**I2C-Master IP Core**

**User Guide**

张泽

Rev. V1.0

2017年12月13日

1. **简介**

I2C是一种简单的串行通讯总线，由飞利浦公司在1980年代为让主板、嵌入式系统或手机用以连接低速周边设备而发展。自2006年11月起，I2C协议是可以被免费使用的，但是芯片厂商仍需要付费以获得I2C从属设备的地址。

I2C只使用两条双向漏极开路（串行数据SDA与串行时钟SCL）并利用电阻进行上拉，I2C允许相当大的工作电压范围，但典型电压等级为+3.3V或者5V。其设备地址包含7bit长度与10bit长度。I2C传输速率有不同的模式：

·标准模式：100Kbit/s

·低速模式：10Kbit/s

·快速模式：400Kbit/s

·高速模式：3.4Mbit/s

本IP的设计旨在使用户能够高效进行开发，减少开发周期。支持7位设备地址与标准/快速/低速模式，暂不支持10位地址操作模式，若相关器件性能支持，该IP也可运行在3.4Mbit/s(所使用的FPGA IO建立时间需满足时序要求)。

**注：本IP仅限学习交流使用，禁止商业用途。**

1. **I2C Master Core特性**

·符合Philips I2C标准

·支持主动挂起I2C总线进入等待状态

·支持多种传输速率

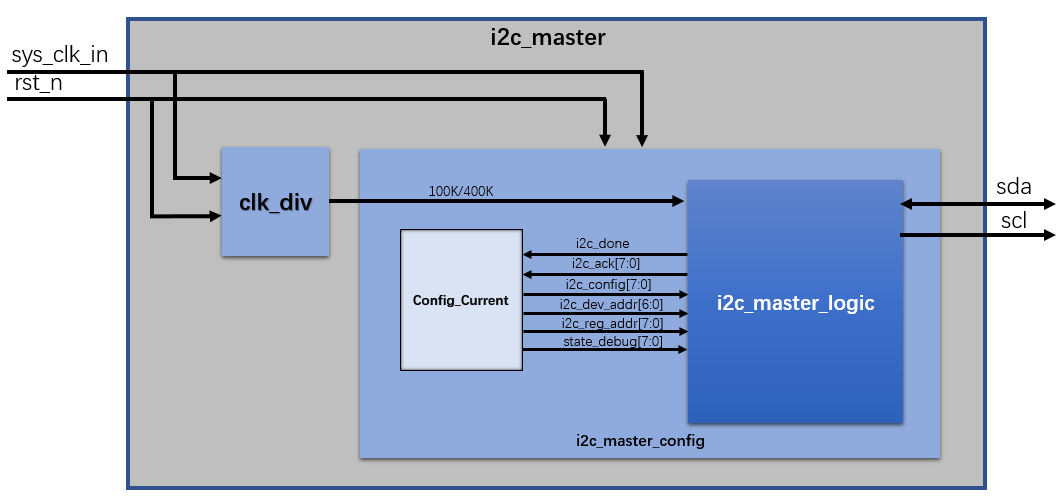
·支持重复读/写操作

·支持非标准I2C设备（如无寄存器地址操作）

·提供ACK触发信号

·提供状态调试寄存器

1. **I2C Master Core结构**



I2C\_Master\_Core由4部分组成，分别为i2c\_master, i2c\_master\_config, i2c\_master\_logic与clk\_div，功能分别如下：

·i2c\_master : 为内核的top module，调用所有module并建立连接。

·i2c\_master\_config : 为内核配置module, 用户可修改该module实现i2c通信。

·i2c\_master\_logic : 为内部逻辑实现module，若预设操作模式没有涵盖到某种I2C设备，你可以自行修改该module中的内容，建立一种新的状态跳变模式。

·clk\_civ : 时钟分频module，可进行任意整数分频，点击[详细了解](http://www.stepfpga.com/doc/5._%E6%97%B6%E9%92%9F%E5%88%86%E9%A2%91)。

1. **Signal & I/O Ports**

4.1 i2c\_master\_logic Module

I2C\_Master\_Logic Module是I2C Master Core逻辑实现部分，所包含接口如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Width | Direction | Description |
| clk\_in | 1 | Input | 时钟输入，典型值为100Kbit/s |
| rst\_n | 1 | Input | 复位信号输入，低电平有效 |
| scl | 1 | output | I2C总线时钟信号Serial Clock |
| sda | 1 | Inout | I2C总线数据信号Serial Data |
| i2c\_read\_data | 8 | Output | I2C总线读出数据 |
| i2c\_device\_address | 7 | Input | I2C从机设备地址，7bits |
| i2c\_reg\_address | 8 | Input | I2C目标寄存器地址 |
| i2c\_write\_reg\_data | 8 | Input | I2C写入目标寄存器数据 |
| i2c\_config | 8 | Input | 配置I2C操作模式信号 |
| write\_done | 1 | Output | 写完成标志位 |
| read\_done | 1 | Output | 读完成标志位 |
| state\_debug | 8 | Output | 状态指示寄存器 |
| i2c\_ack | 8 | Output | 从机响应/主机响应位 |

4.1.1 clk\_in

时钟输入信号，该信号直接影响I2C总线工作频率，典型输入频率为100Kbit/s，可使I2C总线工作在100K标准模式下，如果从机设备支持快速或高速模式，该时钟频率可响应输入更高频率，如400Kbit/s或者3.4Mbit/s。

4.1.2 rst\_n

复位输入信号，该信号为被拉低时，电路进入复位状态。

4.1.3 scl & sda

I2C通讯总线，硬件电路需配置为上拉，管脚约束建议同样调整为为上拉。

4.1.4 i2c\_read\_data

该寄存器用于存储主机在从机中读取到的数据。

4.1.5 i2c\_device\_address

该信号为从机设备地址，暂只支持7位的设备地址。

4.1.6 i2c\_reg\_address

该信号为读/写操作目标寄存器的地址，由外部输入。

4.1.7 i2c\_write\_reg\_data

该信号为I2C写入目标寄存器数据，由外部输入。

4.1.8 i2c\_config

该信号为配置I2C工作模式的外部输入信号，本I2C Master IP核支持如下工作模式，分别对应输入信号为：

·i2c\_config= 8`h00

I2C挂起，进入等待模式(WAIT)，等待状态下主机将SCL拉高并释放SDA总线。

·i2c\_config= 8`h01

I2C单次写入模式(I2C\_Single\_Write\_Byte)，标准1Byte数据写入模式。

·i2c\_config= 8`h02

I2C连续写入模式(I2C\_Continuous\_Write\_Byte)，主机对从机目标寄存器进行连续写入1Byte的数据，当主机发送到从机的1Byte数据并接受到ACK信号时，不会停止I2C总线，而是继续写入1Byte数据。该模式不会主动停止。

·i2c\_config= 8`h03

I2C直接写入状态(I2C\_Write\_Directly)，主机直接对从机设备进行1Byte的数据写入，即成功访问到设备并接受到ACK信号之后，直接写入8bit的数据即可。

·i2c\_config= 8`h04

I2C单次读取状态(I2C\_Single\_Read\_Byte)，标准1Byte数据读取模式。

·i2c\_config= 8`h05

I2C连续读取状态(I2C\_Continuous\_Read\_Byte)，主机对从机目标寄存器进行连续的数据读取操作，即在通讯的过程中，主机成功读取从机目标寄存器数据后，不会发送NACK信号，而是发送ACK信号并再次读取从机目标寄存器的数据。

·i2c\_config= 8`h06

I2C直接读取状态(I2C\_Read\_Directly)，主机对从机设备进行直接读取数据操作，即成功访问到设备并接受到ACK信号后，直接再次进行START模式，进行数据的读取。

4.1.9 write\_done

写入完成并接收到从机响应后，该信号发出低电平脉冲，该信号在连续写入模式下无效。

4.1.10 read\_done

读取完成并接受到从机响应后，该信号发出低电平脉冲，该信号在连续读取模式下无效。

4.1.11 state\_debug

该信号为程序运行状态指示寄存器。

4.1.12 i2c\_ack

该信号为响应指示信号，对应关系如下：

i2c\_ack[0] : 写入设备地址从机响应位

i2c\_ack[1] : 写入寄存器地址从机响应位

i2c\_ack[2] : 写入寄存器数据从机响应位

i2c\_ack[3] : 读取寄存器地址从机响应位

i2c\_ack[4] : 成功读取寄存数据后主机发送ACK

i2c\_ack[5] : 成功读取寄存器数据后主机发送NACK

i2c\_ack[4] : 预留，默认值为0

i2c\_ack[5] : 预留，默认值为0

4.2 i2c\_master\_config module

i2c\_master\_config module是主要功能为对I2C运行模式及寄存器地址、数据进行配置。相关寄存器与i2c\_master\_logic module相同，在此不在赘述。

4.3 clk\_div module

本module为时钟分频模块，通过对系统时钟信号进行分频可生成I2C所需时钟，你可以点击此处[了解](http://www.stepfpga.com/doc/5._%E6%97%B6%E9%92%9F%E5%88%86%E9%A2%91)该clk\_div module的详细说明。

1. **I2C运行模式**

本I2C\_Master IP可以配置为多种操作模式（详见3.1.8小节），对应状态跳变如下所示。你可以在i2c\_master\_config module中拉取i2c\_done与i2c\_ack标志位信号实现状态跳变；此外，你也可以更改i2c\_master\_logic中关于状态跳变部分的代码，实现其它变种I2C的通讯模式。

5.1 I2C单次写入模式(I2C\_Single\_Write\_Byte)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | 寄存器地址 | ACK | 寄存器数据 | ACK | STOP |

5.2 I2C单次读取模式(I2C\_Single\_Write\_Byte)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | 寄存器地址 | ACK | START2 | 设备地址+读 | ACK |
| 读取数据 | NACK | STOP |

5.3 I2C连续写入模式(I2C\_Continuous\_Write\_Byte)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | 寄存器地址 | ACK | 寄存器数据 | ACK | 寄存器数据 |
| ACK | 寄存器数据 | ACK | ··· | | | | |

5.4 I2C连续读取模式(I2C\_Continuous\_Read\_Byte)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | 寄存器地址 | | ACK | | START2 | | 设备地址+读 | ACK |
| 读取数据 | ACK | 读取数据 | ACK | 读取数据 | | ACK | | 读取数据 | | ··· |

5.5 I2C直接写入模式(I2C\_Write\_Directly)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | 寄存器数据 | ACK | STOP |

5.6 I2C直接读取模式(I2C\_Read\_Directly)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 起始信号 | 设备地址+写 | ACK | START2 | 设备地址+读 | ACK | 读取数据 |
| NACK | STOP |

5.7 等待状态（WAIT）

等待状态下，SCL置1，SDA被释放。

注：

1. 当访问从机设备地址没有响应时，程序将重新访问从机设备地址，直至设备响应。
2. 除访问设备地址的ACK信号外，从机没有反馈ACK信号，程序将停止运行，此时可调取state\_dubug或i2c\_ack信号进行调试。
3. **应用示例&相关信号说明**

6.1 配置流程说明

I2C Master Core的配置流程主要分为3步：

1. 调用通讯过程中涉及到的i2c\_ack信号并进行逻辑处理，作为状态跳变的触发条件；
2. 根据实际需求，描述状态跳变；
3. 在不同的状态中，配置i2c设备地址、设备寄存器地址与所写入的数据/存储读取的数据。

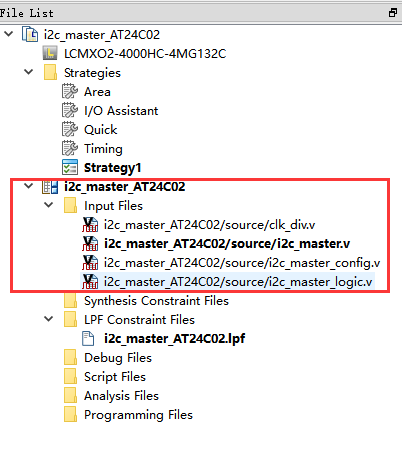
在本小节中，会以操作AT24C02为例，详细介绍的配置方法。详细实现操作AT24C02操作如下：

·在AT24C02的00寄存器中写入8`hBB。

·读取AT24C02中00寄存器的数据。

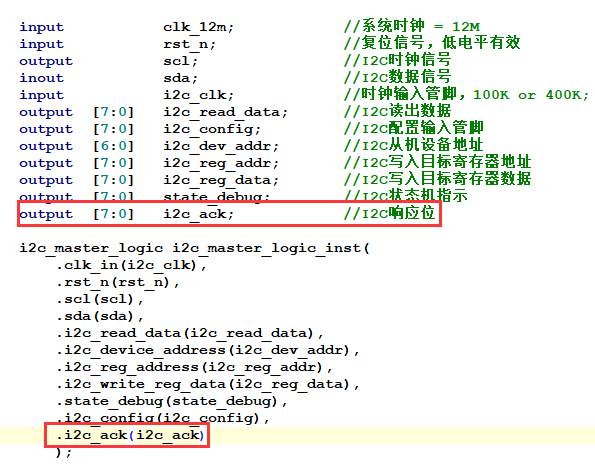
实现流程:

首先，我们创建一个新工程，添加I2C\_Master IP的Verilog文件进入工程之中：



根据需求，需要进行一次写操作、一次读操作、之后停止I2C通讯，所以我们需要一个写完成的信号(可用写入完成标志位i2c\_ack[2]上升沿作为触发，详见6.2小节)，一个单次读完成的信号(i2c\_ack[5]，主机发送NACK标志位)，作为状态变化的触发。

打开i2c\_master\_config，找到i2c\_ack信号：



新建一个名为i2c\_flag的线网型变量，该信号等于（i2c\_ack[2]|i2c\_ack[5]）:

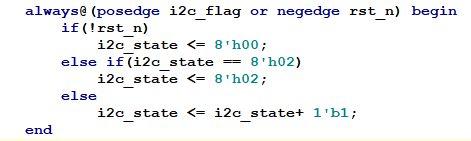


这样，我们就实现了一个触发信号，之后再利用这个触发信号实现状态跳变，根据需求，程序可以分位三个状态，第一个状态为写入状态，第二个状态为读取状态，第三个状态为总线挂起。实现方式如下;

新建一个寄存器，位宽要足够容纳下总状态数:



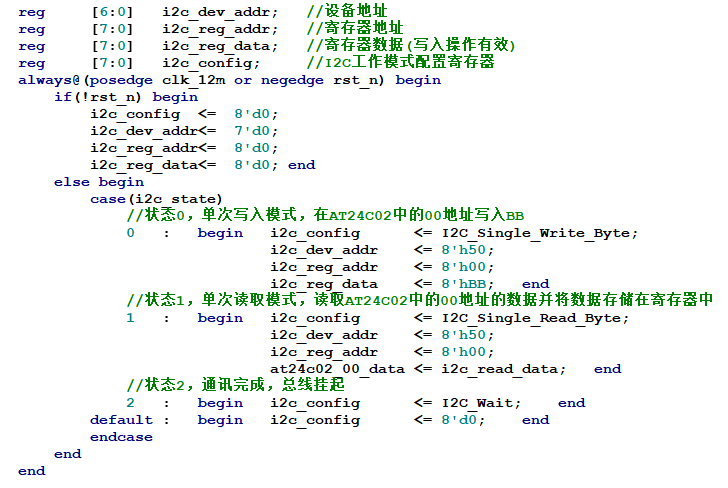
描述三个状态的转移关系:



这样我们就实现了写入-读取-挂起的状态跳变，之后，我们建立一个8位的寄存器，用于存储读取出的数据，寄存器命名为at24c02\_00\_data;



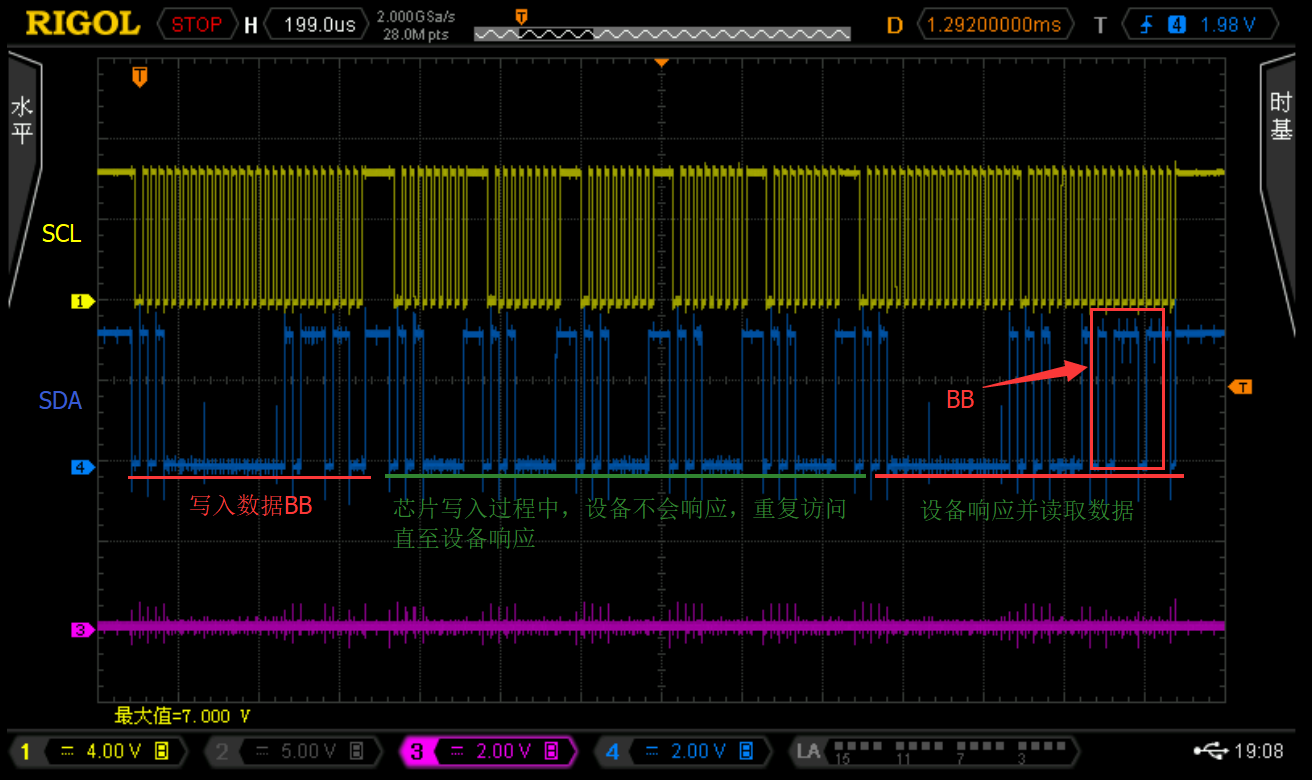
接着， 我们在对应的状态中配置i2c的运行模式，设备地址，寄存器地址，写入的数据等参数，并保存读取的数据至at24c02\_00\_data寄存器中：



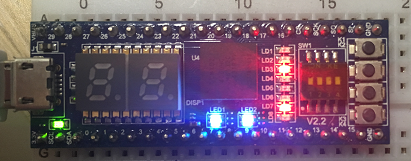
这样，我们就完成了对AT24C02的写入、读取、和I2C的挂起操作，之后，还要将I2C总线配置为内部上拉模式：



综合工程并下载，抓取I2C通讯的波形如下:



将at24c02\_00\_data寄存器输出至LED上，读出内容为“10111011”，即写入的“BB”。



你可以在“i2c\_master/source/i2c\_master\_AT24C02”下找到该工程。

6.2 连续读写(需要器件支持)配置方式应用示例

连续读写通常需要从机支持，对于连续读写操作来说，在连续读或者连续写的过程中，不会返回i2c\_done的完成脉冲信号，你需要拉取i2c\_ack寄存器对应位的信号作为标志，来控制状态跳变，本小节以PCF8591为例，实现将存储在Rom中的数据转化为电压值进行输出，相关参数如下：

测试平台：STEP Baseboard

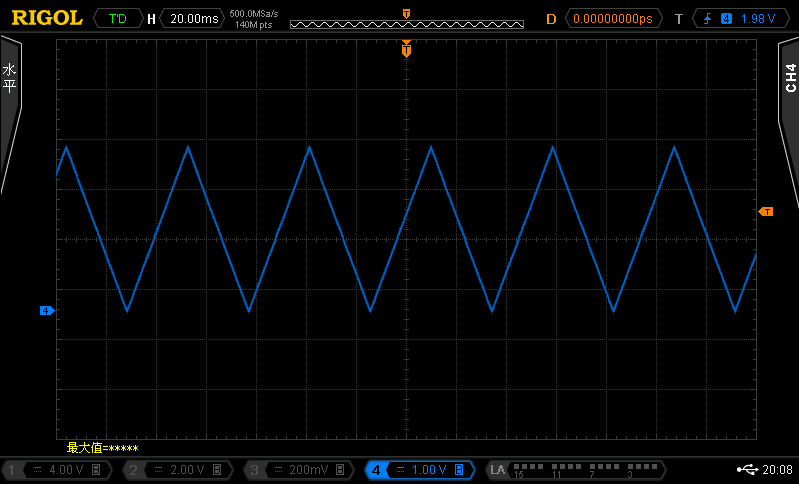
目标设备地址(i2c\_dev\_addr) = 8`h48；

目标寄存器地址(i2c\_reg\_addr) = 8`h00;

写入数据(i2c\_reg\_data) = rom\_data

你可以在“i2c\_master/source/i2c\_master\_PCF8591\_DAC”目录中找到该工程。

运行该程序，量取板卡DAC输出管脚，可查看到输出波形如下：



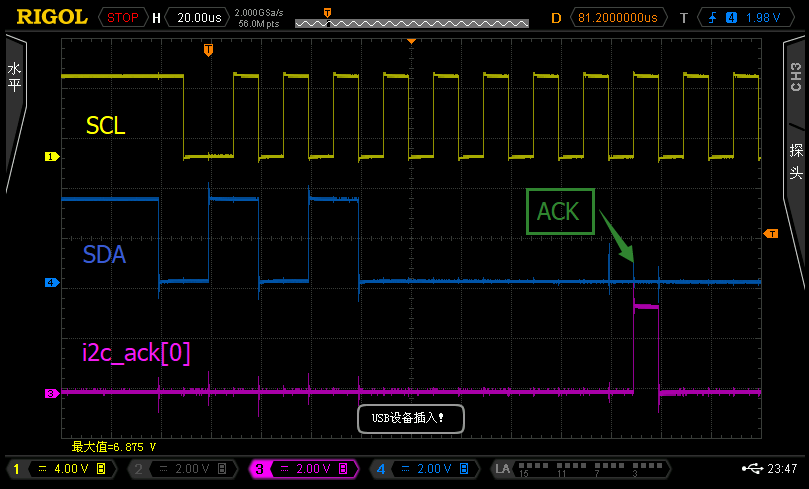
6.3 直接读写(需要器件支持)配置方式应用示例

某些从机设备需要在访问设备地址后直接对寄存器进行读写操作，你可以在“i2c\_master/source/i2c\_master\_SHT20”目录中找到该工程。

6.4 i2c\_ack信号

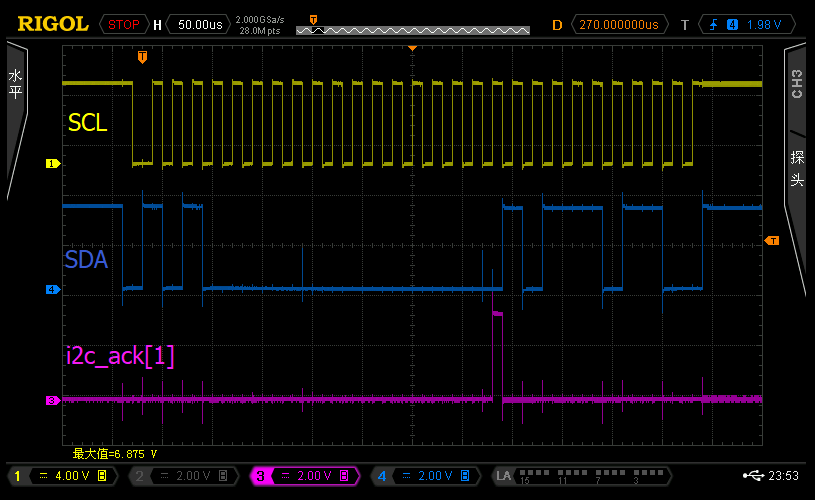
6.4.1 i2c\_ack[0] - 写入设备地址从机响应位

i2c\_ack[0]在设备响应之后，会发送一个高电平脉冲，信号如下：

、

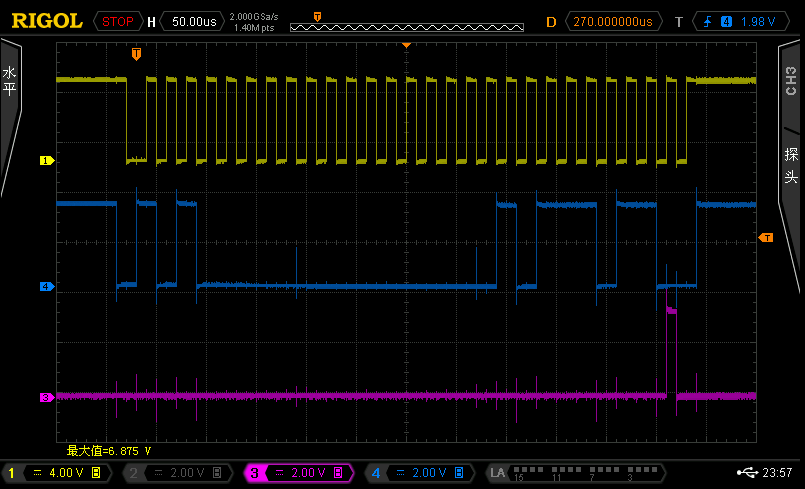
6.4.2 i2c\_ack[1] - 写入寄存器地址从机响应位

i2c\_ack[1]在写入从机设备的寄存器地址并收到响应之后，会发送一个高电平脉冲，信号如下：



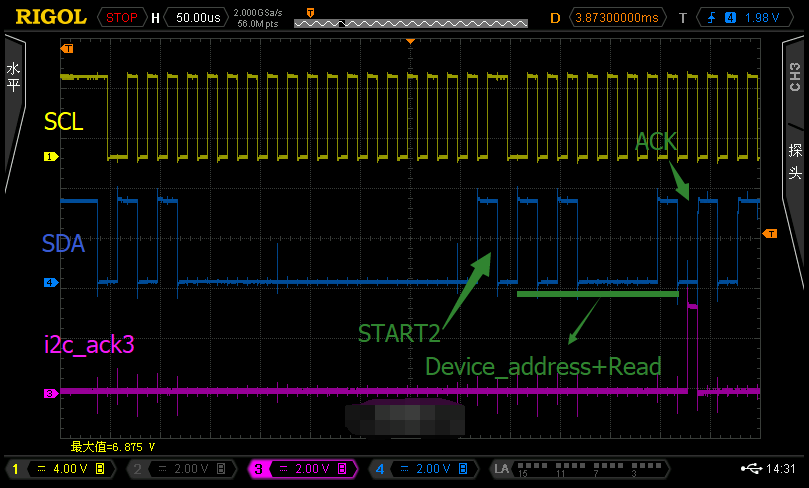
6.4.3 i2c\_ack[2] – 写入寄存器数据从机响应位

i2c\_ack[2]在写入从机设备的寄存器数据并收到响应之后，会发送一个高电平脉冲，信号如下：



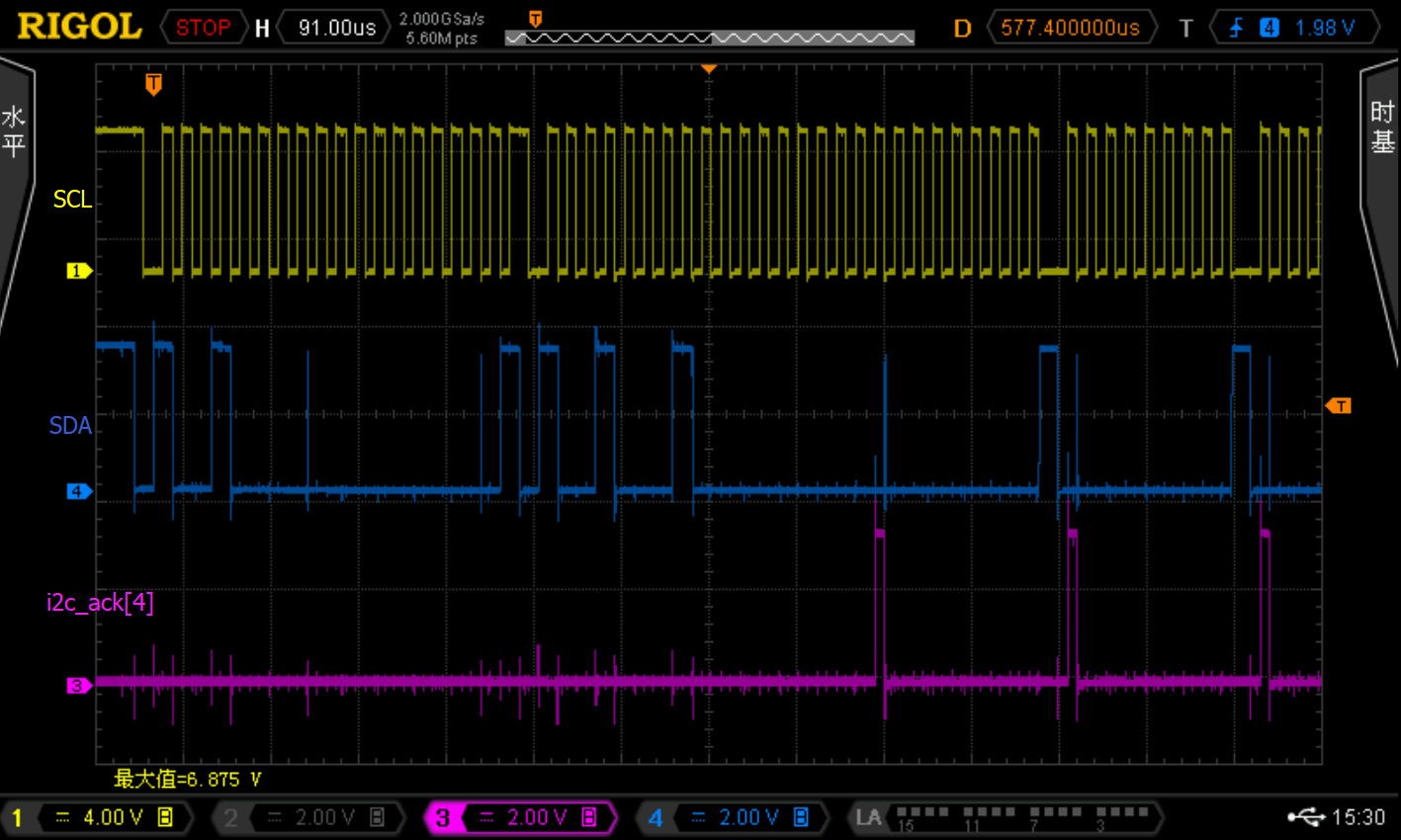
6.4.4 i2c\_ack[3] – 设备地址+读从机响应位

i2c\_ack[3]在读取操作过程中，写入从机设备地址+读并收到响应信号后，会发送一个高电平脉冲，信号如下：



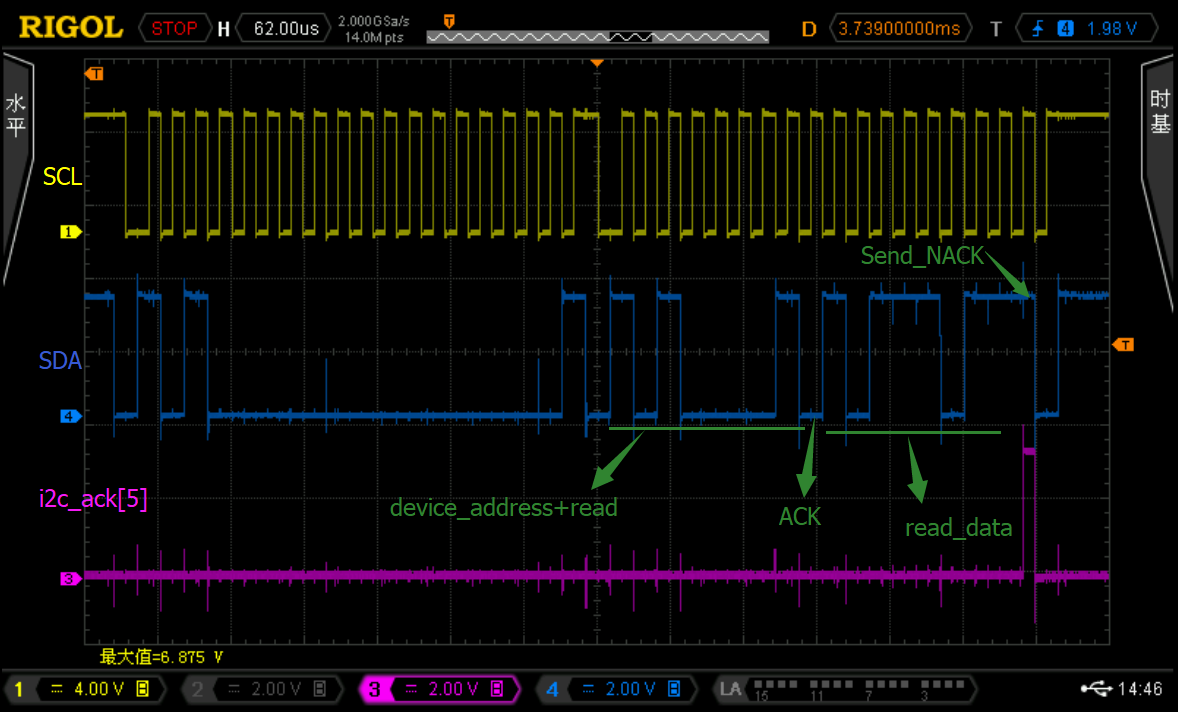
6.4.5 i2c\_ack[4] **– 主机发送ACK标志位**

i2c\_ack[4]在连续读取操作过程中，，每次成功读取数据后，会发送一个高电平脉冲，信号如下：



6.4.6 i2c\_ack[5] **– 主机发送NACK标志位**

i2c\_ack[5]在单词读取操作过程中，成功读取数据后，主机发送NACK信号，i2c\_ack[5]会发送一个高电平脉冲，信号如下：



1. 版本说明

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本号 | 修订人 | 改动 |
| 2017.12.16 | V0.0 | 张泽 | 最初版本 |