

Spansion® 模拟和微控制器产品



本文档包含有关 Spansion 模拟和微控制器产品的信息。尽管本文档内有原来开发该产品规格的公司名称“富士通”或“Fujitsu”，该产品将由 Spansion 提供给现有客户和新客户。

规格的延续

本文档内容并不因产品供应商的改变而有任何修改。文档内容的其他更新，均为改善文档而进行，并已记录在文档更改摘要。日后如有需要更改文档，其更改内容也将记录在文档更改摘要。

型号的延续

Spansion 将继续提供型号以“MB”开始的现有产品。如欲订购该类产品，敬请使用本文档内列出的产品型号。

查询更多信息

如欲查询更多关于 Spansion 存储器、模拟产品和微控制器产品及其解决方案的信息，请联系您当地的销售办事处。

书末出版说明

本文档介绍的产品，其设计、开发和制造均基于一般用途，包括但不限于普通工业使用、普通办公使用、个人使用及家庭使用，不应用于：(1) 存在严重风险或危险，除非能够保证极高的安全性，否则可能对公众造成严重影响，甚至可能直接造成死亡、人员伤害、物品损坏或其他损失的用途（如核设施的核反应控制、飞机飞行控制、空中交通控制、公共交通控制、医学生命支持系统、武器系统的导弹发射控制），或者(2) 不允许出现故障的用途（如潜艇中继器和人造卫星）。请注意，对于您和 / 或任何第三方由于将产品用于上述用途而造成的任何索赔和损失，Spansion 不承担任何责任。任何半导体设备都可能发生故障。您必须在自己的设施和装置中加入安全设计措施，如冗余、防火、防止电流过载及其他异常运行情形等，以防由于此类故障而造成伤害、损坏或损失。如果根据日本 Foreign Exchange and Foreign Trade Law、美国US Export Administration Regulations 或其他国家（地区）的适用法律的规定，本文档中介绍的任何产品是在出口方面受到特别限制的商品或技术，则这些产品的出口必须预先得到相关政府的许可。

商标和声明

本文档的内容如有变更，恕不另行通知。本文档可能包含Spansion 正在开发的 Spansion 产品的相关信息。Spansion 保留变更任何产品或停止其相关工作的权利，恕不另行通知。本文档中的信息“按原样”提供，对于其精确性、完整性、可操作性、对特定用途的适用性、适销性、不侵犯第三方权利等不提供任何担保或保证，也不提供任何明确的、隐含的或法定的其他担保。对于因使用本文档中的信息而造成的任何形式的任何损失，Spansion 不承担任何责任。

版权所有© 2013 Spansion Inc. 保留所有权利。Spansion®、Spansion 标识、MirrorBit®、MirrorBit® Eclipse™、ORNAND™ 以及它们的组合，是Spansion LLC 在美国和其他国家（地区）的商标和注册商标。使用的其他名称只是一般性参考信息，可能是其各自所有者的商标。

F²MC-8FX 家族

8 位微型控制器

MB95F430 系列

16 位 PPG 定时器

应用笔记



修改记录

版本	日期	作者	修改记录
1.0	2010-03-16	Folix	初稿

本手册包含30页。

1. 本文档记载的产品信息及规格说明如有变动，恕不预先通知。如需最新产品信息和/或规格说明，联系富士通销售代表或富士通授权经销商。
2. 基于本文档记载信息或示意图的使用引起的对著作权、工业产权或第三方的其他权利的侵害，富士通不承担任何责任。
3. 未经富士通明文批准，不得对本文档的记载内容进行转让、拷贝。
4. 本文档所介绍的产品并不旨在以下用途：需要极高可靠性的设备，诸如航空航天装置、海底中继器、核控制系统或维系生命的医用设施。
5. 本文档介绍的部分产品可能是“外汇及外贸管理法”规定的战略物资(或专门技术)，出口该产品或其中部分元件前，应根据该法获得正式批准。

版权©2009 富士通半导体(上海)有限公司

目录

修改记录	2
目录	3
1 概要	5
2 PPG概要	6
2.1 背景	6
2.2 PPG格式	6
3 16 位PPG定时器描述	7
3.1 16 位 PPG定时器的结构图	7
3.2 16 位PPG定时器的引脚	8
3.3 16 位PPG定时器的寄存器列表	8
3.4 16 位PPG向下计数器寄存器	8
3.5 16 位PPG周期设置缓冲寄存器	9
3.6 16 位PPG占空比设置缓冲寄存器	10
3.7 16 位PPG状态控制寄存器	11
3.8 16 位PPG触发源控制寄存器	13
3.9 PPG宏的一般设置程序	13
4 16 位PPG定时器的中断	14
4.1 16 位PPG定时器的中断控制位和中断源	14
4.2 与 16 位PPG定时器中断相关的寄存器和向量表地址	14
5 PPG触发器	15
6 PPG驱动器	18
6.1 外围设备的使用	18
6.2 驱动代码	18

6.2.1	一般定义.....	18
6.2.2	PPG程序	19
7	典型应用	21
7.1	HW设计	21
7.2	范例代码	21
8	更多信息	23
9	附录.....	24
10	代码范例	25

1 概要

本文档介绍了如何在 MB95F430 系列上使用 PPG 功能。

第二章介绍了相关背景以及 PPG 格式。

第三章介绍了 PPG 模块，PPG 寄存器，以及设置程序。

第四章介绍了 PPG 中断。

第五章介绍了 PPG 驱动器。

第六章介绍了 PPG 的应用演示。.

2 PPG 概要

本章简要介绍了 PPG。

2.1 背景

PPG 通常应用于通信，电子，自动化等领域。在通信领域中，占空比或频率可用于识别协议。在电子领域中，PPG 可用于设计开关式电源。在自动化领域中，它可以用于控制电动机。

2.2 PPG格式

PPG 格式包括频率和占空比，下图为 PPG 的波形图。

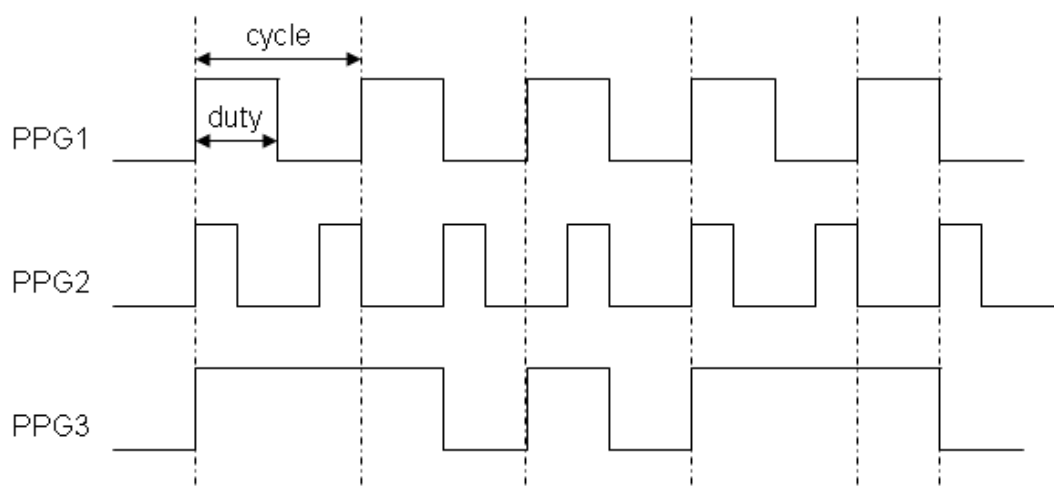


图 2-1: PPG 格式

3 16 位 PPG 定时器描述

16 位 PPG 定时器可输出 PWM 波和单脉冲波，并通过设置寄存器改变波形极性（正常极性↔反向极性）。

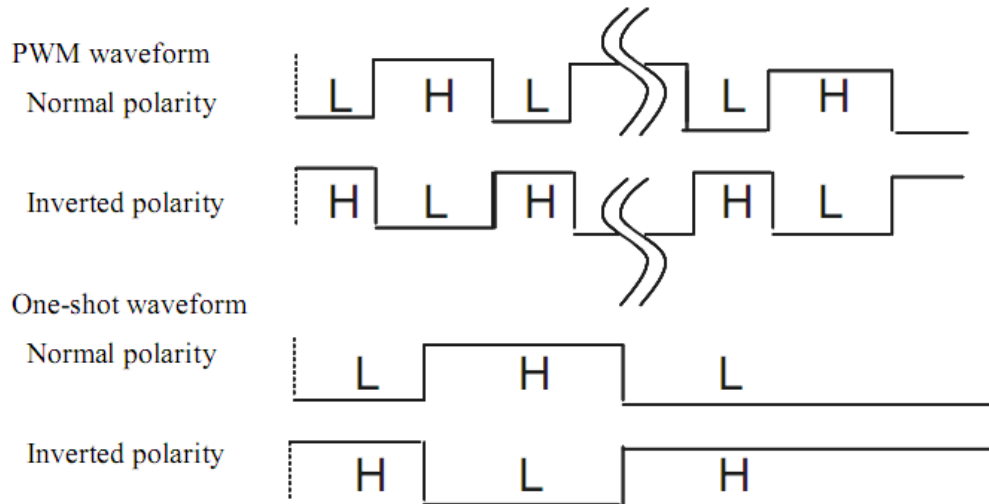


图 3-1: 16 位 PPG 定时器

3.1 16 位 PPG 定时器的结构图

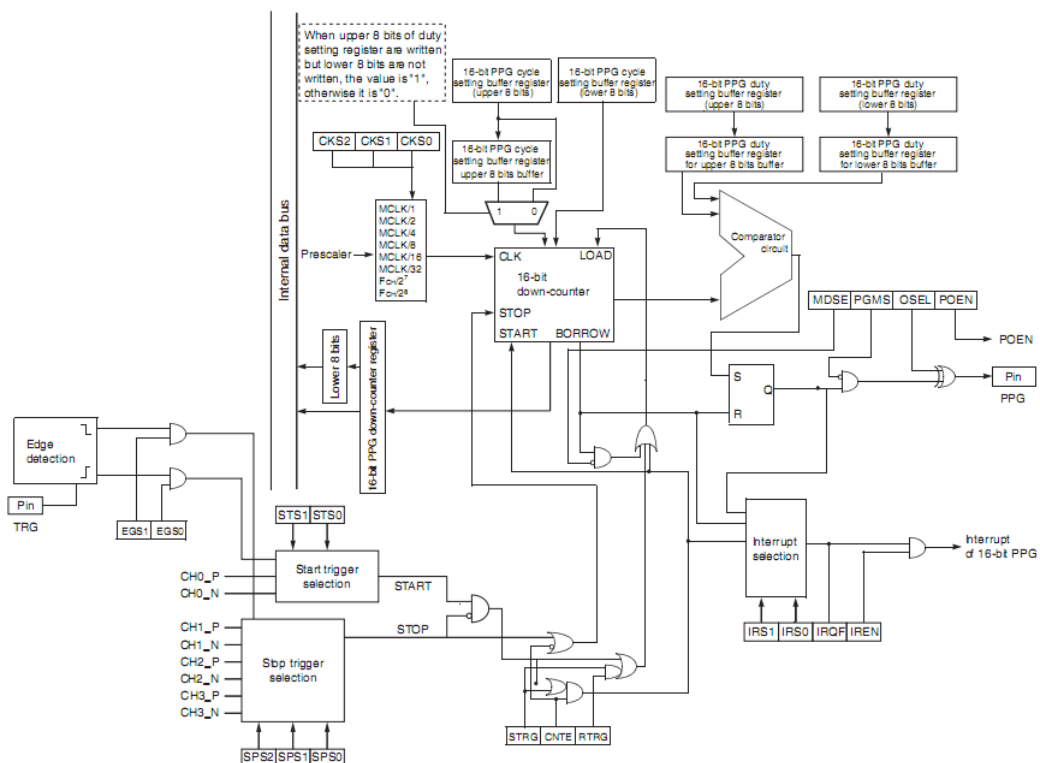


图 3-2: 16 位 PPG 定时器的结构图

3.2 16 位 PPG 定时器的引脚

与 16 位 PPG 定时器相关的引脚有 PPG0 引脚和 TRG0 引脚。

PPG0 为输出引脚。使用 16 位 PPG 状态控制寄存器启用输出（PCNTL0: POEN=1）可输出 PPG 波形。TRG0 可通过硬件触发器启动 16 位 PPG 定时器。

3.3 16 位 PPG 定时器的寄存器列表

寄存器	描述
PDCRH0	16-bit PPG down counter register (upper)
PDCRL0	16-bit PPG down counter register (lower)
PCSRH0	16-bit PPG cycle setting buffer register (upper)
PCSRL0	16-bit PPG cycle setting buffer register (lower)
PDUTH0	16-bit PPG duty setting buffer register (upper)
PDUTL0	16-bit PPG duty setting buffer register (lower)
PCNTH0	16-bit PPG status control register (upper)
PCNTL0	16-bit PPG status control register (lower)
PTGS0	16-bit PPG trigger source control register

3.4 16 位 PPG 向下计数器寄存器

16 位 PPG 向下计数器寄存器（高位，低位）（PDCRH0, PDCRL0）形成一个 16 位寄存器，用于从 16 位 PPG 向下计数器读取计数值。

寄存器的初始值全部为“0”。

请使用以下方法读取该寄存器。

使用 MOVW 指令（使用 16 位访问指令读取 PDCRH0 寄存器地址）。

使用 MOV 指令，先读取 PDCRH0，然后读取 PDCRL0（读取 PDCRH0 自动复制向下计数器的低 8 位至 PDCRL0）。

这些寄存器为只读，烧写操作无效。

16-bit PPG down counter register (upper) PDCRH0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAA _H PDCRH0	DC15	DC14	DC13	DC12	DC11	DC10	DC09	DC08	00000000 _B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

16-bit PPG down counter register (lower) PDCRL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAB _H PDCRL0	DC07	DC06	DC05	DC04	DC03	DC02	DC01	DC00	00000000 _B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

图 3-3: 16 位 PPG 向下计数器寄存器

3.5 16 位 PPG 周期设置缓冲寄存器

16 位 PPG 周期设置缓冲寄存器用于为 PPG 产生的输出脉冲设置周期。

这些寄存器形成一个 16 位寄存器，用于为 PPG 产生的输出脉冲设置周期。这些寄存器中值被加载至向下计数器。

请使用以下方法烧写数据至这些寄存器。

使用 MOVW 指令（使用 16 位访问指令烧写至 PCSRH0 寄存器地址）

使用 MOV 指令，先烧写至 PCSRH0，然后烧写至 PCSRL0。

烧写数据至 PCSRH0 后（烧写数据至 PCSRL0 前），如果向下计数器开始加载，之前有效的 PCSRH0/PCSRL0 值将被加载至向下计数器。如果在计数期间修改 PCSRH0/PCSRL0 值，修改值将在向下计数器下一次加载后生效。

不要设置 PCSRH0 和 PCSRL0 为 00H，设置 PCSRH0 为 01H，或者设置 PCSRL0 为 01H。

16-bit PPG cycle setting buffer register (upper) PCSRH0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0FAC _H PCSRH0	CS15	CS14	CS13	CS12	CS11	CS10	CS09	CS08	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

16-bit PPG cycle setting buffer register (lower) PCSRL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAD _H PCSRL0	CS07	CS06	CS05	CS04	CS03	CS02	CS01	CS00	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

图 3-4: 16 位 PPG 周期寄存器

3.6 16 位 PPG 占空比设置缓冲寄存器

16 位 PPG 占空比设置缓冲寄存器为 PPG 生成的输出脉冲控制占空比。

这些寄存器形成一个 16 位寄存器，用于为 PPG 生成的输出脉冲控制占空比。从 16 位 PPG 占空比设置缓冲寄存器到占空比设置寄存器的数据转移与向下计数器的读取同时执行。

请使用以下方法烧写数据至这些寄存器。

使用 MOVW 指令（使用 16 位访问指令烧写至 PDUTH0 寄存器地址）

使用 MOV 指令，先烧写至 PDUTH0，然后烧写至 PDUTL0。

烧写数据至 PDUTH0 后（烧写数据至 PDUTL0 前），如果向下计数器开始加载，16 位 PPG 占空比设置缓冲寄存器的值不会转移至占空比设置寄存器。

16 位 PPG 占空比设置寄存器的值与输出脉冲之间的关系如下。

16 位 PPG 周期设置缓冲寄存器和占空比设置寄存器的值相同时，如果极性正常，始终输出“H”；如果极性反向，始终输出“L”。

占空比设置寄存器设置为“00B”时，如果极性正常，始终输出“L”；如果极性反向，始终输出“H”。

占空比设置寄存器的值大于 16 位 PPG 周期设置缓冲寄存器的值时，如果极性正常，始终输出“L”；如果极性反向，始终输出“H”。

16-bit PPG duty setting buffer register (upper) PDUTH0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0FAE _H PDUTH0	DU15	DU14	DU13	DU12	DU11	DU10	DU09	DU08	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

16-bit PPG duty setting buffer register (lower) PDUTL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAF _H PDUTL0	DU07	DU06	DU05	DU04	DU03	DU02	DU01	DU00	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

图 3-5: 16 位 PPG 占空比寄存器

3.7 16 位 PPG 状态控制寄存器

16 位状态控制寄存器用于启用-禁用 16 位 PPG 定时器，并为软件触发器，再次触发控制中断以及输出极性设置操作状态。该寄存器还可以检查操作状态。

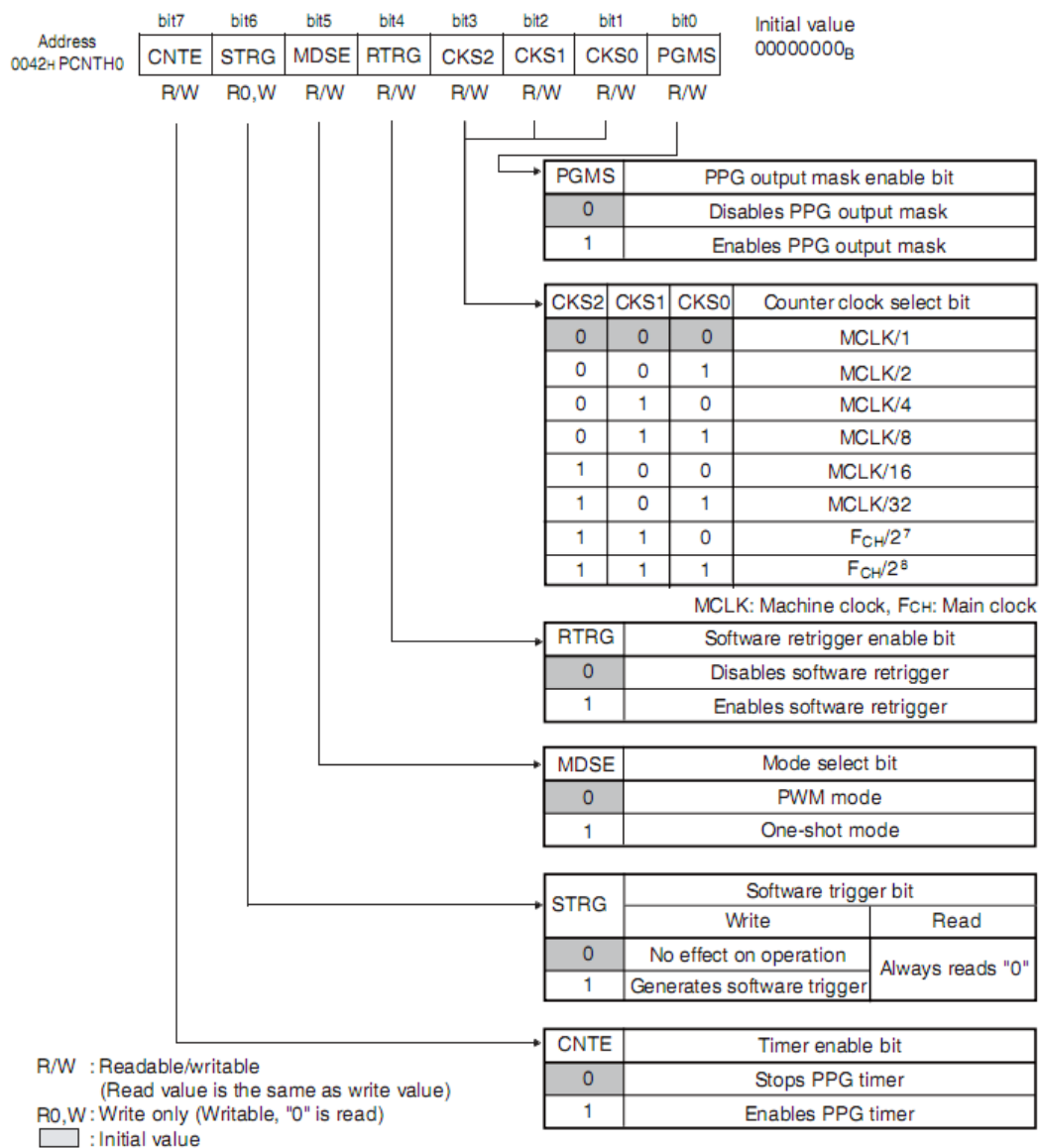


图 3-6: 16 位 PPG 状态控制寄存器-上位字节 (PCNTH0)

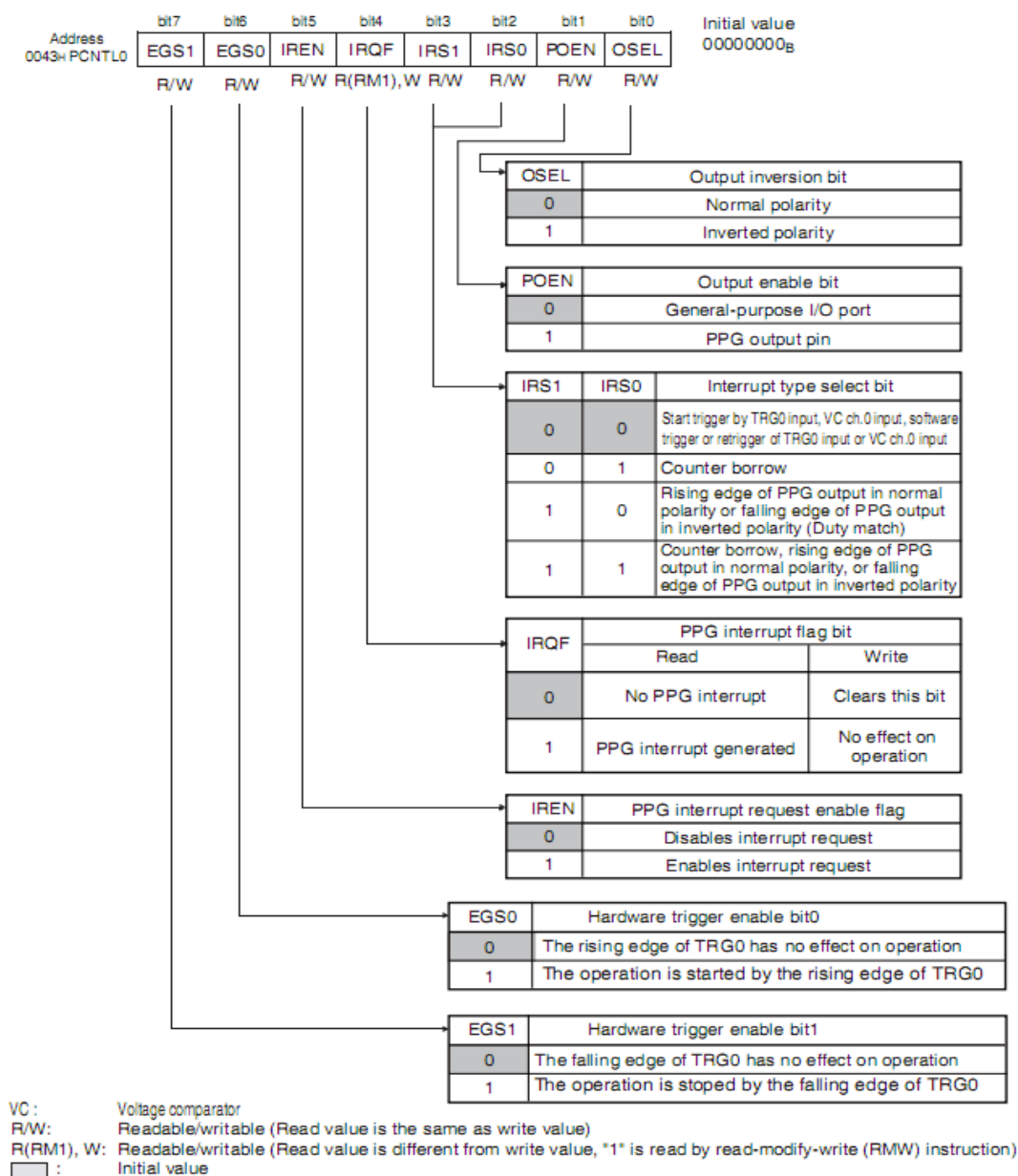


图 3-7: 16 位 PPG 状态控制寄存器-下位字节 (PCNTL0)

3.8 16 位 PPG 触发源控制寄存器

16 位触发源控制寄存器控制 16 位 PPG 定时器的触发源。

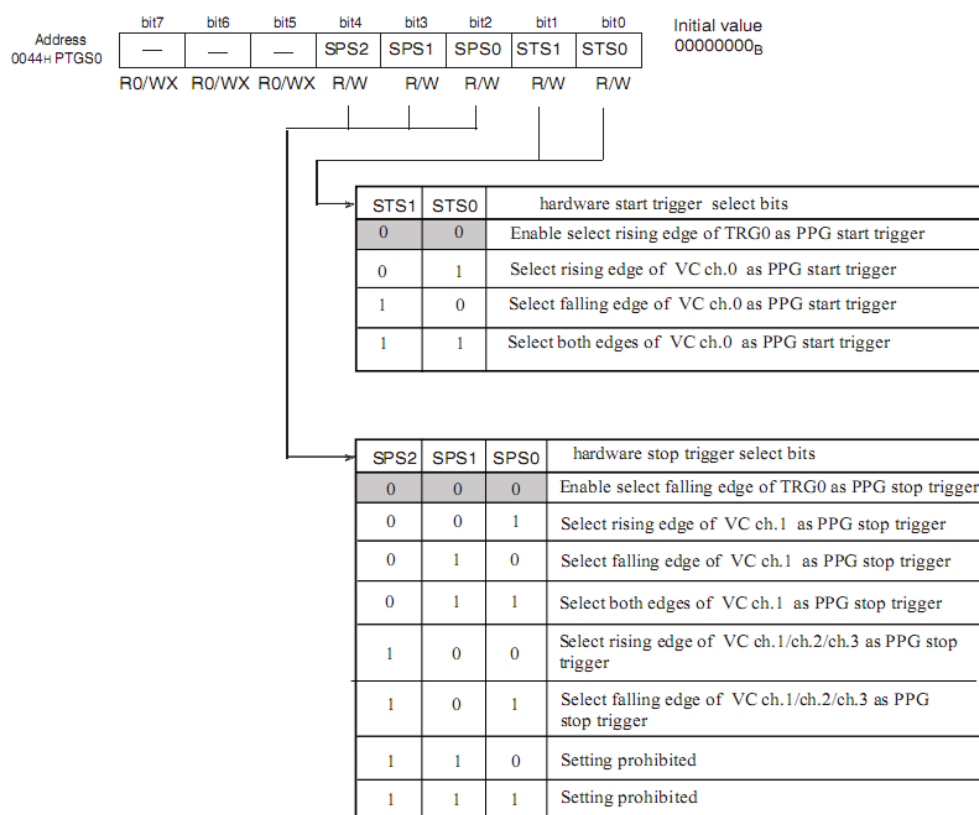


图 3-8: 16 位 PPG 触发源控制寄存器 (PTGS0)

3.9 PPG宏的一般设置程序

初始设置

- 1). 设置中断级别 (ILR3, ILR4)。
- 2). 启用硬件触发器和中断, 选择中断类型, 并启用输出 (PCNTL0)。
- 3). 选择计数时钟和模式, 并启用定时器操作 (PCNTH0)。
- 4). 设置周期 (PCSRH0, PCSRL0)。
- 5). 设置占空比 (PDUTH0, PDUTL0)。
- 6). 通过软件触发器启动 PPG (PCNTH0: STRG = 1)。

中断处理

- 1). 处理任何中断。
- 2). 清除中断请求标志 (PCNTL0: IRQF)。

4 16 位 PPG 定时器的中断

本章介绍了 PPG 中断。

16 位 PPG 定时器在以下情况下可产生中断请求。

触发器或计数器借位发生时

在正常极性下产生 PPG 上升沿时

在反向极性下产生 PPG 下降沿时

中断操作由 PCNTL 寄存器的 IRS1 (bit3) 和 IRS0 (bit2) 控制。

4.1 16 位 PPG 定时器的中断控制位和中断源

Item	Description
Interrupt flag bit	PCNTL0:IRQF
Interrupt request enable bit	PCNTL0:IREN
Interrupt type select bits	PCNTL0:IRS1, IRS0
Interrupt sources	PCNTL0:IRS1, IRS0=00 _B Hardware trigger by TRG Pin input of 16-bit down-counter, software trigger and retrigger
	PCNTL0:IRS1, IRS0=01 _B Counter borrow of 16-bit down-counter
	PCNTL0:IRS1, IRS0=10 _B Rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity
	PCNTL0:IRS1, IRS0=11 _B Counter borrow of 16-bit down-counter, rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity

4.2 与 16 位 PPG 定时器中断相关的寄存器和向量表地址

Interrupt source	Interrupt request no.	Interrupt level setting register		Vector table address	
		Register	Setting bit	Upper	Lower
16-bit PPG timer ch. 1	IRQ17	ILR4	L17	FFD8 _H	FFD9 _H

5 PPG 触发器

本章介绍了 PPG 触发器。

PPG 由 TRG 输入引脚的信号输入或内部电压比较器（VC）的输入激活。

条件 1：PPG 触发开始和触发停止不同时发生。

(1). TRG

当 STS1 和 STS0 设置为 00B，SPS2，SPS1 和 SPS0 设置为 000B，EGS1 和 EGS0 设置为 11B，并使用 TRG 硬件触发输入时，16 位 PPG 定时器在上升沿开始操作，并在检测到下降沿后停止操作。此外，16 位 PPG 定时器也在之后的上升沿开始操作。

选择 TRG 输入硬件触发器后，有效的 TRG 输入硬件触发器可重新触发操作，不管 RTRG 位的重新触发设置如何。

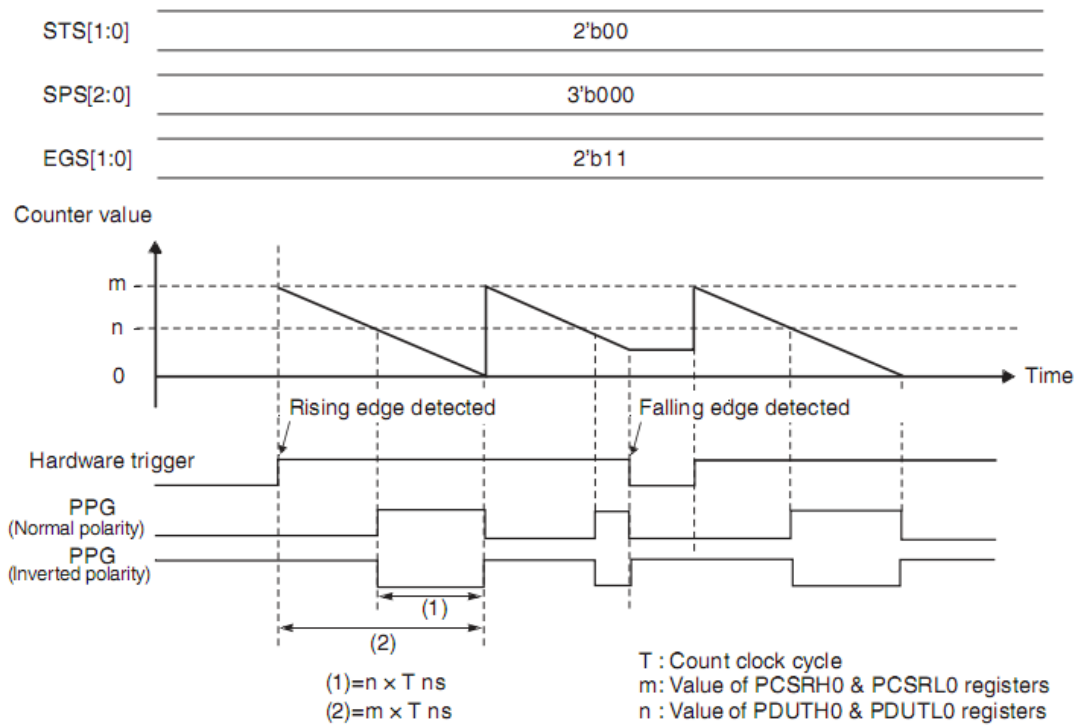


图 5-1： PWM 模式中的 TRG 触发器

(2). 电压比较器（VC）

当 STS1 和 STS0 设置为 01B，SPS2，SPS1 和 SPS0 设置为 101B，并使用来自 VC ch. 0 的硬件触发输入时，16 位 PPG 定时器在 VC ch. 0 的上升沿开始操作，在检测到 VC ch.1/2/3 下降沿时停止操作。

此外，16 位 PPG 定时器也在之后的 VC ch. 0 上升沿开始操作。

选择 VC ch. 0 输入硬件触发器后，有效的 VC ch. 0 输入硬件触发器可重新触发操作，不管 RTRG 位的重新触发设置如何。

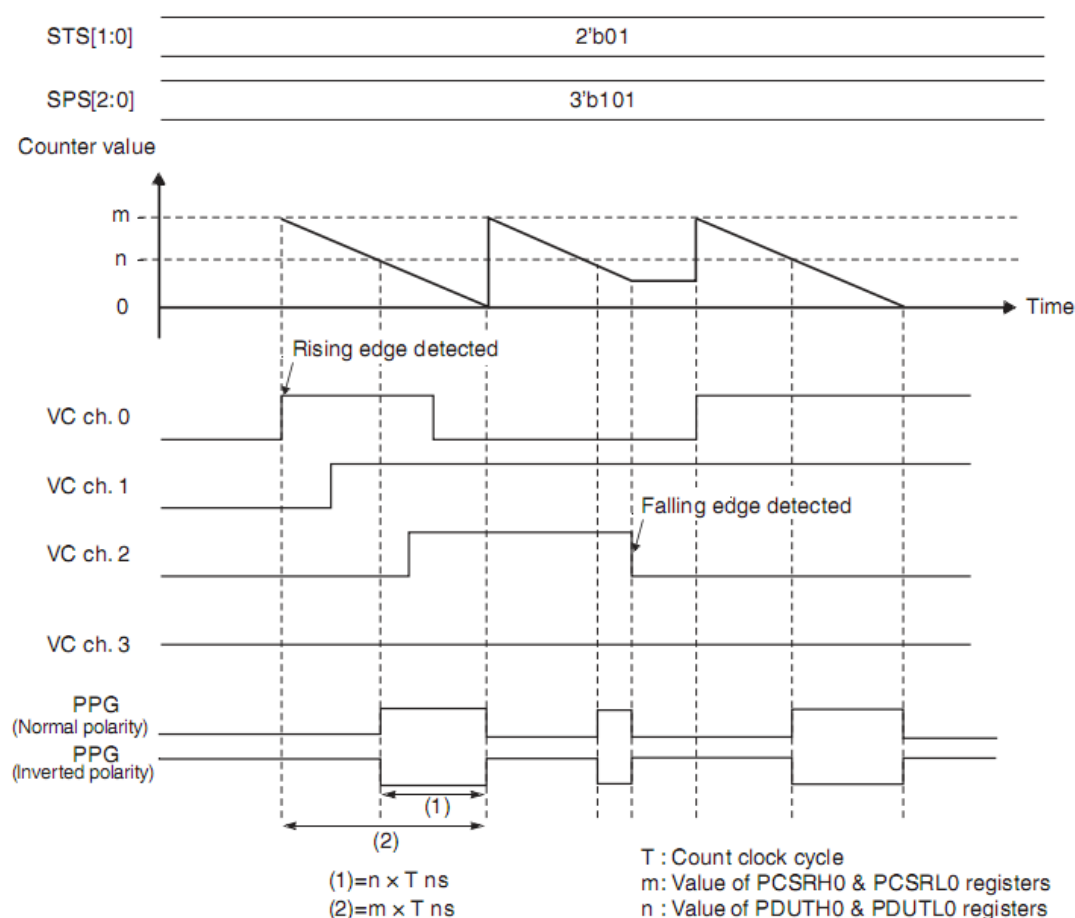


图 5-2: PWM 模式的 TRG 触发器

条件 2: PPG 触发开始和触发停止同时发生。

当 STS1 和 STS0 设置为 01B，SPS2, SPS1 和 SPS0 设置为 001B 时，如果 VC ch. 0 上升沿和 VC ch. 1 上升沿的硬件触发同时发生，PPG 将停止（停止触发器的优先级高于开始触发器）。

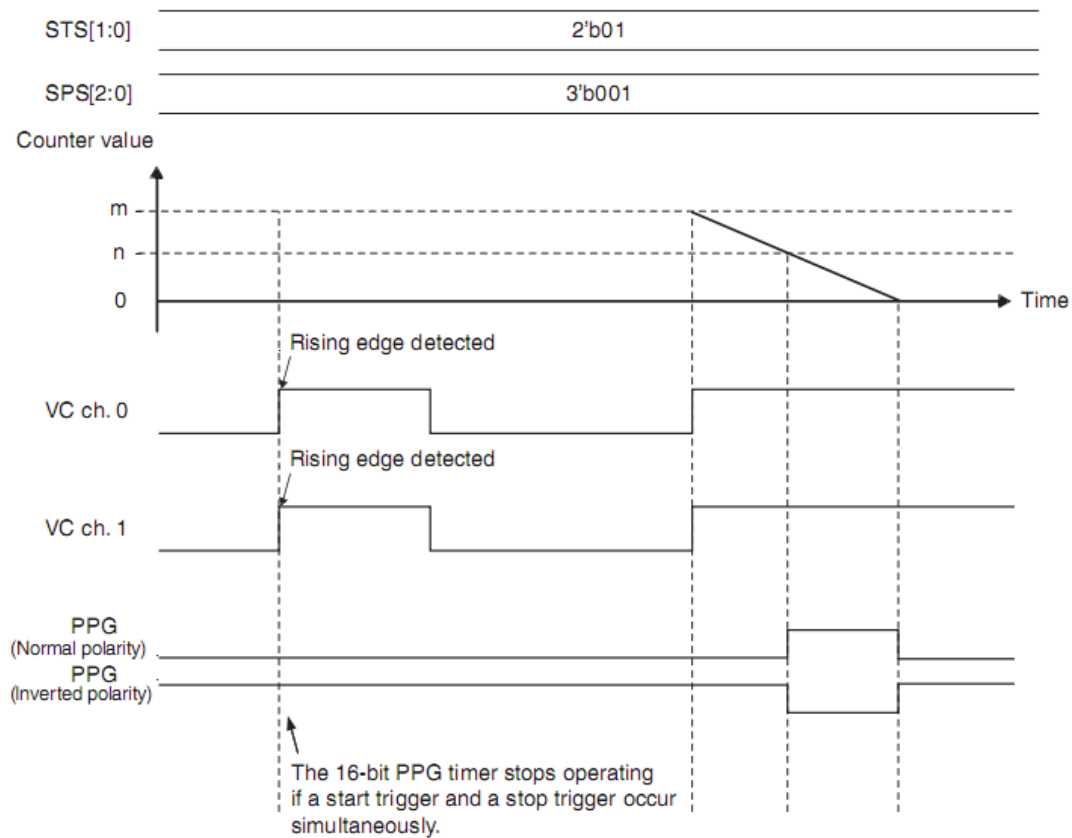


图 5-3: PWM 模式中触发开始和停止同时发生

6 PPG 驱动器

本章介绍了 PPG 驱动器。

6.1 外围设备的使用

MCU 引脚的使用如下：

PPG0：用作 PPG 波形输出

TRG0：用作 PPG 触发器

6.2 驱动代码

6.2.1 一般定义

```
typedef unsigned char  BOOLEAN;

typedef unsigned char  INT8U;           /* Unsigned  8 bit quantity */
typedef signed   char  INT8S;           /* Signed    8 bit quantity */
typedef unsigned int   INT16U;          /* Unsigned 16 bit quantity */
typedef signed   int   INT16S;          /* Signed   16 bit quantity */
typedef unsigned long  INT32U;          /* Unsigned 32 bit quantity */
typedef signed   long  INT32S;          /* Signed   32 bit quantity */


#define BOOL            BOOLEAN
#define BYTE            INT8U
#define UBYTE           INT8U
#define WORD            INT16U
#define UWORD           INT16U
#define LONG            INT32S
#define ULONG           INT32U
#define UCHAR           INT8U
#define UINT             INT16U
#define DWORD           INT32U
```

```
#define TRUE          1

#define FALSE         0

#define BYTE_LO(w)    ((UBYTE)(w))

#define BYTE_HI(w)    ((UBYTE)((UWORD)(w)>>8)&0xFF))
```

6.2.2 PPG程序

void PPGOpen()

返回 : 无

参数 : 无

描述 : 打开 PPG 函数

示例 : PPGOpen();

```
void PPGOpen()
{
    DDR7_P73=1;

    PCNTH0_PGMS=0;//Open PPG
```

void PPGClose ()

返回 : 无

参数 : 无

描述 : 关闭 PPG 函数

示例 : PPGClose();

```
void PPGClose()
{
    DDR7_P73=1;

    PCNTH0_PGMS=1;//Close PPG
}
```

```
void PPGSet(WORD usCycle,WORD usDuty)
```

返回 : 无

参数 : usCycle, is PPG cycle(us)
usDuty, is PPG duty(us)

描述 : 设置 PPG 函数

示例 : PPGSet(40,10);

```
void PPGSet(WORD usCycle,WORD usDuty)
{
    DDR7_P73=1;
//PPG0
    PCSRH0=BYTE_HI(usCycle); //Upper Byte
    PCSRL0=BYTE_LO(usCycle); //Lower Byte
    PDUTH0=BYTE_HI(usDuty); //Upper Byte
    PDUTL0=BYTE_LO(usDuty); //Lower Byte
    PCNTH0=0xF7; //Mask=1,1.000us(MCLK/8),OneShot,Soft Trigger
    PCNTL0=0x0E; //OSEL=L,IREN=0
}
```

7 典型应用

本章介绍了 PPG 的典型应用。

7.1 HW设计

本应用将使用 PPG 驱动蜂鸣器。VC0 上升沿开始 PPG；VC1 上升沿停止 PPG（PTGS0=0x05）。MCU 为 MB95F430K，硬件设计如下图所示。

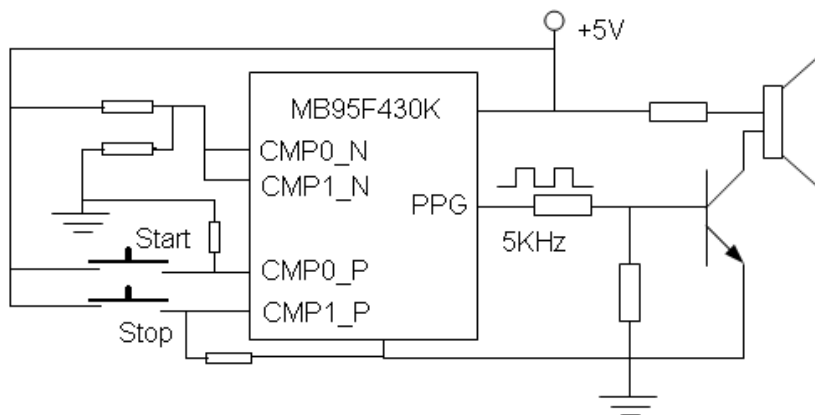


图 7-1：硬件设计

7.2 范例代码

```
void main(void)
```

返回 : 无

参数 : 无

描述 : 系统主程序

示例 : main();

```

void main(void)
{
    __DI();

    __set_il(3);

    InitIrqLevels();

    WDTL = 0xA5; //Disable WTG

    WDTL = 0x96;

    WATR = 0xEE;

    SYCC = 0xF0; //Main Clock
    SYCC2 = 0xF4; //Main Clock
    SYSC = 0xBC; //BUZZ(P01)
    SYSC2 = 0x02; //PPG(P73),Disable I2C
    while(!STBC_MRDY);

    __EI();

    PTGS0 = 0x05; //VC0 rising edge start PPG, VC1 rising edge stop PPG
    PPGSet(200, 10);

    PPGOpen();

    PPGClose();
}

```


8 更多信息

关于富士通半导体更多的产品信息，请访问以下网站：

英文版本地址：

http://www.fujitsu.com/cn/fsp/services/mcu/mb95/application_notes.html

中文版本地址：

http://www.fujitsu.com/cn/fss/services/mcu/mb95/application_notes.html

9 附录

图 2-1: PPG格式	6
图 3-1: 16 位PPG定时器	7
图 3-2: 16 位 PPG定时器的结构图	7
图 3-3: 16 位PPG向下计数器寄存器	9
图 3-4: 16 位 PPG周期寄存器	9
图 3-5: 16 位 PPG占空比寄存器	10
图 3-6: 16 位PPG状态控制寄存器-上位字节 (PCNTH0)	11
图 3-7: 16 位PPG状态控制寄存器-下位字节 (PCNTL0)	12
图 3-8: 16 位PPG触发源控制寄存器 (PTGS0)	13
图 5-1: PWM模式中的TRG触发器	15
图 5-2: PWM模式的TRG触发器.....	16
图 5-3: PWM模式中触发开始和停止同时发生	17
图 7-1: 硬件设计	21

10 代码范例

main.c

```
/*-----  
--*/  
/* Title : Sample main program for sensorless DC inverter control  
*/  
/* Aurthor : Folix Li  
*/  
/* Date : 26th Nov 2999  
*/  
/*-----  
--*/  
#include "mb95430.h"  
#include "TypeDef.h"  
  
/*-----  
--*/  
/* PWM Control  
/*-----  
--*/  
void PPGOpen()  
{  
    DDR7_P73=1;  
    PCNTH0_PGMS=0;  
}  
  
void PPGClose()  
{  
    DDR7_P73=1;  
    PCNTH0_PGMS=1;  
}  
  
void PPGSet(WORD usCycle,WORD usDuty)  
{  
    DDR7_P73=1;  
    //PPG0  
    PCSRH0=BYTE_HI(usCycle);//Upper Byte  
    PCSRL0=BYTE_LO(usCycle);//Lower Byte
```

```

    PDUTH0=BYTE_HI(usDuty);//Upper Byte
    PDUTL0=BYTE_LO(usDuty);//Lower Byte
    PCNTH0=0xF7;//Mask=1,1.000us(MCLK/8),OneShot,Soft Trigger
    PCNTL0=0x0E;//OSEL=L,IREN=0
}

void main(void)
{
    __DI();
    __set_il(3);
    InitIrqLevels();

    WDTN =0xA5;
    WDTL =0x96;

    WATR =0xEE;
    SYCC =0xF0;//Main Clock
    SYCC2=0xF4;//Main Clock
    SYSC  =0xBC;//BUZZ(P01)
    SYSC2 =0x02;//PPG(P73),Disable I2C
    while(!STBC_MRDY);
    __EI();

    PTGS0=0x05;//VC0 rising edge start PPG,VC1 rising edge stop PPG
    PPGSet(200,10);
    PPGOpen();
    PPGClose();
}

```

VECTORS.C

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.
FUJITSU */

/* SEMICONDUCTOR ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR
*/

/* ELIGIBILITY FOR ANY PURPOSES.
*/

```

```
/*          (C) Fujitsu Semiconductor Europe GmbH
*/
/*-----
--
    VECTORS.C
    - Interrupt level (priority) setting
    - Interrupt vector definition
    29.09.04  1.00  HWe    V30L29
-----
--*/

#include "mb95430.h"

/*-----
--
    InitIrqLevels()

    This function pre-sets all interrupt control registers. It can be used
    to set all interrupt priorities in static applications. If this file
    contains assignments to dedicated resources, verify that the
    appropriate controller is used.

    NOTE: value 0xFF disables the interrupt and value 0 sets highest
    priority.

    NOTE: For all resource interrupts exists 3 Interrupt level registers
    (ILRx).

        Each register sets the level for 4 different resources (IRQx).

    NOTE: This list is prepared for the 8FX-emulation chip MB95FV100 'Horn'.
        Not all resources will be supported by all 8FX-devices
-----
--*/

void InitIrqLevels(void)
{
/*  ILRx          IRQs defined by ILRx */

    ILR0 = 0xFF;    // IRQ0:  external interrupt ch0 | ch4
                   // IRQ1:  external interrupt ch1 | ch5
                   // IRQ2:  external interrupt ch2 | ch6
                   // IRQ3:  external interrupt ch3 | ch7
```

```

    ILR1 = 0xFF;      // IRQ4:  UART/SIO  ch0
                      // IRQ5:  8/16-bit timer ch0 (lower)
                      // IRQ6:  8/16-bit timer ch0 (upper)
                      // IRQ7:  Output Compare ch0

    ILR2 = 0xFF;      // IRQ8:  Output Compare ch1
                      // IRQ9:  none
                      // IRQ10: Voltage Compare ch0
                      // IRQ11: Voltage Compare ch1

    ILR3 = 0xFF;      // IRQ12: Voltage Compare ch2
                      // IRQ13: Voltage Compare ch3
                      // IRQ14: 16-bit free run timer
                      // IRQ15: 16-bit PPG0

    ILR4 = 0xFF;      // IRQ16: I2C  ch0
                      // IRQ17: none
                      // IRQ18: 10-bit A/D-converter
                      // IRQ19: Timebase timer

    ILR5 = 0xFF;      // IRQ20: Watch timer
                      // IRQ21: none
                      // IRQ22: none
                      // IRQ23: Flash Memory
}

/*-----
--
    Prototypes

    Add your own prototypes here. Each vector definition needs is proto-
    type. Either do it here or include a header file containing them.

-----
--*/
__interrupt void DefaultIRQHandler(void);

/*-----
--
    Vector definiton

```

Use following statements to define vectors.

All resource related vectors are predefined.

Remaining software interrupts can be added hereas well.

```
-----  
--*/  
  
#pragma intvect DefaultIRQHandler 0    // IRQ0:  external interrupt ch0 |  
ch4  
  
#pragma intvect DefaultIRQHandler 1    // IRQ1:  external interrupt ch1 |  
ch5  
  
#pragma intvect DefaultIRQHandler 2    // IRQ2:  external interrupt ch2 |  
ch6  
  
#pragma intvect DefaultIRQHandler 3    // IRQ3:  external interrupt ch3 |  
ch7  
  
  
#pragma intvect DefaultIRQHandler 4    // IRQ4:  UART/SIO ch0  
#pragma intvect DefaultIRQHandler 5    // IRQ5:  8/16-bit timer ch0 (lower)  
#pragma intvect DefaultIRQHandler 6    // IRQ6:  8/16-bit timer ch0 (upper)  
#pragma intvect DefaultIRQHandler 7    // IRQ7:  Output Compare ch0  
  
  
#pragma intvect DefaultIRQHandler 8    // IRQ8:  Output Compare ch1  
#pragma intvect DefaultIRQHandler 9    // IRQ9:  none  
#pragma intvect DefaultIRQHandler 10   // IRQ10: Voltage Compare ch0  
#pragma intvect DefaultIRQHandler 11   // IRQ11: Voltage Compare ch1  
  
  
#pragma intvect DefaultIRQHandler 12   // IRQ12: Voltage Compare ch2  
#pragma intvect DefaultIRQHandler 13   // IRQ13: Voltage Compare ch3  
#pragma intvect DefaultIRQHandler 14   // IRQ14: 16-bit free run timer  
#pragma intvect DefaultIRQHandler 15   // IRQ15: 16-bit PPG0  
  
  
#pragma intvect DefaultIRQHandler 16   // IRQ16: I2C ch0  
#pragma intvect DefaultIRQHandler 17   // IRQ17: none  
#pragma intvect DefaultIRQHandler 18   // IRQ18: 10-bit A/D-converter  
#pragma intvect DefaultIRQHandler 19   // IRQ19: Timebase timer  
  
  
#pragma intvect DefaultIRQHandler 20   // IRQ20: Watch timer  
#pragma intvect DefaultIRQHandler 21   // IRQ21: none  
#pragma intvect DefaultIRQHandler 22   // IRQ22: none  
#pragma intvect DefaultIRQHandler 23   // IRQ23: Flash Memory  
  
  
/*-----  
--
```

```
DefaultIRQHandler()
```

This function is a placeholder for all vector definitions.

Either use your own placeholder or add necessary code here

(the real used resource interrupt handlers should be defined in the main.c).

```
-----
--*/
__interrupt void DefaultIRQHandler(void)
{
    __DI();                                // disable interrupts
    while(1)
        __wait_nop();                      // halt system
}
```