

# TIMH 前沿消隐 demo 的说明

## 目录

- 1. 在干什么 ..... 2
- 2. 示意图元素的说明 ..... 2
  - 2.1 波形 ..... 2
- 3. 外设的配置说明 ..... 2
  - 3.1 FPCI(故障 PCI) ..... 2
  - 3.2 LEB 配置 ..... 3
- 4. 重要的情况说明 ..... 3
  - 4.1 原因解释 ..... 4

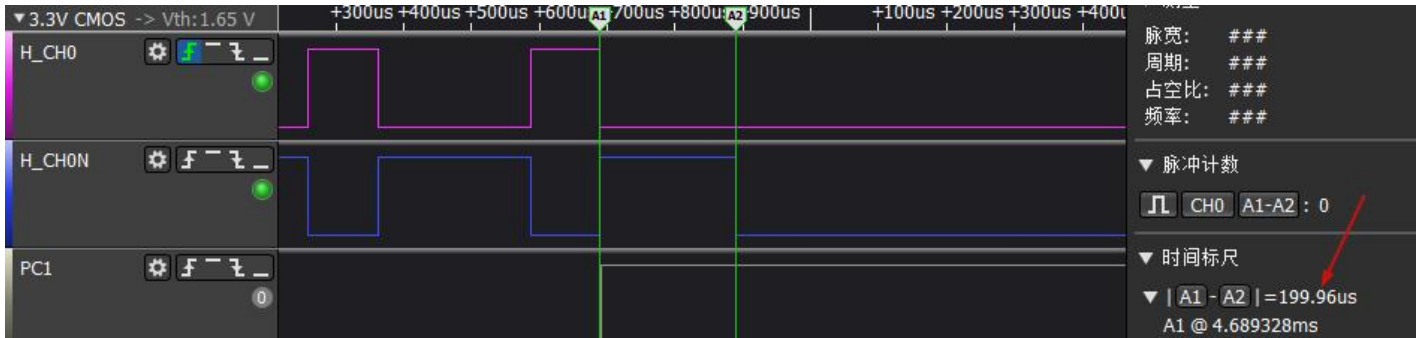


图 0 波形示意图

## 1.在干什么

- TIMH0 输出波形，配置 LEB 功能，启动沿设置为 **PWM\_H 下降沿**，时长 **200us**，选择 **FPCI 模块**控制波形。  
PC1 跳变沿指示驱动信号 ‘1’ 到 FPCI 模块（FPCI active 信号置 1）的时刻，从而立即产生覆盖行为。**覆盖行为**为设置为 **PWM\_H** 和 **PWM\_L** 都强制输出 **0**。
- [timh/leading edge blanking LEB/output waveform.kvdat](#) 文件是波形的抓取文件。

## 2.示意图元素的说明

### 2.1 波形

- **紫色**：PWM\_H，在引脚 PC5 输出。
- **蓝色**：PWM\_L，在引脚 PC8 输出。
- **灰色**：引脚 PC1。

## 3.外设的配置说明

### 3.1 FPCI(故障 PCI)

FPCI 的配置如下图。

```
161 fpciInit.pciSource = PCI_PSS_SOFTWARE; /* FPCI source */
162 fpciInit.softwarePCI_Mode = SWPCIM_TO_RECV_LOGIC;
163 fpciInit.pciAQSS = PCI_AQSS_LEB_ACTIVE; /* AQSS : LEB active */
164 fpciInit.pciTERM = PCI_TERM_AUTO_TERMINATE;
165 fpciInit.acceptMode = PCI_ACP_LATCHED; /* FPCI accept mode : Latched */
166 overWrite.faultPwm_H = 0; /* set FPCI overwrite level, Pwm_H = 0 */
167 overWrite.faultPwm_L = 0; /* set FPCI overwrite level, Pwm_L = 0 */
```

图 3.1 FPCI 配置

### 3.2 LEB 配置

LEB 配置如下图。目标 **LEB 时长 200us**，配置值 0x9C38。公式： $((0.0002(s) * 200,000,000) / 8) - 1) << 3$ 。启动沿为 **PWM\_H 下降沿**。

```
171  /*
172   * Final step : config LEB.
173   * Target LEB : 200us.
174   * Start edge : PWM_H falling edge.
175   * ( ((0.0002(s) * 200,000,000) / 8) - 1 ) << 3 = 0x9C38
176   */
177   leb_trg_sel.PWM_H_Fall = LEB_trige_EN;
178   leb_trg_sel.PWM_H_Rise = LEB_trige_DIS;
179   leb_trg_sel.PWM_L_Fall = LEB_trige_DIS;
180   leb_trg_sel.PWM_L_Rise = LEB_trige_DIS;
181
182   DDL_TIMH_LEB_Config(TIMH0, 0x9C38, &leb_trg_sel);
```

图 3.2 LEB 配置

### 4.重要的情况说明

- PCI 的 active 信号置 1 的时刻，落在 LEB 计时范围内，可以产生预期的消隐（图 0）效果。下图 4.1 为用户手册 TIMH，17.4.4.7 节对 LEB 的描述。下面展示的情况是，active 信号置 1 时刻越过 LEB 计时范围。

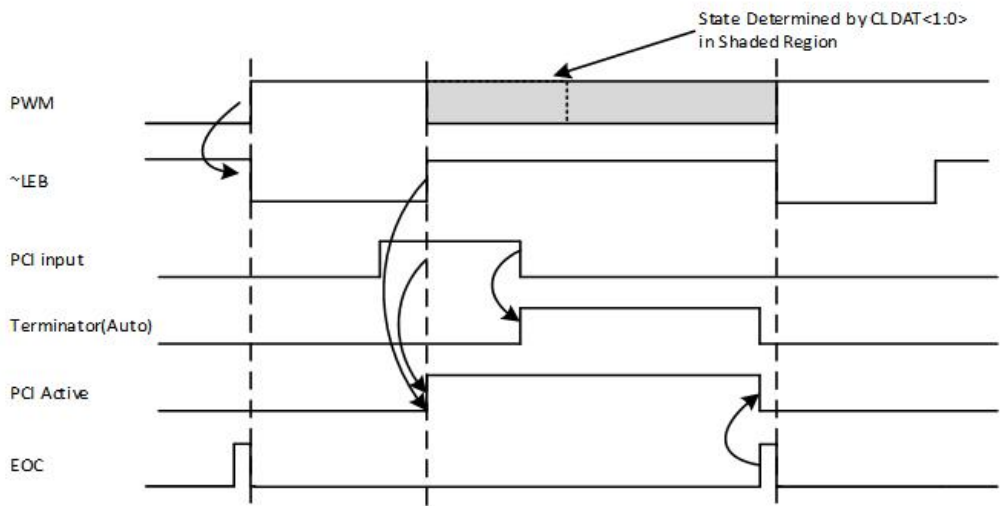


图 4.1 LEB 效果示意图

- 越过 LEB 计时范围的波形如下图 4.2。[timh/leading\\_edge\\_blanking\\_LEB/error\\_output\\_waveform.kvdat](#) 文件是波形的抓取文件。

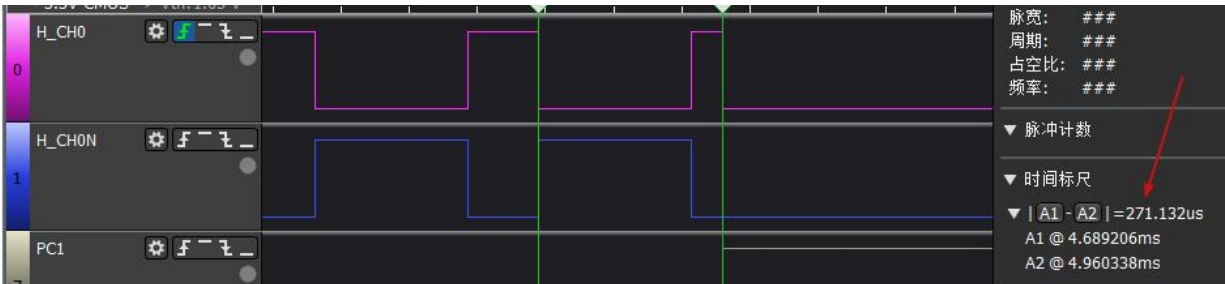


图 4.2 越过 LEB 计数范围

## 4.1 原因解释

- TIMH0 产生的 PWM 周期，低电平时长如下图 4.3 内，红色划线部分所示。

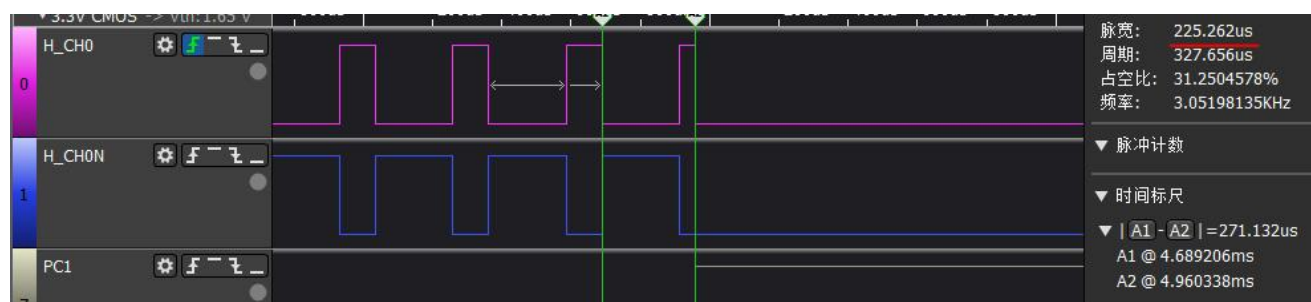


图 4.3 周期内 PWM\_H 低电平时长

- 而 LEB 启动沿设置为 PWM\_H 下降沿，时长 200us。如下图 4.4 红色箭头所示。

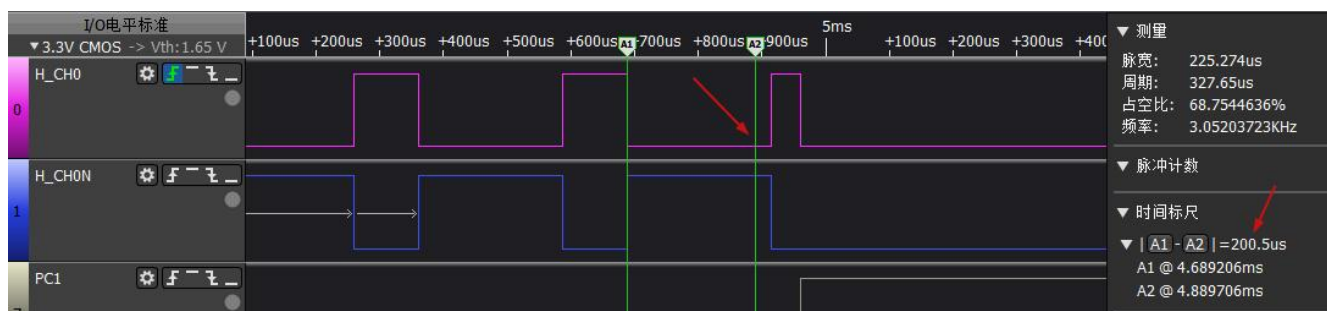


图 4.4 设定的 LEB 计数范围

- 可以看出 PCI 的 active 信号产生的时刻越过了 LEB 计数范围 71.132us 左右。同时又未达到下一个周期 PWM\_H 的下降沿。则产生图 4.2 的效果（LEB 计时看上去未被 PWM\_H 下降沿启动，未推迟 PCI 的 active 信号到 PWM\_H 下降沿之后生效）。
- 通过屏蔽如下图 4.5 的代码可以得到上述的效果。

```

187
188 // /* !! make sure timing that driving signal 1 to FPCI do not exceed LEB time length !!*/
189 // TIMH0->STAT |= TIMH_PGxSTAT_TRIGA;
190 // while((TIMH0->STAT & TIMH_PGxSTAT_TRIGA) != TIMH_PGxSTAT_TRIGA);
191 // TIMH0->STAT |= TIMH_PGxSTAT_TRIGA;
192
193 /* Toggle PA1 to mark the timing when software drive signal 1 to FPCI */
194 DDL_GPIO_TogglePin(GPIOC,GPIO_PIN_1);
195 /*drive signal 1 to FPCI, PWM output overwritten*/
196 __DDL_F_PCI_SWPCI_SET(TIMH0, 1);
197

```

图 4.5 屏蔽的代码

该段代码作用，是确保 PCI 的 active 信号在 PWM\_H 下降沿之后置 1，使其在 LEB 计数范围内。此处利用 Trig\_A 比较点，将其设置为 PWM\_H 下降沿的时刻，不断查询 Trig\_A 的状态位即可。Trig\_A 的配置如图 4.6 红色箭头所示。

```
132  /* config TIMH_0's init struct */
133  timh0_BaseInit.dutyCycle      = 0x4FFF;
134  timh0_BaseInit.period        = 0xFFFF;
135  timh0_BaseInit.phase         = 0;
136  timh0_BaseInit.trig_xValue.trgiA = 0x4FFF;
137  timh0_BaseInit.clockSource    = TIMHx_CLOCKSOURCE_CLK_IN;
138
139  /* Init TIMH_0 to TIMH_2 */
140  DDL_TIMH_Init(TIMH0, &timh0_BaseInit);
141
```

图 4.6 Trig\_A 比较点的配置