWM的发送。

*5.2.3*通信结束

DMA完成最后一组数据搬移后，产生DMA completed中断，同时PWM也完成了发送，利用DMA的中断服务函数中进行发送完成的处理。

* 1. 通信流程

其流程如下图所示：

No

模块初期化

取得通信内容

数据重构

配置DMA

PWM Start

DMA Transfer completed？

DMA Interrupt

End

Yes

硬件动作

* 1. 结构关系

### 文件结构

完全符合Autosar协议。

### 分层结构

* 1. 接口设计

### 外部接口

提供给上层调用的接口有：

|  |  |  |
| --- | --- | --- |
| 接口定义 | 提供模块 | 接口说明 |
| LFDrv\_Init | LFDriver | 初始化LFDriver用的Clock，GPIO以及TAU模块。 |
| LFDrv\_Deinit | LFDriver | 反初始化LFDriver用的Clock，GPIO以及TAU模块。 |
| LFDrv\_SetDataBuffer | LFDriver | 获取通信内容，并将数据重构 |
| LFDrv\_ClearDataBuffer | LFDriver | 清除Buffer中的内容 |
| LFDrv\_RestCmd | LFDriver | 清零Buffer的Index |
| LFDrv\_AsyncTrsmStart | LFDriver | 开始通信 |
| LFDrv\_AsyncTrsmStop | LFDriver | 停止通信 |
| LFDrv\_GetStatus | LFDriver | 获取通信状态 |
| LFDrv\_DMACmpInterrupt | LFDriver | DMA completed中断服务函数 |

### 内部接口

内部接口主要用于模块内部使用：

|  |  |  |
| --- | --- | --- |
| 接口定义 | 使用函数 | 接口说明 |
| LFDrv\_ClkInit | LFDrv\_Init | 初始化clock设定 |
| LFDrv\_IOInit | LFDrv\_Init | 初始化GPIO设定 |
| LFDrv\_PwmConfig\_Init | LFDrv\_Init | 初始化PWM设定 |
| LFDrv\_ClkDeinit | LFDrv\_Deinit | 反初始化clock设定 |
| LFDrv\_IODeinit | LFDrv\_Deinit | 反初始化GPIO设定 |
| LFDrv\_PwmConfig\_Deinit | LFDrv\_Deinit | 反初始化PWM设定 |
| LFDrv\_DmaConfig | LFDrv\_SetDataBuffer | 配置DMA用于传送PWM的周期占空比以及极性的数据 |

# 详细设计

* 1. 算法设计
  2. 函数设计

按照AUTOSAR API标准设计。

* 1. 详细列表信息
     1. 数据列表

*Internal Data*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No.*** | ***Variable Name*** | ***Event Name*** | ***Type*** | ***Value Range*** | ***Storage*** | ***Description*** |
| *1* | *PwmDataBuffer[22]* | *-* | *uint8* | *[0x00,0xFF]* | *Ram* | *array to*  *store communication data* |
| *2* | *PwmDataBufferIdx* | *-* | *uint8* | *[0x00,0xFF]* | *Ram* | *variable to store the Pwm Data Buffer’s*  *Index* |
| *3* | *PwmDataBitIdx* | *-* | *uint8* | *[0x00,0xFF]* | *Ram* | *variable to store the Pwm Data Bit’s*  *Index* |
| *4* | *PwmDutyBuffer[180]* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *array to store the Pwm Duty* |
| *5* | *PwmDutyBufferIdx* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *variable to store the Pwm Duty Buffer’s*  *Index* |
| *6* | *PwmPeriodBuffer[180]* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *array to store the Pwm Period* |
| *7* | *PwmPeriodBufferIdx* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *variable to store the Pwm Period Buffer’s*  *Index* |
| *8* | *PwmTolBuffer[180]* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *array to store the Pwm TOL* |
| *9* | *PwmTolBufferIdx* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *variable to store the Pwm TOL Buffer’s*  *Index* |
| *10* | *PwmRdtBuffer[2]* | *-* | *uint16* | *[0x0000,0xFFFF]* | *Ram* | *array to store the Pwm RDT* |
| *11* | *PwmTrsmState* | *-* | *uint8* | *[0x00,0xFF]* | *Ram* | *variable to store the communication state* |

* + 1. 函数列表

**手动填充代码实现功能:**

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_Init | |
| Syntax | void LFDrv\_Init (void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 初始化LFDriver模块。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_Deinit | |
| Syntax | void LFDrv\_Deinit (void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 反初期化LFDriver模块。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_SetDataBuffer | |
| Syntax | boolean LFDrv\_SetDataBuffer(uint8 \*databuff,uint8 Siz) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | \*databuff | Siz |
| Parameters (out) | None | None |
| Return value | rv | rv |
| Description | 取得通信内容，并重构数据，配置DMA。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_ClearDataBuffer | |
| Syntax | void LFDrv\_ClearDataBuffer(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 清空data buffer。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_RestCmd | |
| Syntax | void LFDrv\_RestCmd(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 清零各个buffer的Index。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_AsyncTrsmStart | |
| Syntax | void LFDrv\_AsyncTrsmStart(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 启动counter，开始通信。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_AsyncTrsmStop | |
| Syntax | void LFDrv\_AsyncTrsmStop(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | 关闭counter，停止通信。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_GetStatus | |
| Syntax | LFDrvStatusReturnType LFDrv\_GetStatus(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | LFDrvRetState | LFDrvRetState |
| Description | 取得通信状态。 | |
| Call By |  | |

|  |  |  |
| --- | --- | --- |
| ID | N/A | |
| Name | LFDrv\_DMACmpInterrupt | |
| Syntax | void LFDrv\_DMACmpInterrupt(void) | |
| Sync/Async | Synchronous | |
| Reentrancy | Non Reentrant | |
| Parameters (in) | None | None |
| Parameters (out) | None | None |
| Return value | None | None |
| Description | DMA completed 中断服务函数，通信完成回调。 | |
| Call By |  | |

# 附录 1