LED Processor 研究

SWD 聂勇

版 本 历 史

版本/状态	责任人	起止日期	备注
V1.0/正式	聂勇	10May2011	建立 LED Processor 研究文档
V1.1/正式	聂勇	12May2011	修改文档中一些纰漏

目 录

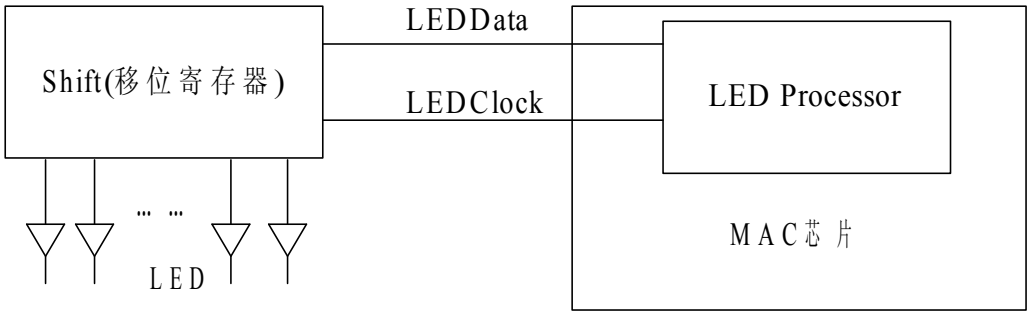
1. 说明

交换机端口的 LED 指示灯有两种驱动方式，一是通过 PHY 芯片来驱动，二是通过 MAC 芯片来驱动。本文档的 LED Processor 就是 Broadcom 公司通过 MAC 芯片驱动端口 LED 指示灯的具体实现。

2. LED 串行接口

LED Processor 是如何来驱动 LED 指示灯的呢？首先我们从硬件连接来看如何实现的，这就涉及到 LED 串行接口。如下图 2- 1 LED Processor 串行接口 所示，LED Processor 的输出为两个 pin 引脚，一跟为数据信号线，叫做 LEDData，另一根为时钟信号线，叫做 LEDClock。

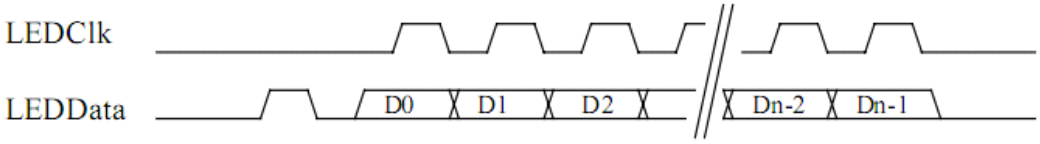
移位寄存器为外部器件，目的就是锁存 LEDData 信号，驱动 LED 灯发光。如果你对移位寄存器的工作原来不清楚，请首先阅读数字设计方面的知识。



2- 1 LED Processor 串行接口

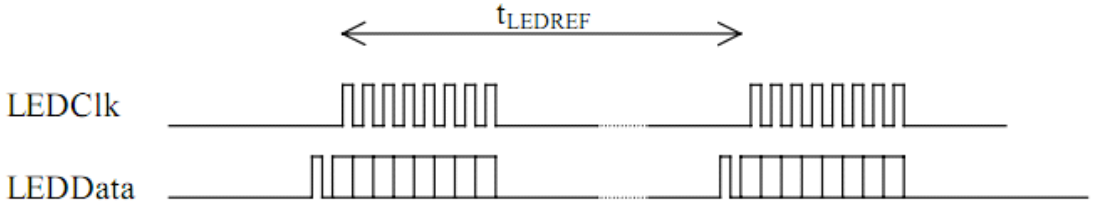
下面，我们就来介绍 LEDData 和 LEDClock 的时序关系。在下图 2- 2 LEDData/LEDClock 相位关系中，一共有 N 位数据被锁存在移位寄存器中，那么对应的 N 个 LED 获得了数据，将被点亮或者熄灭。

默认情况下，LEDClock 的频率为 5MHz，也就是 200ns。如果一共有 48 个 LED，那么从 D0 到 D47 全部移位到对应 LED 的 pin 引脚上，只需要 9.6us 的时间。这个时间，对于 LED 灯的亮灭没有影响，可以忽略。



2- 2 LEDData/LEDClock 相位关系

LED Processor 会定时更新一次移位寄存器上的所有数据，如下图 2- 3 LED Processor 更新操作所示。默认情况下，跟新的频率为 30Hz，也就是 33ms。



2- 3 LED Processor 更新操作

LED 串行接口，只是 LED Processor 的外部表现。上图表述的就是 LED Processor 会每隔 33ms 输出一串时钟和数据，持续时间大概为 9.6us（都是在默认配置下，48 个 LED）。

3. LED Processor

本节要解决的问题，LED Processor 是如何获得 LEDData 信号线上的数据，如何将 LEDData 和端口的状态相联系等等。这些，都是 LED Processor 作为一个 8bit 的处理器需要解决的问题。

3.1 LED Processor 与 Host Processor

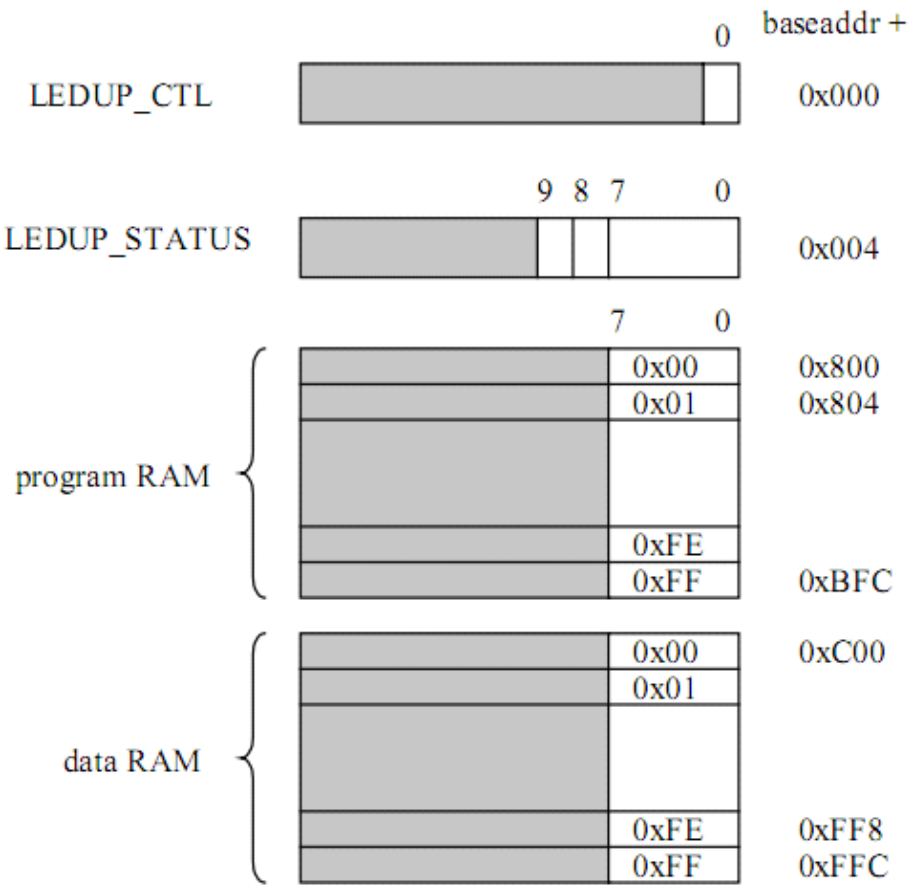
Host Processor 指 MAC 芯片中集成的主控制 CPU。例如，Hawkeye 方案的 5331xS 系列，在 MAC 芯片中都是集成了一个 MIPS 架构的 CPU，集成的 CPU 叫做 ICS（Internal CPU Subsystem in Switch）。这本文档中，和 LED Processor 对比，就是一个主从机的关系，所以我们把它叫做 Host Processor。

LED Processor 是一个集成在 CMIC（CPU Management Interface Controller）中的简单，微型的 8 位处理器。它十分适合完成控制 LED 的任务。LED Processor 和 Host Processor 内存空间的交互，是通过直接 PCI 内存映射完成。

下图 3- 1 LED Processor 和 Host Processor 的内存关系就是通过 Host Processor 看到的 LED Processor 的空间分布。

例如，在 5331x 系列的寄存器手册 Section 10: CMIC LED 章节中就有如下寄存器和地址空间描述：CMIC_LEDUP_CTRL 寄存器，CMIC_LEDUP_STATUS 寄存器，CMIC_LEDUP_PROGRAM_RAM 空间段，CMIC_LEDUP_DATA_RAM 空间段。这些其实就是 LED Processor 的寄存器，内存等在 Host Processor 中的映射。

那么，LED Processor 的寄存器，内存到底是怎么样的呢，请看下一节 LED Processor 处理器架构。



3- 1 LED Processor 和 Host Processor 的内存关系

3.2 LED Processor 处理器架构

LED Processor 是一个典型的 8 位处理器。寄存器，地址空间，PC 指针等的长度都是 8 位。处理器中集成了两片独立的 256 字节的内存，一片用于存放程序，叫做 Program Space，另一篇用于存放数据，叫做 Data Space。

Data Space 中的数据，LED Processor 和 Host Processor 都能够读写；Program Space 中的数据，LED Processor 只能够从中取指令，不能够读写其中的内容，而 Host Processor 就能够读写该内存空间的数据。

3.2.1 Data Space

Data Space 中的一部分内存空间被用于记录了每个端口的状态以及要传送到 LEDData 引脚上的串行信号。下面，我们来了解 Data Space 的使用情况。

Address range	Use
0x00-0x01	status bits for port 0
0x02-0x03	status bits for port 1
…	…
0x3E-0x3F	status bits for port 31
0x40-0x5F	LED scan chain assembly area
0x60-0xFF	undesignated

3- 2 Data RAM 功能分布图

如上图 3- 2 Data RAM 功能分布图所示，0x00~0x3F 的 64 字节空间被依次分配给了端口 0~31，其中，每个端口使用 2 字节的空间来表示该端口的状态。这 2 个字节每位的含义，如下图 3- 3 端口状态的比特位含义所示。

Bit	Meaning	Status=0	Status=1
0	RX	no frames received	frames received
1	TX	no frames transmitted	frames transmitted
2	Collision	no collisions	collisions have occurred
[4:3]	Speed	00=10 Mbps, 01=100 Mbps, 10=1000 Mbps	
5	Duplex	half duplex	full duplex
6	Flow Control	no pause handshake	pause handshake successful
7	Link Up	link down	link up
8	Link Enabled	link disabled	link enabled
9-13	unused		
14	False	always 0	
15	True	always 1	

3- 3 端口状态的比特位含义

LED scan chain 就是存放将要传送到 LEDData 引脚上的串行信号。一共 32 字节，256 个比特位。也就是说，最多一共可以点亮 256 个 LED 灯。

3.2.2 寄存器集/地址模式/子程序

了解本节的知识，你需要有一些处理器架构方面的知识和经验。

首先是 LED Processor 的寄存器集（Register Set）。寄存器集包括了寄存器，状态位，以及硬件堆栈，如下所述。

- 2 个 8 位的寄存器，A 和 B。这两个寄存器就是作为处理器的累加器使用。
- 2 个状态位，C 位和 Z 位。C 含义是 carry，当 ALU 的运算溢出的时候，C 位就会被置 1；Z 含义是 zero，当 ALU 的运算结果为 0 的时候，Z 位就会被置 1。
- 1 个深度为 4，宽度为 1 比特的 T 堆栈。T 堆栈被用来建立起串行 LED 数据流。

下面介绍 LED Processor 的地址模式。由于缺少指令位来编码所有的地址模式（一条指令只有 8 位，用来编码的指令位就更少了，这是所有 8 位处理器都存在的问题），所以并不是所有的指令都支持所有的地址模式。

Address Mode	Nomenclature examples	The instruction refers to:
Register	`A` `B`	the contents of the `A` or `B` register

Immediate	`3`
	`0x21`
	`label`
	`+(3<<2)+1`
		an immediate 8b constant value

Indirect	`(A)` `(B)`	the data RAM location pointed to by the contents of the `A` or `B`

Absolute	`(3)`
	`(0x21)`
	`(label)`
	`((3<<2)+1)`
		the data RAM location pointed to by the immediate 8b constant value

3- 4 地址模式

Example	Explanation
LD A,B	The contents of the B register are copied to the A register
LD A,3	The immediate value 3 is copied to the A register
LD B,(0x21)	Data RAM location 0x21 is copied to the B register
LD (0x21),B	Data RAM location 0x21 is written with the value of the B register
LD A,(phase)	The symbolic label "phase" represents an 8b constant that selects which data RAM location supplies the new value of the A register
LD (A),077	The data RAM location indexed by the contents of register A is overwritten with the 8b octal constant 077 (63 decimal).
JMP 0x30	The program continues fetching instructions from address 0x30.

3- 5 地址模式示例

最后，我们介绍一下 LED Processor 的子程序调用，也就是跳转和返回问题。和真正的微处理器不一样，LED Processor 没有实现数据内存中的调用/返回堆栈。取而代之的是一个保存在寄存器中的深度为 2 的返回地址堆栈。

堆栈的深度为 2，如果有多余 2 条 CALL 指令执行而没有 RET，硬件并不会有异常检测，这就要求程序员非常注意编程的时候不要出现多余 2 条的 CALL 指令而没有 RET。

3.2.3 LED Processor 指令集

看完本节，你就可以阅读和编写 LED Processor 的汇编代码了。

LED Processor 的指令一共有三类，如下表所示。

类别	说明	示例
typical uP data operations	微处理器数据操作指令	LD、XOR、DEC……
branching operations	分支操作指令	J、JMP、CALL、RET
LED-specific operations	LED 状态操作指令	PORT、PUSHST、TOR……

3- 6 LED Processor 指令分类

typical uP data operations 和 branching operations 就是大部分微处理器都有的指令，这里不做介绍。下面详细的介绍 LED-specific operations。LED-specific operations 就是专门为实现 LED Processor 的功能而设计的指令。

PORT 指令：指定 0~31 哪个端口的数据要被操作。所谓端口的数据是指位于 DATA Space 的前 64 字节的数据，每个端口对应 2 个字节 16 比特。

PUSHST 指令：将某个端口的 2 字节 16 比特数据中的某一位送入 T 堆栈中。

PUSH 指令：将参数的最低位送入 T 堆栈中。

POP 指令：将 T 堆栈栈顶的一个比特出栈，放入 C 位中。

TAND、TOR、TXOR、INV 指令：TAND、TOR、TXOR 指令是将 T 堆栈的栈顶两个比特进行相应的运算，然后将结果再存放到 T 堆栈中。INV 指令是将 T 堆栈的栈顶一个比特取反，结果存放到 T 堆栈中。

PACK 指令：将 T 堆栈中的数据送入 LED scan chain 中。

SEND 指令：说明 LED scan chain 的长度为对少 bit。然后 LED Processor 就会将 LED scan chain 中规定长度的数据发送出去。LED scan chain 位于 DATA Space 的 0X40~0X5F 处，一共 256 个比特。

4. LED Processor 工具

Broadcom 的 SDK 中提供了进行 LED Processor 程序开发的基本工具。包括一个汇编器，一个反汇编器，一个仿真器。

需要强调的是，SDK 中提供的是用 C 编写的源文件。你需要针对你的当前的操作系统编译成相应的可执行文件，然后才能够运行这些工具。

4.1 SDK 中添加定制的 LED Processor 程序

在获得了编译好的 LED Processor 程序之后，如果运行调试，如何添加到 SDK 中呢？首先，SDK 提供了几个命令用于控制 LED Processor。我们可以使用这些命令来测试我们的 LED Processor。

led stop

led load

led start

那么，如何将 LED Processor 程序添加到 SDK 中呢？你只需要将你定制的程序添加到文件$SDK/rc/rc.soc 对应的机型下面就可以。另外，你也可以在自己的 APP 程序中添加，SDK 提供了 soc_ledproc_config()函数接口。

有关 DEMO 程序，请参考$SDK/src/appl/diag/ledproc.c 文件。