acam
mess•electronic

时间数字转换器

应用指南

TDC-GP22

使用 TDC-GP22 时间数字转换器的激光测距仪

28th January 2014
Document-No: AN034_en.pdf V1.3

Published by acam-messelectronic gmbh

Disclaimer / Notes

Support / Contact

For a complete listing of Direct Sales, Distributor and Sales Representative contacts, visit the acam web site at:

http://www.acam.de/sales/distributors/

For technical support you can contact the acam support team in the headquarters in Germany or the Distributor in your country. The contact details of acam in Germany are:

support@acam.de or by phone +49-7244-74190.

内容

1 介绍

有几种不同的方法来实现一个激光测距仪。原理上它们是相似的，都是发射光脉冲，然后接受。一旦返回脉冲被接收到，一些简单的算术和几何公式被用来计算光脉冲的飞行距离。

例如:

- 三角测量
- 调频连续波
- 飞行时间
- …

这些不同的方法都有自己的优，缺点：

	测量距离	精度	评论
三角测量	< 10 米	几 µm	依赖于表面，便宜，结实
调频连续波	< 200 米	近似 10 cm	低生产成本，测量速度慢
飞行时间	几公里	几 mm	反应时间短，价格昂贵，没有光圈

眼睛保护注意事项

对于长的距离，接受到的信号强度很弱。所以要提高激光的功率。出于对眼睛的保护，激光的总功率是受限制的。因此，它是强制性的：为长距离应用，使用脉冲激光。激光脉冲宽度越短，更高的峰值幅。这个应用手册描述的是基于 Acam 公司 TDC-GP22 时间数字转换器的激光测距仪。所以它只描述了 TDC 的使用，而没有描述光学和模拟电路部分。

1.1 Time-of-Flight

飞行时间 –这是一个光脉冲前往目标并返回的时间。已知光的速度，和精确的测量结果：飞行时间，可以计算出距离。许多脉冲顺序被发射，平均响应时间是最常用的。这种技术要求非常精确的亚纳秒定时电路。测量距离用激光或激光扫描仪很好的被建立–你会发现这个技术在地质测量系统，安全系统，生产控制系统中，甚至高尔夫系统中。基于要测量的距离，不同的方法被使用。

小距离，通过三角测量。使用这种方法实现的方案是在微米范围内，但只有几米的最大范围是有限的。

对于 100 米左右的距离，人经常使用的相移测量技术。进行激光调制，传出和传入的光之间的相移给出距离。为了达到毫米范围内的分辨率，非常高的采样率是必要的。只有带有高电流消耗的低测量频率是可能的。使用时间数字转换器，人们有了捷径来数字化飞行时间。它使直接测量光的飞行时间成为可能。原理很简单，但是细节是难点！总所周知，光的飞行速度很快。

.

$$c = 299\,792\,458$$

$$c \coloneqq velocity\ of\ light$$

因此，必须能够处理极短的时间。在仅仅 1 微秒的时间里，光线就可以穿过 300 米！高分辨率意味着在时间测量中的最高精度。一般情况下，光线能够通过一个物体和镜子反射回来，所有光线通过两倍的实际距离。

所以我们得出：

$$distance = \frac{c}{2} * ToF$$

Typical data:

ToF	**Distance**
66.67 µs	10 km
2.4 µs	360 m
6.6 ns	1 m
66 ps	10 mm
10 ps	1.5 mm
6.6 ps	1 mm

这个范围非常适合使用 TDC-GP22。单个芯片 TDC 有一个单一的分辨率 90 ps ，它相当于 13.5 mm 距离。 通过平均可以提高分辨率到 < 10 ps or < 1.5 mm。TDC-GP22 是一个非常通用的集成电路，它可以在不同的模式下被使用。具体哪个模式被选择，我们看一下具体的说明：

2 测量范围从 0 到 300m

TDC-GP22 – 测量范围 1

概述

- 测量范围从 3.5 ns 到 2.0 µs (0 到 2.0 µs 在不同的 Stop 通道之间)
- Typ. 90 ps 分辨率 rms (13.5 mm)
- 可以通过平均来达到更好的分辨率
- 直到每秒 500,000 次测量
- 20 ns 最小脉冲间隔，最多接受 4 个脉冲
- 对于每一个通道的四次采样能力
- 对于每个通道可选择上升 和/或 下降沿触发
- 窗口功能的使能引脚
- 典型应用：手持激光测距仪：可以达到 300 m， 声速枪

框架图

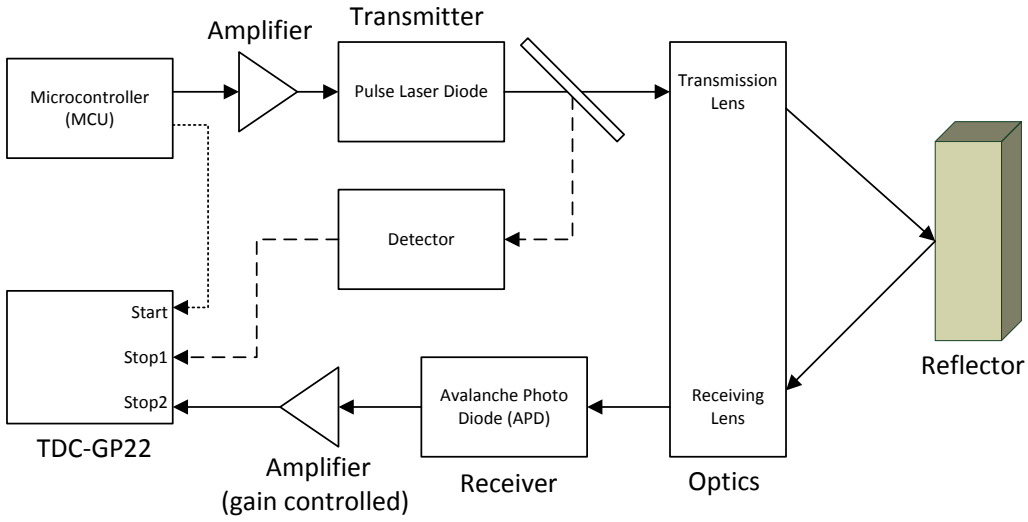


图 2.1 框架图 – 测量范围 1

.温度对于发射和接受途径上影响，可以通过测量参考光束和反射光束之间的间隔来消除。

所以，TDC 从微处理器得到一个假测量的开始。光学参考触发 Stop1,反射光束触发 Stop2

优势:

- 可以测量到零纳秒
- 带有激活的噪声单元的平均可以提高测量结果 (EN_STARTNOISE = “1”)。
- TDC 有必须的统计功能实现通过平均提高分辨率。

2.1 具有独立校准的单次测量

通过 SPI 接口发送 Start_Cal_TDC 指令来进行单独校准。

时间测量的原始数据被内部存储。采样的数量能够从状态寄存器 bits 3 to 8 被看到。当校准被激活的情况下，TDC 测量一个和两个内部参考时钟(Tref * 1, 2 or 4)的周期。校准的原始数据 Cal1 和 Cal2 被内部存储。

时序 - 单次测量无校准

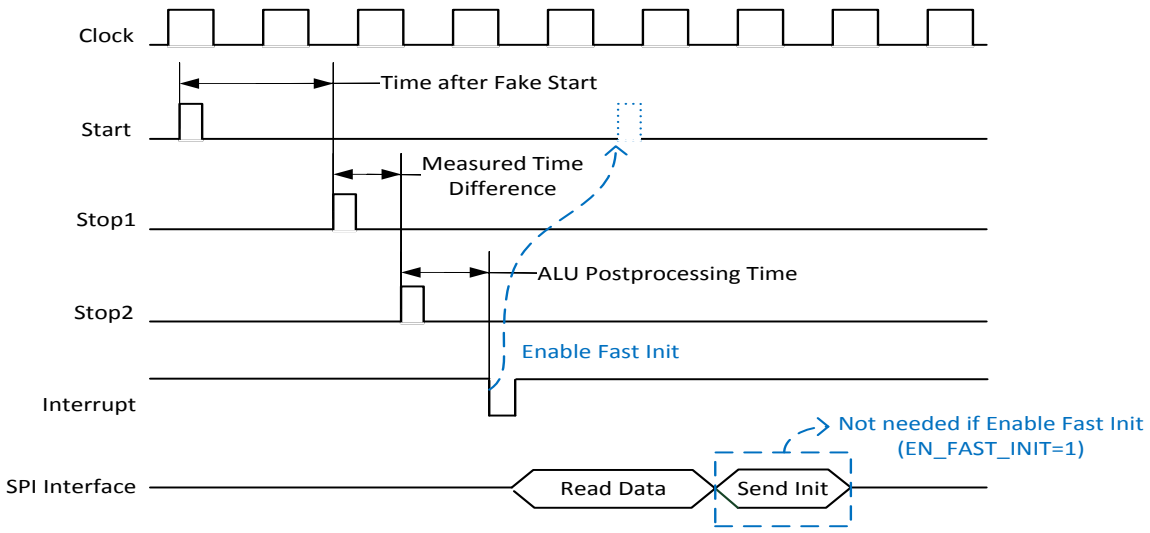


图 2.2 时序 - 单次测量无校准

TDC-GP22 提供快速初始化的可能，因此：EN_FAST_INIT = 1。

有了这个设置 TDC 自动重新初始化，只要结果已被计算出来。它不再需要发送一个 INIT 的操作码。对于高数据速率应用很有意思。

时序 - 单独校准

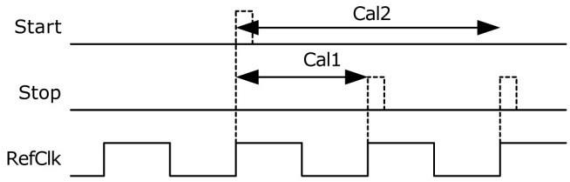


图 2.3 时序-单独校准

设置

设置 register 0, bit 11, MESSB2 = 0

选择测量范围 1

设置 register 5, bit 28, EN_STARTNOISE = 1

对于开始通道，额外的噪声单元启动。它是专门为 TDC 得到一个虚拟的开始，然后测量 STOP1 和 STOP2 的时间差。

设置 register 1, bit 15, EN_FAST_INIT = 1

当数据被读出后，TDC 已经做好准备进行下一次测量。此模式只适用于最高速度的应用。一般情况下用于未校准的测量并且只有一个 Stop。

配置芯片 1 (测量范围 1):

寄存器	值	典型的例子配置
Register 0	'h00241000	DIV_CLKHS = 2, 4 MHz 陶瓷振荡器内部除以 4，内部周期 = 1 µs, 最大的测量时间 2xTref = 2µs START_CLKHS = 1, 晶振持续开启 CALIBRATE = 0, 必须在测量模式 2 下才能开启 NO_CAL_AUTO = 1, 必须在测量模式 2 下才能进行自动校准 MESSB2 = 0, 测量模式 1 对于测量< 2 µs. NEG_STOP/NEG_START = 0, 所有设置为上升沿
Register 1	'h19C90000	HIT2 = 1, HIT1 = 9: 计算 1.Stop CH2 – 1.Stop CH1 在测量模式 1 中 EN_FAST_INIT = 1, 启动快速初始化 HITIN2 = 1, 在 CH2 中，一个采样 HITIN1 = 1, 在 CH1 中，一个采样
Register 2	'hA0000000	EN_INT = b101, Timeout 给出中断

		ALU ready. (see also register 6) RFEDGE1 = RFEDGE2 = 0, 只使用上升沿
Register 3	'h00000000	
Register 4	'h20000000	
Register 5	'h10000000	CON_FIRE = b000, 脉冲发生器的输出配置 EN_STARTNOISE = 1, switch on
Register 6	'h00000000	

更多的 TDC-GP22 设置，请参见我们的技术手册。

2.1.1 平均和单独校准

通过平均可以得到更好的分辨率，为了达到更高的采样率，进行偶尔校准。

原理流程图

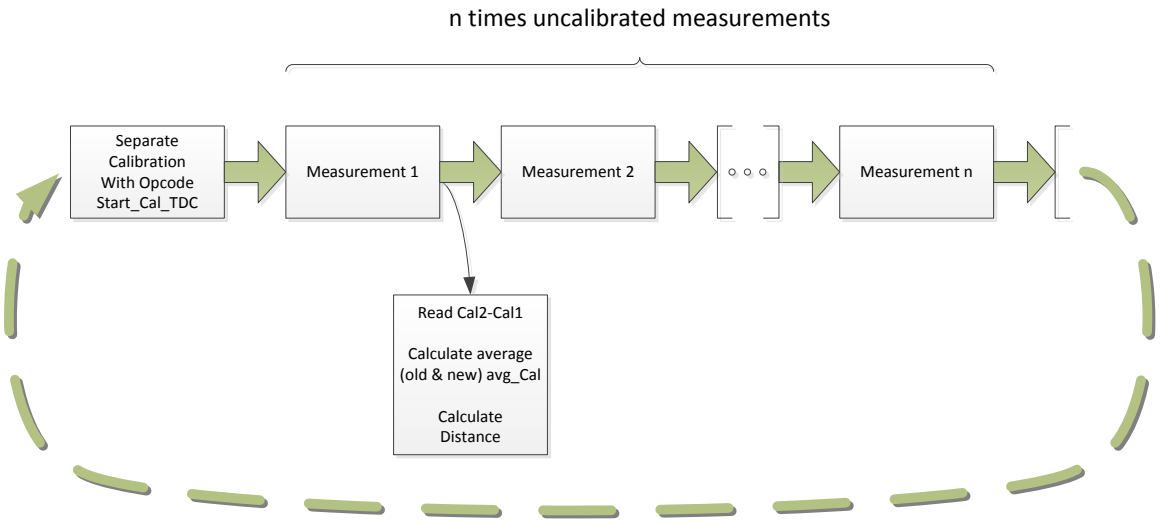


图 2.4 平均及独立校准流程图

$$\text{距离} = ToF\text{的平均结果} * \frac{T_{Ref}}{average\ (Cal2 - Cal1)\ value} * \frac{\text{光速}}{2}$$

2.2 带有自动校准的单次测量

在大多数应用中，自动校准是首选设置。

在这种模式下，用户不需要关心校准。由 TDC 本身进行，一切都是自动完成。但是在这种模式下，测量频率是受限的。

时序图

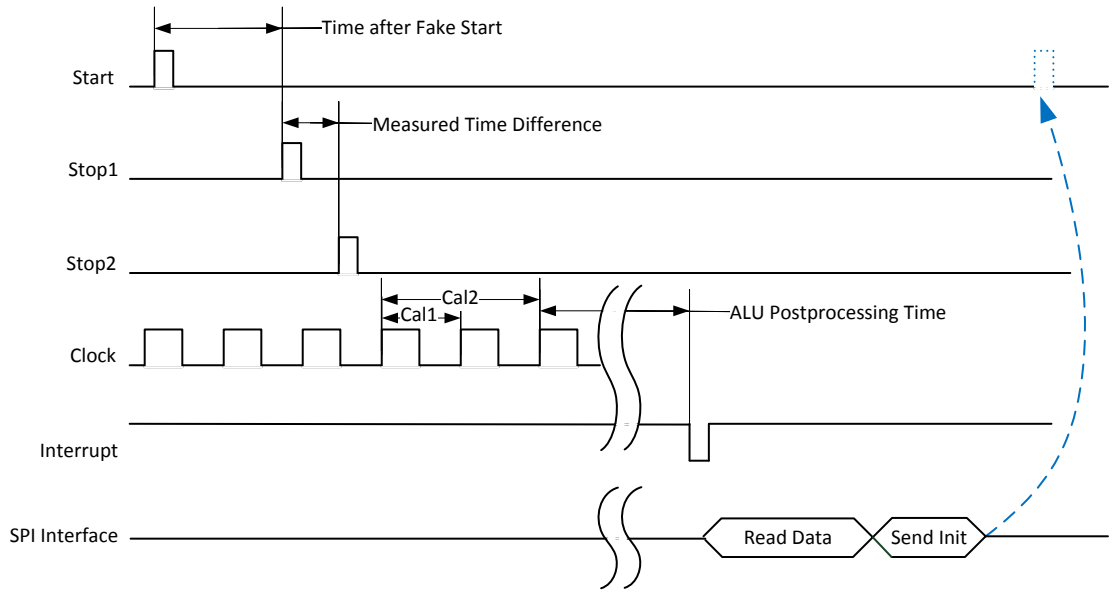


图 2.5 时序 – 带有自动校准的单次测量

设置

Set register 0, bit 11, MESSB2 = 0
选择测量模式 1

Set register 0, bit 12, NO_CAL_AUTO = 0
启用自动校准

Set register 0, bit 13, CALIBRATE = 1
在 ALU 中启动自动校准

Set register 5, bit 28, EN_STARTNOISE = 1
Switch on additional noise unit for start channel. 对于开始通道的额外噪声单元。

它是特别应用于，TDC 得到一个虚拟的启动，然后测量 stop1 和 stop2 之间的时间差。

配置芯片 1 (测量模式 1):

寄存器	值	典型的例子配置
Register 0	‘h00242000	DIV_CLKHS = 2， 4 MHz 陶瓷振荡器内部分频除以 4， internal period = 1 µs，最大测量时间 2xTref = 2µs START_CLKHS = 1，晶振持续开启 CALIBRATE = 1，必须在测量模式 2 下，才能开启。 NO_CAL_AUTO = 0，必须在测量模式 2 下，才可以进行自动校准 MESSB2 = 0，测量模式 1 ，对于测量 < 2 µs. NEG_STOP/NEG_START = 0，所有设置为上升沿
Register 1	’h19490000	HIT2 = 1，HIT1 = 9: calculate 1.Stop CH2 – 1.Stop CH1 在测量模式 1 下 EN_FAST_INIT = 0，off HITIN2 = 1，one expected hit on CH2 HITIN1 = 1，one expected hit on CH1
Register 2	‘hE0000000	EN_INT = b111，通过 Timeout 给出中断，End Hits or ALU ready. (see also register 6) RFEDGE1 = RFEDGE2 = 0，所有设置为上升沿
Register 3	‘h00000000	
Register 4	‘h20000000	
Register 5	’h10000000	CON_FIRE = b000，对于脉冲生成器，输出配置 EN_STARTNOISE = 1，switch on
Register 6	‘h00000000	

更多的 TDC-GP22 设置，请参见我们的技术手册。

3 测量范围从 75 米 到 公里级别

TDC-GP22 – 测量模式 2

测量模式 2 提供远距离测量，可以到几公里。 但是它的测量范围下端受限。 测量模式 2 需要最小 2*Tref 在 Start 和 Stop1 之间。

概述

- 相对于一个开始通道，一个 stop 通道。
- 典型 22 ps / 45 ps / 90 ps 分辨率
- 测量范围从 2 x Tref 到 4 ms @ 4MHz
- 2 x Tref 脉冲对分辨率
- 3 个综合采样能力，完全的自动校准
- 可选择上升沿/下降沿触发
- 集成的可编程窗口，每个 Stop 精度 10 纳秒
- 典型应用: 长距离测距仪(狩猎，高尔夫，测仪器)

框架图

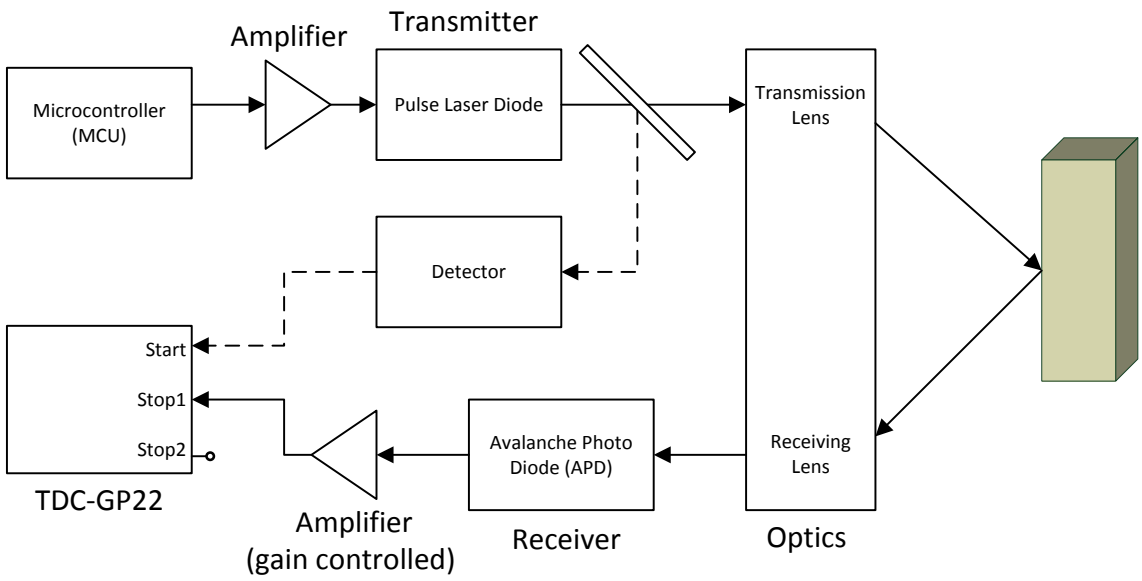


图 3.1 框架图 – 测量模式 2

时序图

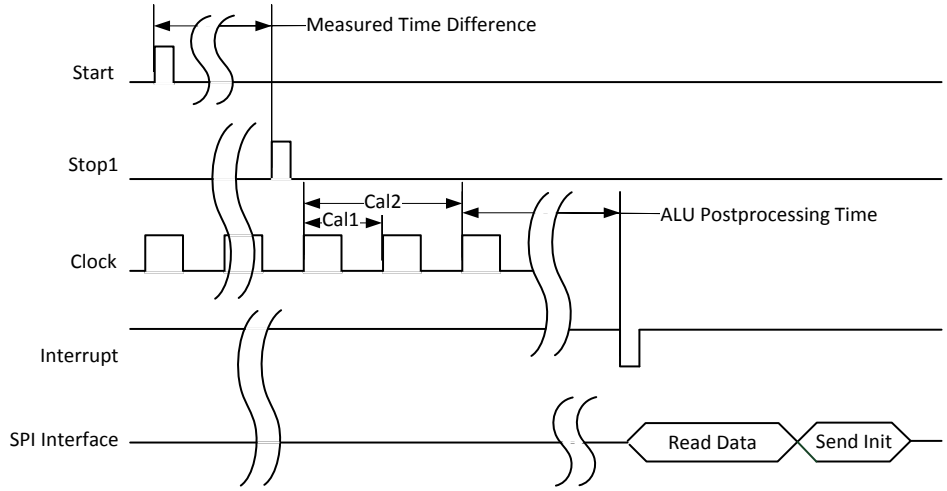


图 3.2 时序 – 测量模式 2

设置

Set register 0, bit 11, MESSB2=1

选择测量模式 2

Set register 0, bit 13, CALIBRATE = 1

在 ALU 中启用校准计算

配置芯片 2 (测量模式 2):

Register	Value	典型配置实例
Register 0	‘h00042800	DIV_CLKHS = 0, 4 MHz 陶瓷振荡器在内部分频，除以 1 START_CLKHS = 1, 晶振持续开启 CALIBRATE = 1, 仅在测量模式 2 中才能被开启 NO_CAL_AUTO = 0, 仅在测量模式 2 中有自动校准功能。 MESSB2 = 1, 测量模式 2 ，对于测量 > 2 µs. NEG_STOP/NEG_START = 0, 所有设置为上升沿
Register 1	’h21420000	HIT2 = 1, HIT1 = 9: 计算 1.Stop CH1 – Start 在测量模式 2 中 EN_FAST_INIT = 0, off HITIN2 = 0, 禁用通道 2 HITIN1 = 2, 在通道 CH1 中，两个采样。
Register 2	‘hE0000000	EN_INT = b111, 通过 Timeout 给出中断， End Hits or ALU ready. (see also register 6) RFEDGE1 = RFEDGE2 = 0, 仅用上升沿
Register 3	‘h00000000	
Register 4	‘h20000000	
Register 5	’h00000000	CON_FIRE = b000, 对于脉冲生成器，输出配置 EN_STARTNOISE = 0, switch off
Register 6	‘h00000000	

更多的 TDC-GP22 设置，请参见我们的技术手册。

4 宽范围应用

有些应用程序可能希望短期和长期相结合的测量范围。

对此，有必要在测量模式 1（短距离）和测量模式 2（长距离）之间进行切换。这可以通过使用由微处理器控制的两个 TDC-GP22 设备，覆盖两个测量模式。

如图所示：

基本原理图

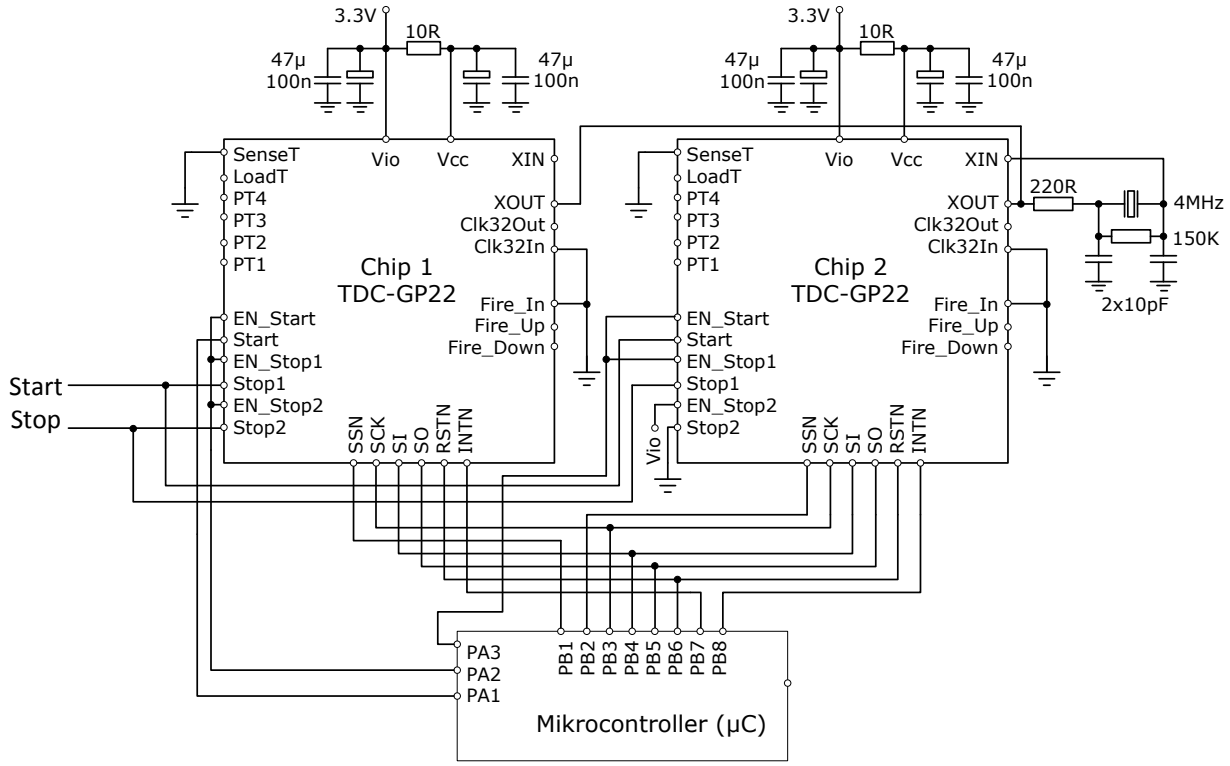


图 4.1 基本原理图

微处理器 (µC) 描述

General Purpose Input/Output (GPIO)	Comment	Connected to...
Port PA1	Start-Input (generated Fake Start)	Chip 1
Port PA2	Enable Pin Start/Stop1/Stop2	Chip 1
Port PA3	Enable Pin Start/Stop1/Stop2	Chip 2
Port PB1	Slave Select (SSN)	Chip 1
Port PB2	Slave Select (SSN)	Chip 2
Port PB3	Clock Serial Interface (SCK)	Chip 1 / Chip 2
Port PB4	Data Input Serial Interface (SI)	Chip 1 / Chip 2
Port PB5	Data Output Serial Interface (SO)	Chip 1 / Chip 2
Port PB6	Reset Input (RSTN)	Chip 1 / Chip 2
Port PB7	Interrupt Flag (INTN)	Chip 1
Port PB8	Interrupt Flag (INTN)	Chip 2

FlowChart

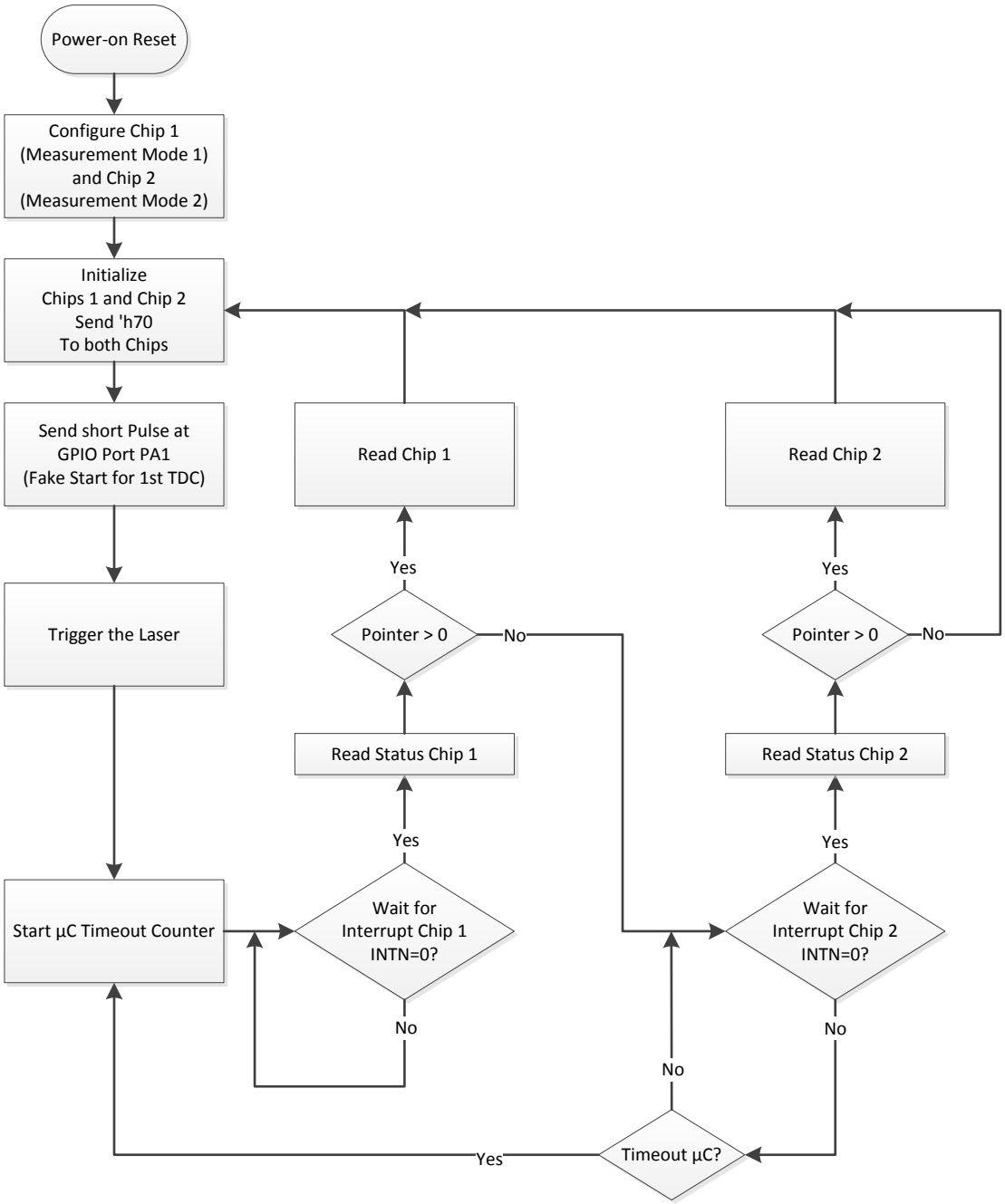


图 4.1 流程图

设置

更多的 TDC-GP22 设置，请参见我们的技术手册。

5 示例代码

这是一个通用的代码示例，对于一个完整的激光测距仪测量流程，包括高速陶瓷谐振器的时钟校准，写入 STM32 微处理器。整个源代码可以在下面的链接下载：

http://www.acam.de/fileadmin/Download/_software/TDC/GP22_main_ANO34.c

5.1 Detailed Flow Chart

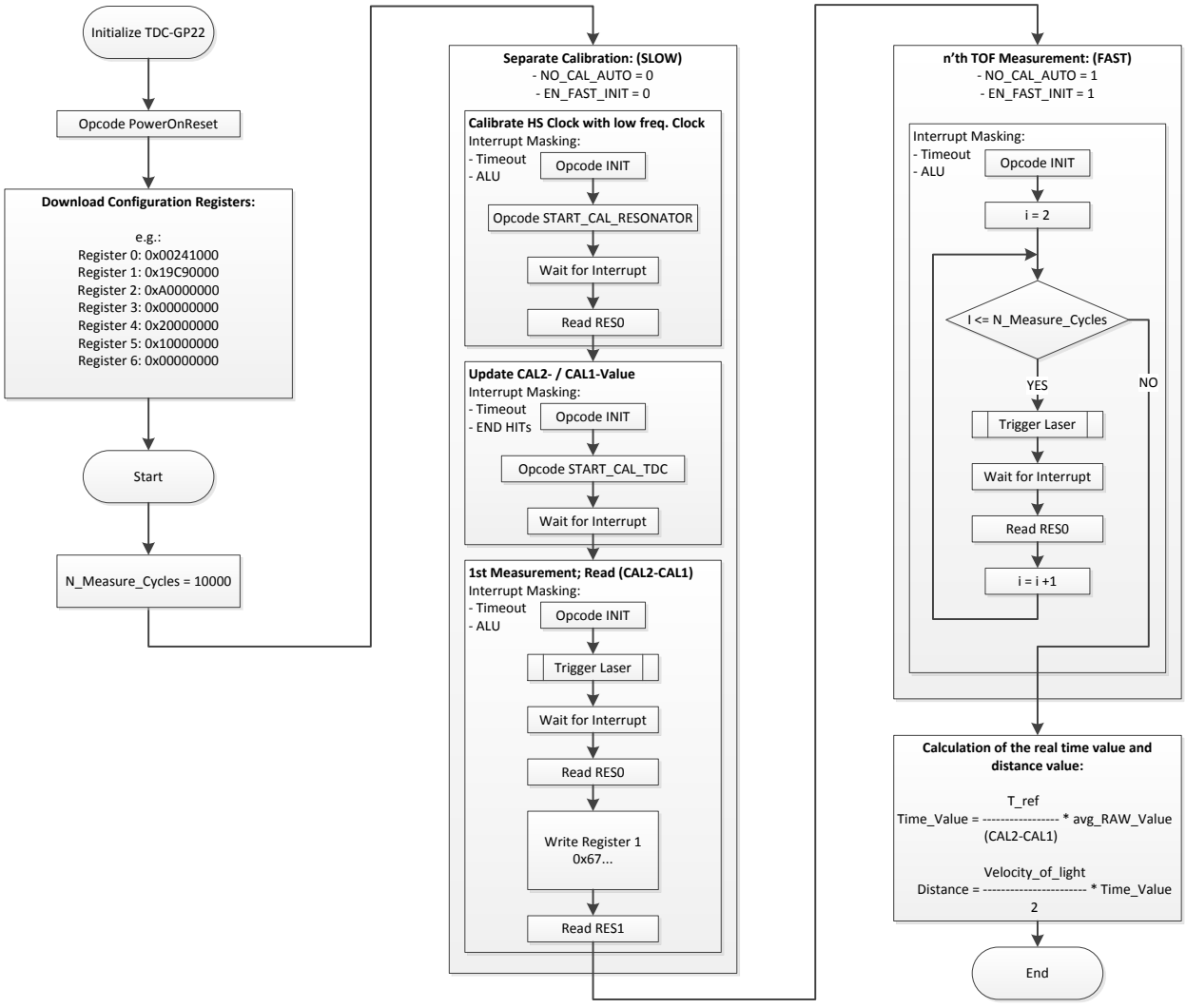


Figure 5.1 n'th Measurement with separate Calibration

5.2 Example Code

The whole source code can be downloaded from our download center.

```
/*****************************************************************************
* Function Name  : main
* Description    : Main program.
* Input          : None
* Output         : None
* Return         : None
*****************************************************************************/
void main(void)
{
  ENTR_CRT_SECTION();
  /* Setup STM32 system (clock, PLL and Flash configuration) */
  SystemInit();

  EXT_CRT_SECTION();

  // Choose your Slot (SPI1, SPI2)
  void* Bus_Type = SPI1;

  /* controlled loop */
  while (Dummy_var!=11) // To control the loop, e.g. (Dummy_var!=7)
  {
    if (Dummy_var==10) Dummy_var=0; // Infinite loop

    if(configured_true==FALSE)
    {
      configured_true = TRUE;
      SPIx_GPIOs_Init(Bus_Type);
      SPIx_Interface_Init(Bus_Type);

#ifdef EMBEDDED_SRAM
      Embed_SRAM_Init();
#endif

      Ext_Interrupt_Init();

      gp22_send_1byte(Bus_Type, Power_On_Reset);
      Dly100us((void*)5);              // 500 us wait for GP22

      //  Setting of the Configuration Registers
      // CR0: DIV_CLKHS = 2, START_CLKHS = 1, CALIBRATE = 0, MESSB2 = 0, NEG_STOP =
NEGSTART = 0, ...
      Register_0        = 0x00240000; // NO_CAL_AUTO = 0
      Register_0_NO_CAL = 0x00241000; // NO_CAL_AUTO = 1

      // CR1: HITIN2 = 1, HITIN1 = 1, ...
      Register_1      = 0x19490000; // EN_FAST_INIT = 0
      Register_1_FAST = 0x19C90000; // EN_FAST_INIT = 1

      // CR2: EN_INT, RFEDGE1 = RFEDGE2 = 0, ...
      // (NOTE: EN_INT = b111, it doesn't work with using EN_FAST_INT)
      Register_2 = 0xA0000000; // EN_INT = Timeout(8) _ End HITs(4) _ ALU interrupt(2)

      // CR3: ...
      Register_3 = 0x00000000;
```

```
      // CR4: ...
      Register_4 = 0x20000000;

      // CR5: CON_FIRE = b000, EN_STARTNOISE = 1, ...
      Register_5 = 0x10000000;

      // CR6: QUAD_RES = 0, ...
      Register_6 = 0x00000000;

      // Writing to the configuration registers (CR)
      gp22_wr_config_reg(Bus_Type, 0x80, Register_0_NO_CAL);
      gp22_wr_config_reg(Bus_Type, 0x81, Register_1_FAST);
      gp22_wr_config_reg(Bus_Type, 0x82, Register_2);
      gp22_wr_config_reg(Bus_Type, 0x83, Register_3);
      gp22_wr_config_reg(Bus_Type, 0x84, Register_4);
      gp22_wr_config_reg(Bus_Type, 0x85, Register_5);
      gp22_wr_config_reg(Bus_Type, 0x86, Register_6);

    }

    // ....................................................................
    // ...START_CAL_RESONATOR..Calibrate High Speed Clock.....................
    // ...START_CAL_TDC.....Update the CAL2- and CAL1-Value...................
    // ...................Laser Rangefinder Measurement CYCLE............LOOP...
    // ........................Caluculate Result Values.......................

    N_Measure_Cycles = 10000;

    diff_Cal2_Cal1_old = diff_Cal2_Cal1_new;

if((Dummy_var==0) | (Dummy_var==10))
{
    //--------------------------------------------------------------------------
    // Start Calibrate High Speed Clock Cycle (-->SLOW)
    // Important Note: NO_CAL_AUTO and EN_FAST_INIT need to be cleared!
    gp22_wr_config_reg(Bus_Type, 0x80, Register_0); // NO_CAL_AUTO = 0
    gp22_wr_config_reg(Bus_Type, 0x81, Register_1); // EN_FAST_INIT = 0

    gp22_send_1byte(Bus_Type, Init);
    gp22_send_1byte(Bus_Type, Start_Cal_Resonator);

    // Wait for INT Slot_x
    if (Bus_Type==SPI1) while (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_4)==1);
    if (Bus_Type==SPI2) while (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_11)==1);

    //Calculate Correction factor
    //The time interval to be measured is set by ANZ_PER_CALRES
    //which defines the number of periods of the 32.768 kHz clock:
    //2 periods = 61.03515625 µs
    CLKHS_freq_corr_fact = 61.03515625/
      gp22_read_n_bytes_int(Bus_Type, 2, 0xB0, 0x00) * CLKHS_freq;              // read
only two bytes

    printf("\n Correction factor for clock = %1.4f\n", CLKHS_freq_corr_fact);

    CLKHS_freq_cal = CLKHS_freq * CLKHS_freq_corr_fact; // Calibrated Clock frequency

    //--------------------------------------------------------------------------
    // Start Separate Calibration Measurement Cycle
    // Important Note: EN_INT = End HITs
    gp22_wr_config_reg(Bus_Type, 0x82, 0x40000000); // End HITs
```

```
    gp22_send_1byte(Bus_Type, Init);
    gp22_send_1byte(Bus_Type, Start_Cal_TDC); // update calibration data

    // Note:
    // The calibration data are not addressed directly after the calibration
    // measurement but after the next regular measurement;

    // Wait for INT Slot_x
    if (Bus_Type==SPI1) while (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_4)==1);
    if (Bus_Type==SPI2) while (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_11)==1);

    // Important Note: After Separate Calibration Measurement Cycle
    // EN_INT = ALU interrupt
    gp22_wr_config_reg(Bus_Type, 0x82, Register_2); // Timeout + ALU interrupt

    //-------------------------------------------------------------------------
    // 1st ToF Measurement plus calibration data readout
    // Note: (NO_CAL_AUTO = 0 / EN_FAST_INIT = 0) --> SLOW
    gp22_send_1byte(Bus_Type, Init);

    //Trigger pulse laser
    //    SetPortHigh;
    GPIO_SetBits(GPIOD, GPIO_Pin_8);    // Output HIGH
    //    SetPortLow;
    GPIO_ResetBits(GPIOD, GPIO_Pin_8);  // Output LOW

    // Wait for INT Slot_x
    if (Bus_Type==SPI1) while (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_4)==1);
    if (Bus_Type==SPI2) while (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_11)==1);

    // First regular measurement (to readout calibration data)
    RAW_Result_int = gp22_read_n_bytes_int(Bus_Type,2,0xB0,0x00);               // read
only two bytes

#ifdef EMBEDDED_SRAM
    Write_Emb_SRAM_uint32_t(RAW_Result_int);  // writes the first value into SRAM
#endif

//    printf("\n 1. Measured RAW Value = %u \n",RAW_Result_int); // RAW value

    // Check Status Register, next free result register
//    printf("Stat_Reg = 0x%04X \n",gp22_read_status_bytes(Bus_Type));

    // readout the new calibration data from result register adr 0x01
    gp22_wr_config_reg(Bus_Type, 0x81, 0x67490000);
    diff_Cal2_Cal1_new = gp22_read_n_bytes_int(Bus_Type,2,0xB0,0x01);           // read
only two bytes

#ifdef EMBEDDED_SRAM
    Write_Emb_SRAM_uint32_t(diff_Cal2_Cal1_new);
#endif

}

    //-------------------------------------------------------------------------
    // Calculate the real time after the hole first cycle loop
    while (diff_Cal2_Cal1_old != 0)
    {
        avg_diff_Cal2_Cal1 = (diff_Cal2_Cal1_new+diff_Cal2_Cal1_old) / 2;

//        printf("\n OLD Cal2-Cal1 RAW Value = %.0f \n",diff_Cal2_Cal1_old);
//        printf("\n NEW Cal2-Cal1 RAW Value = %.0f \n",diff_Cal2_Cal1_new);
```

```
        average_RAW_Result /= N_Measure_Cycles;

        // Used Formulas:
        // ----------------------------------------------------
        //                 T_ref
        // Time_Value = ----------- * measured_RAW_Value
        //               Cal2-Cal1
        // ----------------------------------------------------
        //                   velocity_of_light
        // Distance_Value = ------------------- * Time_Value
        //                            2
        // ----------------------------------------------------

        // For this Source Code would be a Reference Clock used with 1 MHz

        Time_Result = (average_RAW_Result/avg_diff_Cal2_Cal1) * 1000;//time [ns]

        Distance_Result = Time_Result / 6.671281904; //distance [m]

        printf("\n Time Measure Result (ToF) = %.3f ns\n",Time_Result);
        printf(" corresponds to %.3f m of Distance\n",Distance_Result);
        printf(" to reflected point after %u Measurements\n",N_Measure_Cycles);

        diff_Cal2_Cal1_old = 0;
    }

    //-------------------------------------------------------------------------
    // if more than one measure cycle (-->FAST)
    average_RAW_Result = RAW_Result_int; // set first value of average_RAW_Result

    gp22_wr_config_reg(Bus_Type, 0x80, Register_0_NO_CAL); // NO_CAL_AUTO = 1
    gp22_wr_config_reg(Bus_Type, 0x81, Register_1_FAST);   // EN_FAST_INIT = 1
    gp22_send_1byte(Bus_Type, Init);

    //-------------------------------------------------------------------------
    // n'th ToF Measurement
    for (int i=2; i<=N_Measure_Cycles;i++)
    {
        //Trigger pulse laser
        //    SetPortHigh;
        GPIO_SetBits(GPIOD, GPIO_Pin_8);    // Output HIGH
        //    SetPortLow;
        GPIO_ResetBits(GPIOD, GPIO_Pin_8);  // Output LOW

        // Wait for INT Slot_x
        if (Bus_Type==SPI1) while (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_4)==1);
        if (Bus_Type==SPI2) while (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_11)==1);

        RAW_Result_int = gp22_read_n_bytes_int(Bus_Type,2,0xB0,0x00);           // read
only two bytes
//        printf(" %u. Measure RAW Value = %.0f \n",i,RAW_Result_int); // RAW value
        average_RAW_Result += RAW_Result_int;

#ifdef EMBEDDED_SRAM
        Write_Emb_SRAM_uint32_t(RAW_Result_int); // writes the next values into SRAM
#endif

    }

    //-------------------------------------------------------------------------
```

```
    printf("\nNEW CYCLE...\n");

    Dummy_var++; // To Control the loop

#ifdef EMBEDDED_SRAM
    // clear internal SRAM of µC
    sram_mem_offset = 0x0;
#endif

  } // End while Dummy_var

} //End main
```

6 其他

6.1 参考资料

EECS University of Central Florida – Department of Electrical Engineering and Computer Science

http://www.eecs.ucf.edu

Project: “Portable Sensing Field Sensor”, Group-No. 4 (2011)

http://www.eecs.ucf.edu/seniordesign/sp2011su2011/g04/index.html

Project: “Red Eye”, Group-No. 8 (2011)

http://www.eecs.ucf.edu/seniordesign/su2011fa2011/g08/index.html

ECE ILLINOIS University of Illinois – Department of Electrical and Computer

http://www.ece.illinois.edu/

Project “1D LiDAR TOF Rangefinder”, Group-No.28 (2012)

https://courses.engr.illinois.edu/ece445/?f=Projects&sem=spring2012&proj=28#a28

Project “Laser Range Finder”, Group-No. 3 (2007)

https://courses.engr.illinois.edu/ece445/?f=Projects&sem=summer2007&proj=3#a3

6.2 材料指南

Data Sheets

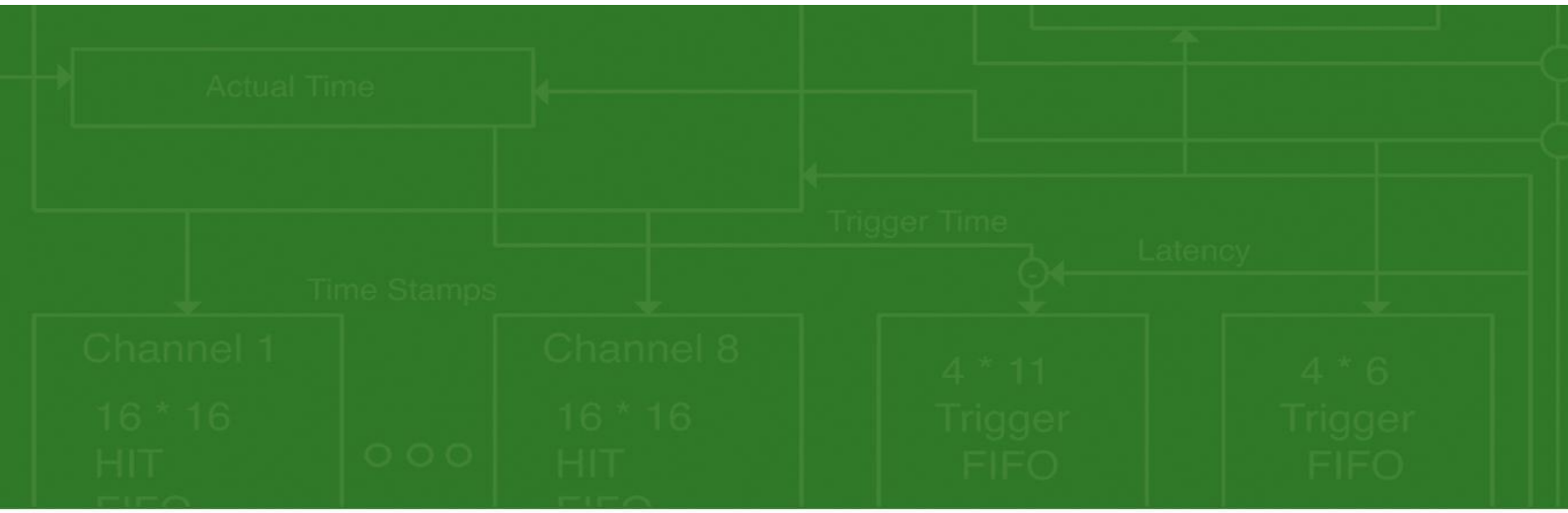
Title	Document-No
TDC-GP22 Universal 2-Channel Time-to-Digital Converter dedicated to Ultrasonic Heat & Water Meters	DB_GP22_x.pdf

The latest version of the available document can be downloaded from the acam website

http://www.acam.de/download-center/time-to-digital-converters/

6.3 文档历史记录

27.02.2013	First release
25.03.2013	Version 1.1 for release, page 14, Basic schematics revised
08.01.2014	Version 1.2 for release, section 7 Example Code including part "Calibrate High Speed Clock" and EN_FAST_INIT modified
28.01.2014	Version 1.3 for release, section 2.1, Register 2 revised; Detailed flow chart of example source code added; Example source code regarding timing optimized



acam-messelectronic gmbh
Friedrich-List-Straße 4
76297 Stutensee-Blankenloch
Germany
Phone +49 7244 7419 – 0
Fax +49 7244 7419 – 29
E-Mail support@acam.de
www.acam.de