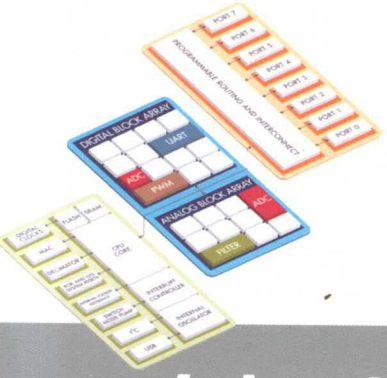
THE CYPRESS UNIVERSITY ALLIANCE PROGRAM

Cypress PSoC设计指南系列

CD-ROM
配光盘
(源程序和课件)



混合信号嵌入式设计实验指南

Laboratory Manual for Introduction to Mixed-Signal, Embedded Design

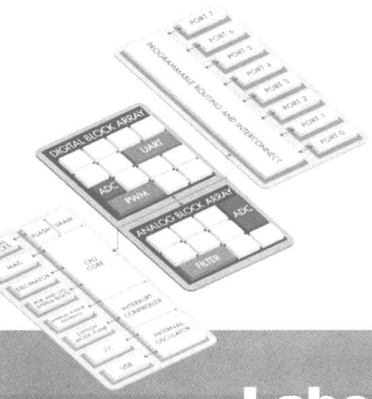
[美] 戴维凡·埃斯（David Van Ess）
[美] 爱德华 H. 柯里（Edward H. Currie） 著
[美] 亚历克斯 N. 杜博里（Alex N. Doboli）
何 宾 译

化学工业出版社

THE CYPRESS

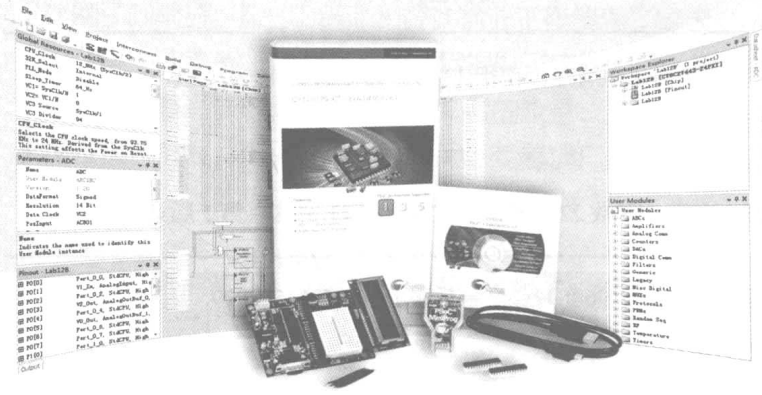
Cypress PS

混合信号嵌入式设计实验指南



Laboratory Manual for Introduction to Mixed-Signal, Embedded Design

[美] 戴维凡·埃斯（David Van Ess）
[美] 爱德华 H. 柯里（Edward H. Currie） 著
[美] 亚历克斯 N. 杜博里（Alex N. Doboli）
何 宾 译



CD-ROM
配光盘
(源程序和课件)

化学工业出版社
·北 京·

图书在版编目（CIP）数据

混合信号嵌入式设计实验指南 / [美]埃斯（Ess，D. V.），[美]柯里（Currie，E.H.），[美]杜博里（Doboli，A.N.）著；何宾译. —北京：化学工业出版社，2012.9

书名原文：Laboratory Manual for Introduction to Mixed-Signal, Embedded Design

ISBN 978-7-122-14958-9

Ⅰ. ①混… Ⅱ. ①埃… ②柯… ③杜… ④何… Ⅲ. ①混合信号-程序设计-指南 Ⅳ. ①TN911.7-62

中国版本图书馆 CIP 数据核字（2012）第 170458 号

Laboratory Manual for Introduction to Mixed-Signal, Embedded Design/by David Van Ess, Edward H. Currie and Alex N. Doboli

ISBN 978-0-9814679-1-7

北京市版权局著作权合同登记号：01-2012-4940

责任编辑：宋 辉　　文字编辑：云 雷

责任校对：陈 静　　装帧设计：韩 飞

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号 邮政编码 100011）

印　　刷：北京永鑫印刷有限责任公司

装　　订：三河市万龙印装有限公司

787mm×1092mm 1/16 印张 10½ 字数 220 千字 2012 年 11 月北京第 1 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899

网　　址：http: // www. cip. com. cn

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：48.00 元

译者序

本书虽然是一本基于 Cypress 公司 PSoC1 平台的实验指导，但是该书的独特之处，正如作者所说的那样，将一个最终的设计目标——带温度补偿的风扇控制器，分解成一个个具体的模块。在一步步实现这些具体模块的基础上，最终实现设计目标。本书另一个独特之处在于，在介绍各个模块的实现过程时，给出了一个个有趣的问题。通过解决这些问题，掌握 PSoC1 的整体结构和各个功能的模块。所以，当我拿到 Cypess 公司所提供的本书外文版时，非常兴奋。因为该书也给国内高等学校工程人才的培养提供了一些新的思路，即如何把理论和实践教学相结合，切实提高学生解决实际工程问题的能力。

本书在翻译过程中做了一些验证工作，原因是：一方面，该书仅提供了实验教学素材和实现的流程，但并没有给出具体的实现代码；另一方面 Cypress 公司 PSoC1 的软件设计平台 PSoC Designer 不断升级，目前已经升级到了 5.2 版本。综合上述原因，需要对书上提供的实验素材全部进行设计、编程和验证，以适应 PSoC1 软件和硬件平台的最新发展。为了给国内相关的使用者提供一个更好的学习资料，本书特别将实验素材的验证程序全部放入光盘赠送给读者。

本书的实验内容虽然是基于 Cypress 公司的 PSoC1 平台，但是这些实验素材同样可以应用于 Cypress 公司的 PSoC3/5 的平台上。

本书由何宾翻译，彭勃为数字实验部分内容的翻译提供了帮助，常晓磊为模拟实验部分内容的翻译提供了帮助。北京交通大学电子信息工程学院的研究生杨光伟负责本书所有设计例子的验证。本书的翻译和设计验证工作得到了美国 Cypress 公司中国区大学计划经理魏荣博士的支持，她为本书的编写提供了软件和硬件平台的支持，在此向她表示感谢。

译者在翻译该书时，力图遵循作者的原意，并对书中的一些错误进行了改正，但是由于译者水平有限，难免会有不足之处，也恳请广大读者批评指正。

译者

前 言

混合信号嵌入式设计实验指南

本实验手册作为教科书《混合信号及嵌入式设计导论》(作者：Doboli 和 Currie）的配套书籍。该书中的例子是基于 Cypress 半导体公司混合信号阵列器件，这个混合信号阵列器件集成了片上控制器，将这个混合阵列称为可编程片上系统[Programmable System on(a) Chips, PSoC]。每一个 PSoC 都包含一个 8 位微控制器、高可配置性的模拟和数字外设。

本手册分为 12 个章节，前 6 节集中在数字电路，后 6 节集中在模拟电路。每一章节的设计使你熟悉一个特殊的概念。但是，每个章节是实现最终目标的基础。这个最终目标是指实现一个温度补偿的风扇控制器。这是对基于 PSoC 的混合信号及嵌入式设计功率方面的一个说明性的例子。该实验要求最少的设备，即假设有可用的 CY3210 PSOCEvall 评估板。

选择这个特殊的例子，有以下几个原因：

- 结合了模拟、数字和软件设计。
- 介绍一些基本的控制理论概念。
- 需要使用几种不同类型的串行数字通信。

PSoC 最吸引人的特征之一，是它具有可以实时动态地可重配置模块的能力。即，根据不同的目的，在不同的时间，使用相同的模块通过 PSoC 的数字和模拟资源，来提供最佳的利用。

使用相同的资源来做更多的事情，这样就允许设计者创建更多的应用，这些应用更加廉价、功耗更低、占用 PCB 资源更少，以及占用更小的物理空间。

作为一种教学工具，PSoC 提供了一种包含全套开发工具（汇编器，链接器，调试器和编程器）的完整混合信号平台。

目 录

混合信号嵌入式设计实验指南

1 数字实验

1.1 实验 1—CPU 和通用 I/O

目标：本实验介绍 PSoC 微控制器（CPU）以及和 I/O 端口的接口。注意：本实验分成三个部分：实验 1A，实验 1B 和实验 1C。在进行下一个实验前需要完成前一个实验。如果成功地完成了本实验，将实现下面的目标：

- 能使用 PSoC Designer 创建一个新的工程；
- 了解 CPU 的基本操作和功能；
- 了解影响 CPU 性能的系统参数；
- 可以在汇编语言中声明全局变量；
- 计算一个 CPU 控制循环的周期时间；
- 了解如何读和写一个 I/O 端口；
- 配置 I/O 引脚用于不同的驱动模式；
- 使用影子寄存器隔离输入和输出之间的相互作用；
- 使配置的 I/O 端口读瞬间的按键状态；
- 写一个去抖动例程用于正确地读这个开关。

假设：

- 在计算机上已经安装了 PSoC Designer 和 PSoC Programmer。

要求的材料：

- CY3210 PSoCEval1 板；
- 面包板线。

要求的仪器：

- 示波器。

相关的参考资料：

- Cypress Technical Reference Manual；
- Cypress Assembly Language User Guide；
- Cypress Tele-Training Video Module 1：Introductory Module；
- Cypress Tele-Training Video Module 2：Getting Started Designing。

1.1.1 实验 1A—CPU

步骤 1：开始一个新的工程。

- 打开 PSoC Designer，开始一个新的工程。
- 选择“base part”为 CY8C27443-24PXI，工程名字为“Lab1A”（除非明确地指出来，所有的工程都将使用 CY8C27443-24PXI 作为 base part）。

- 选择“Assembler”选项。
- 转入器件编辑器界面，切换到如图 1.1 所示的互联视图中。

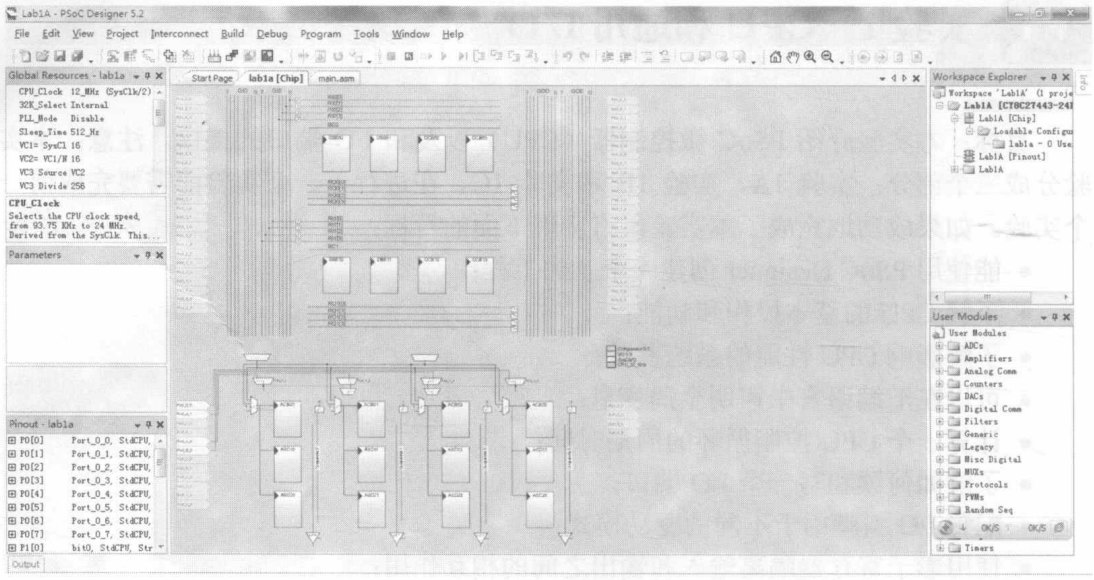


图 1.1 PSoC Designer 的互联视图

步骤 2：设置必要的全局参数。

- 全局资源窗口（Global Resources）位于 PSoC Designer 窗口的左上角，如图 1.2 所示，该窗口包含了 20 个参数，但只有 CPU_Clock 和 Supply Voltage 影响 CPU 的操作。

Global Resources - lab1a

CPU_Clock	3_MHz (SysClk/8)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	256
SysClk Source	Internal 24_MHz
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD	4.81V (5.00V)
LVDThrottleBack	Disable
Supply Voltage	5.0V
Watchdog Enable	Disable

图 1.2 PSoC Designer 全局资源窗口

• CPU_Clock 参数中有 8 个选项。当供电电压保证大于 4.75V 的时候，只使用 SysClk（24MHz）。

• Supply Voltage 中有两个电压设置。如图 1.3 所示，对于 3.3V 来说，可允许设置的最快 CPU 时钟频率是 SysClk/2(12MHz)。

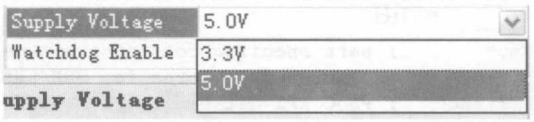


图 1.3 PSoC Designer 供电电压子菜单

除非有其他描述，后面所有的实验工程将这些参数设置为如图 1.3 和图 1.4 所示的 5.0V 和 12MHz。

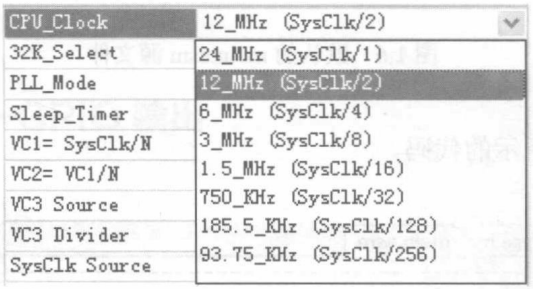


图 1.4 PSoC Designer 全局资源时钟子菜单

步骤 3：设置端口 1（Port 1）引脚的驱动模式。

• PSoC Designer 窗口的左下角是端口选择窗口。在该窗口中，将端口 1 引脚设置为强驱动；

• 如图 1.5 所示，重命名端口引脚；

• 生成应用（当改变互联视图中的任何东西，都将重新生成应用）。

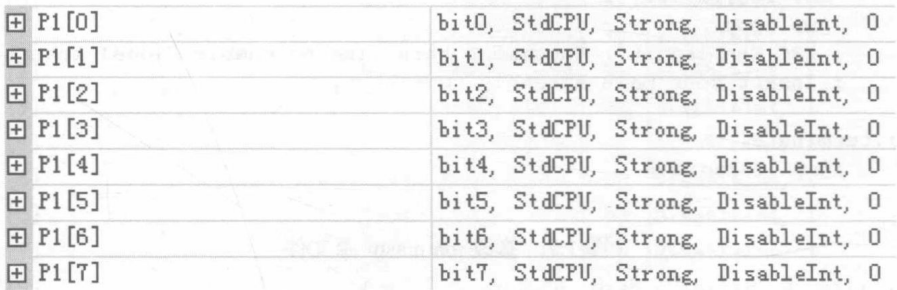


图 1.5 端口驱动

步骤 4：写软件。

- 转到应用程序编辑器，打开如图 1.6 所示的 main.asm 文件。

Start Page | lab1a [Chip] | ctype.h | main.asm

```
;-----------------------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------------------

include "m8c.inc"        ; part specific constants and macros
include "memory.inc"     ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"    ; PSoC API definitions for all User Modules

export _main

_main:

    ; M8C_EnableGInt ; Uncomment this line to enable Global Interrupts
    ; Insert your main assembly code here.

.terminate:
    jmp .terminate
```

图 1.6 默认的 main.asm 源文件

- 添加如图 1.7 所示的代码。

Start Page | lab1a [Chip] | ctype.h | main.asm

```
;-----------------------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------------------

include "m8c.inc"        ; part specific constants and macros
include "memory.inc"     ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"    ; PSoC API definitions for all User Modules

export _main

_main:
loop:
    mov A,reg[PRT1DR]
    inc A
    mov reg[PRT1DR],A
    jmp loop
    ; M8C_EnableGInt ; Uncomment this line to enable Global Interrupts
    ; Insert your main assembly code here.

;.terminate:
    ;jmp .terminate
```

图 1.7 修改 main.asm 源文件

练习 1A-1：这些代码的作用是什么？

练习 1A-2：循环的周期时间是多少（CPU 周期）？

练习 1A-3：对于一个 12MHz 的 CPU 时钟，每一个引脚的输出频率是多少？

Bit0=_____ Bit1=_____

Bit2=_____ Bit3=_____

Bit4=_____ Bit5=_____

Bit6=_____ Bit7=_____

步骤 5：下载代码并运行。

- 建立这个应用，验证代码没有错误；
- 下载程序到 Eval1 板子，并且运行；使用示波器察看每个引脚上的波形。

练习 1A-4：这些测量和计算结果是否一致？

- 返回互联视图，将 CPU 的时钟改为 3MHz；
- 重新生成应用，重新建立工程，下载到板子，并且运行这个应用。

练习 1A-5：当前的 CPU 时钟设置为 3MHz 时，引脚的信号应该如何变化？

练习 1A-6：示波器测量验证了该变化结果吗？

1.1.2 实验 1B—GPIO 输出

步骤 1：复制 Lab1A。

- 打开工程 Lab1A。
- 转到 File 菜单，选择 Save Workspace As...
- 将 New Project Name 设置为 Lab1B。

步骤 2：设置端口 1 的驱动模式。

- 端口选择窗口在 PSoC Designer 屏幕的左下角。如图 1.8 所示，将端口 1 引脚的驱动模式从 Strong 改成 Pull Up。

⊞ P0[5]	Port_0_5, StdCPU, High Z Analog, DisableInt, 0
⊞ P0[6]	Port_0_6, StdCPU, High Z Analog, DisableInt, 0
⊞ P0[7]	Port_0_7, StdCPU, High Z Analog, DisableInt, 0
⊞ P1[0]	bit0, StdCPU, Pull Up, DisableInt, 1
⊞ P1[1]	bit1, StdCPU, Pull Up, DisableInt, 1
⊞ P1[2]	bit2, StdCPU, Pull Up, DisableInt, 1
⊞ P1[3]	bit3, StdCPU, Pull Up, DisableInt, 1
⊞ P1[4]	bit4, StdCPU, Pull Up, DisableInt, 1
⊞ P1[5]	bit5, StdCPU, Pull Up, DisableInt, 1
⊞ P1[6]	bit6, StdCPU, Pull Up, DisableInt, 1
⊞ P1[7]	bit7, StdCPU, Pull Up, DisableInt, 1
⊞ P2[0]	Port_2_0, StdCPU, High Z Analog, DisableInt, 0

图 1.8 PSoC Designer 端口选择窗口

写一个端口使得对端口寄存器进行设置。引脚上的逻辑电平取决于驱动寄存器的值，驱动模式和连接到引脚的外部电路。读端口可以从引脚直接取出数据。当配置成 Pull Up 时，输出有一个到逻辑低的强阻抗驱动，以及到逻辑高的电阻驱动。当这些引脚驱动为高时，能通过外部电路拉低。

步骤 3：在 Eval1 板子上增加必要的跳线。

- 使用跳线将板子上的 Bit0 和 Bit6 短接在一起。这两个引脚是“线与”关系。当任何一个引脚驱动为低时，这两个引脚都将被拉低。

练习 1B-1：给出信息，即每个引脚上的波形是什么？

步骤 4：下载代码并运行。

- 设置 CPU_Clock 全局参数为 12MHz。
- 重新生成应用，重新建立工程，下载到板子上，并且运行。
- 使用示波器察看每个引脚的波形。

练习 1B-2：观察到的和预测的结果是否一致？

步骤 5：使用影子寄存器。

- 转到应用程序编辑器，打开如图 1.9 所示的 main.asm 文件。

main.asm

```
;-------------------------------
; Lab1A Control Software
;-------------------------------

include "m8c.inc"        ; part specific constants and macros
include "memory.inc"     ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"    ; PSoC API definitions for all User Modules

export _main

_main:
loop:
   mov   A,reg[PRT1DR]
   inc   A
   mov   reg[PRT1DR],A
   jmp loop
```

图 1.9 main.asm 源文件

- 添加如图 1.10 所示的代码。

需要注意：

- 这里添加了一个代码，用于为变量分配一个字节的 RAM。这个特殊的变量称为 bShadow。
- 通过 Export 描述，使得 bShadow 作为一个全局变量。这个变量可以被工程中其

他所有的文件使用。

Start Page | Lab1B [Chip] | main.asm* | Lab1B.lst

```
;-----------------------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------------------

include "m8c.inc"        ; part specific constants and macros
include "memory.inc"     ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"    ; PSoC API definitions for all User Modules


export bShadow
area bss(RAM)
   bShadow: BLK 1        ; Reserve a RAM location named bSHadow
area text(ROM,REL)

export _main

_main:
     mov[bShadow],0
loop:
     inc [bShadow]
     mov A,[bShadow]
     mov reg[PRT1DR],A
     jmp loop
    ; M8C_EnableGInt ; Uncomment this line to enable Global Interrupts
    ; Insert your main assembly code here.

;.terminate:
    ;jmp .terminate
```

图 1.10 修改源文件支持使用影子寄存器

- bShadow 的内容格式为[bShadow]。

现在就将输出寄存器的内容保存在变量中，这样就可以断开 I/O 输入/输出之间的相互作用。

练习 1B-3：现在已经断开了输入和输出之间的相互作用，每个输出的波形应该是什么样的？

练习 1B-4：循环的周期时间（CPU 周期）是多少？

步骤 6：下载和运行。

- 重新建立工程，下载设计到板子上，并且运行；
- 使用示波器察看 Bit6、Bit7 的触发器。

练习 1B-5：观察到的结果和预测的是否一样？

1.1.3 实验 1C—GPIO 输入

步骤 1：创建新的工程。