Body Area Network

Demo Test Report

haobo.gao

April 25, 2019

Foxconn ZZDC

Introduce

RS485 feature

- 485 is Fully balanced
- 485 handles multiple drivers and receivers
- Better common-mode noise rejection (-7 to +12Volts)
- Sensitivity of ś200mV in receivers
- Drivers give up to 5 volts balanced output
- Can stand contention, driver shuts down by itself
- High input resistance (12K ohms)
- Hysteresis of 50 mv to overcome diff. noise

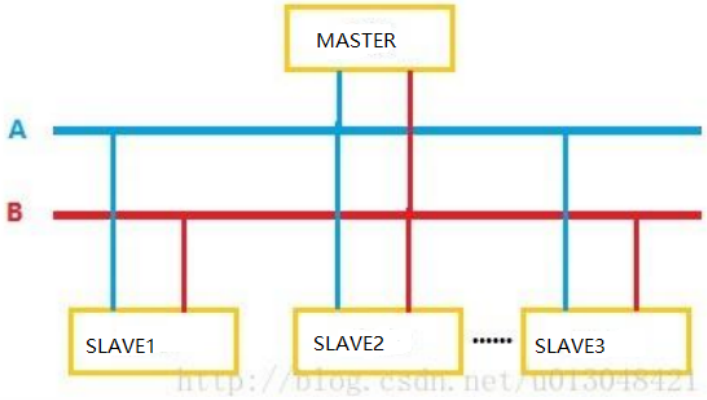


Figure: 485NET

The Principle of The BAN's Clock Synchronization

- In order to synchronize the device clock, it is necessary to add a clock line between Master and Slave, where Master generates synchronous clock square wave.

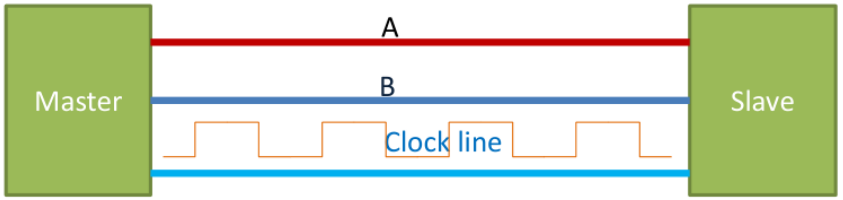


Figure: synchronous

Our Design

We designed it like this：

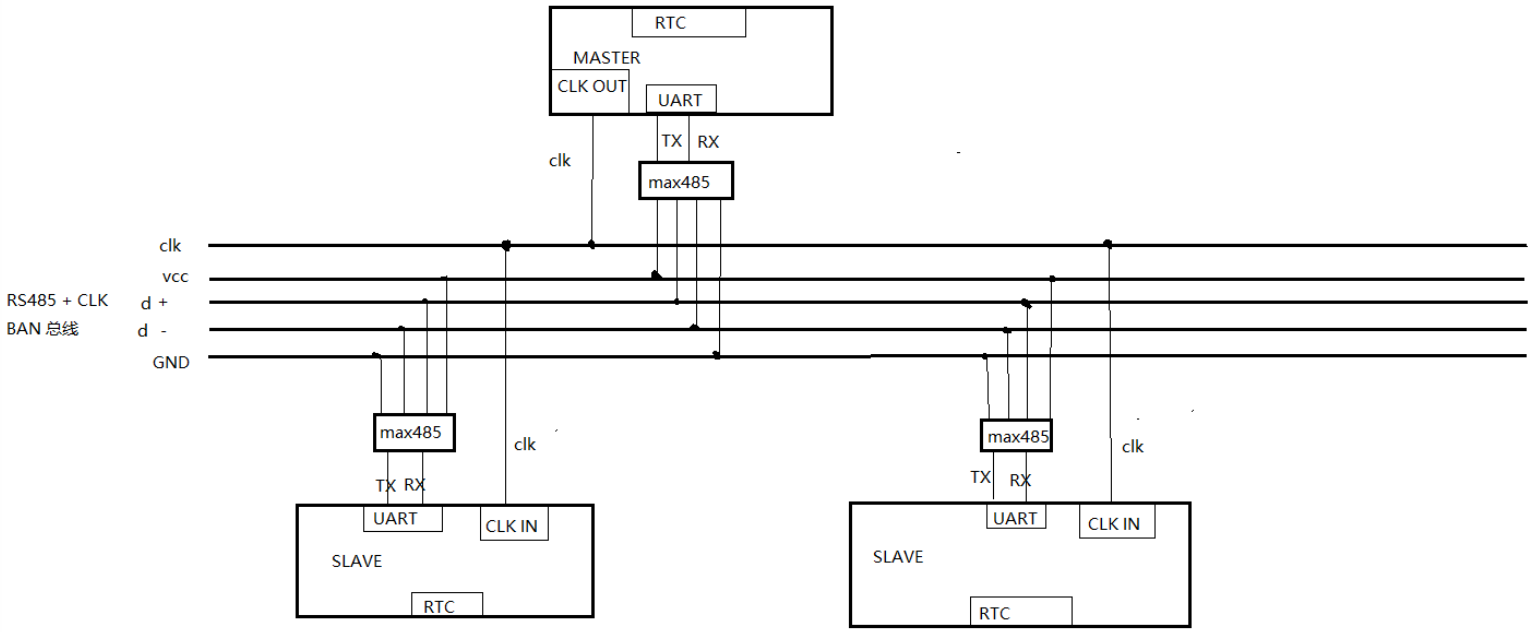


Figure: BAN

BUS

1. Adding a clock line from RS485 to form the basic hardware circuit of BAN bus.
2. The clock line is used to generate Millisecond-level pulses for clock synchronization.Later aliases:SYNC_CLK
3. D + D - is a data signal after differential processing.

figure6 assumed that the device mounted on the bus has one Master and two Slaves.

Local Clock Counter & SYNC_CLK

- Local Clock Counter , Each device has a local clock counter. The timing granularity is 100us.
- SYNC_CLK,The device on BAN bus uses clock line to complete clock synchronization. Master's timer generates synchronization pulse, Slave's timer captures synchronization pulse to complete synchronization.

Synchronization process

- a. master and slave power on and initialize peripherals.
- b. master starts sending square wave signals,at The first rising edge of square wave signal:
  1. The master starts Master's Local Clock Counter(MLCC).
  2. Slave captures this rising edge at the same time.Then start Slave's Local Clock Counter(SLCC).
  3. The delay of the rising edge of the clock line transmission can be neglected, so we think that MLLC and SLLC start counting at the same time.
- c. Slave calibrates SLCC for each subsequent SYNC_CLK rising edge.

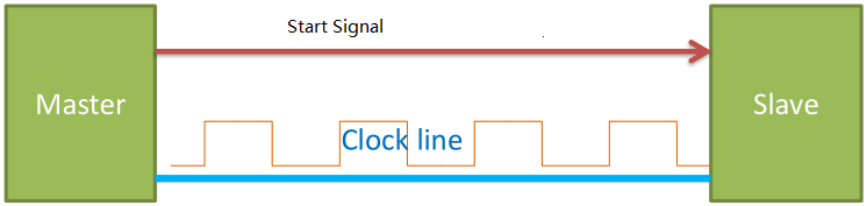


Figure: BAN

Clock Difference Calibration

SYNC_CLK is produced by Master with fixed period and precise time.Let's take a period of 16 milliseconds for example.

MLLC and SLLC start counter at the sametime(first SYNC_CLK rising edge). Their timing granularity is small, 100 microseconds.

So when the system runs, there are two counts on each device, the Local Clock Count(Every 100 microseconds plus 1) and the SYNC_CLK Count(Every 16 milliseconds plus 160).Slave calibrates SLCC at each subsequent SYNC_CLK rising edge by Make those two counts the same.

Clock Difference Calibration

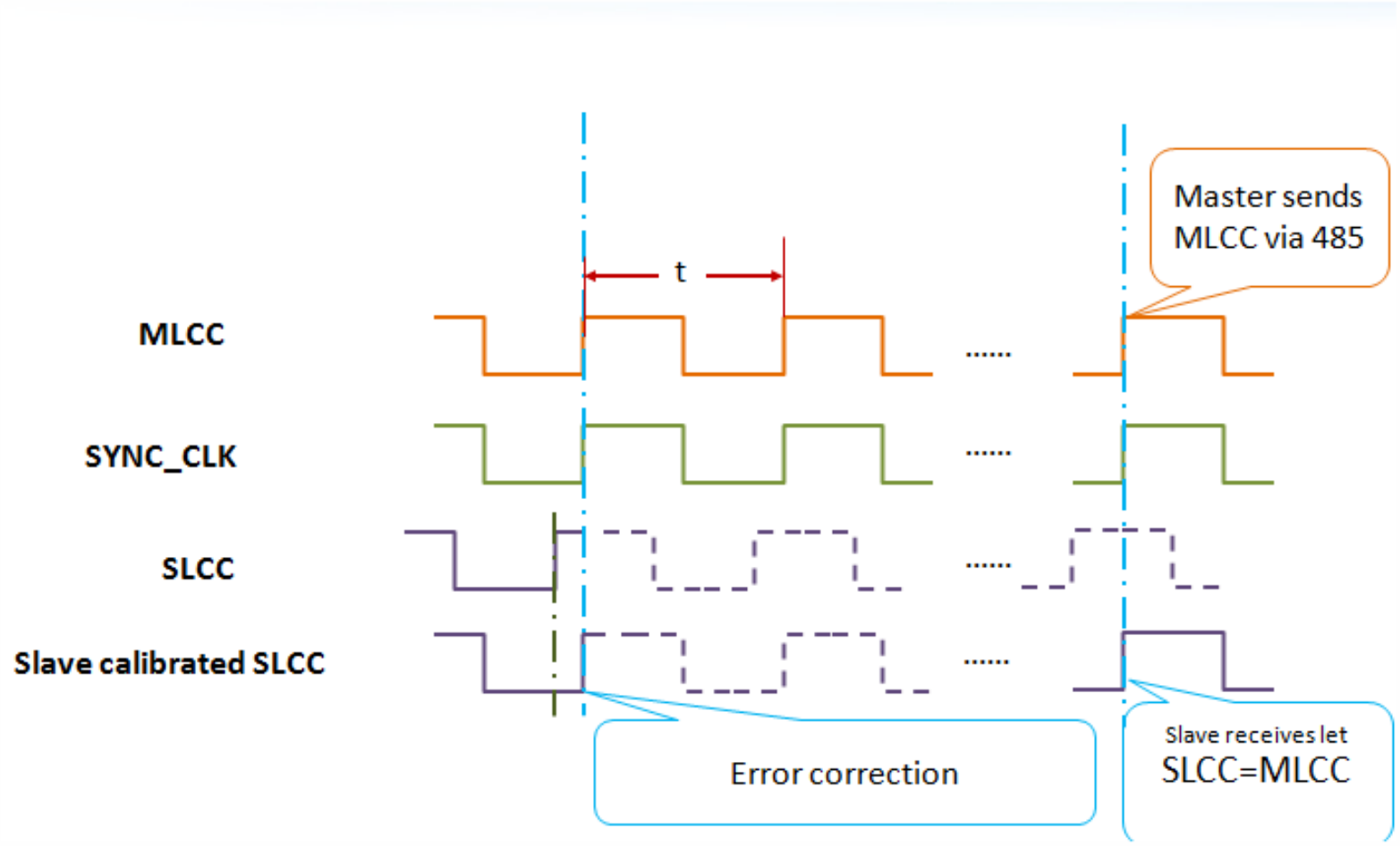


Figure: BAN

Demo implementation

The following is the demo wiring and structure：

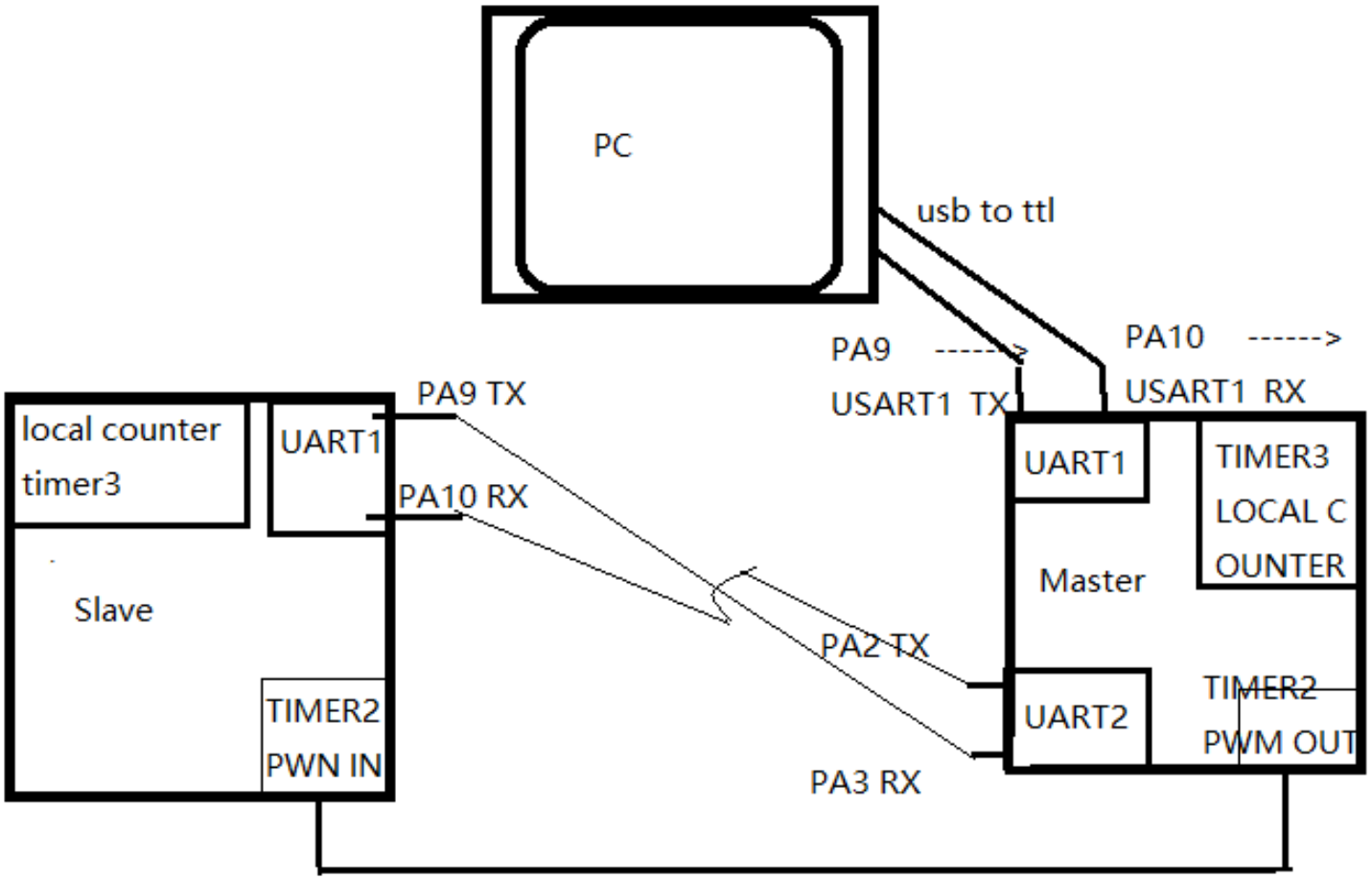


Figure: demo

workflow

Demo workflow: two instructions

1. Master and Slave Power on。Peripheral Driver Initialization,Slave enters the rising edge capture state and waits for an rising edge coming.
2. When Master receives "begin" instruction from PC.
   - a.Forward to Slave first.
   - b. start timer3 for MLCC(Master Local Clock Counter).
   - c. start timer2 to Generating Pulse Square Wave for synchronous。
3. When Slave receives the "begin" instruction from Master, and Slave captures the first rising edge of SYNC_CLK:
   - a. start timer3 for SLCC(Slave Local Clock Counter).
   - b. Continue to capture the rising edge.
4. "report" instructions:
   - a.when Master receives the "report" instruction and reports its own timestamp(MLCC) to PC.
   - b.when Slave receives the "report" instruction, reports its own timestamp(SLCC) to Master, and Master forwards it to PC.

Demo Workflow: Synchronize details

local_tm_t is the type of time described in Master and Slave code:

```
typedef struct local_time_struct{
 unsigned char sec:7;    //second
 unsigned char min:7;    //min
 unsigned char hour:5;   //hour
 unsigned int  day;      //day
 unsigned int  jiffies; //LCC
 unsigned int  jiffies_pwm;    //SYNC_CLK counter
} local_tm_t;
```

Slave Rising Edge Capture Interrupt

tick_sync is called every 8ms to synchronize the local timestamp:

```
/* every 8ms 80*100us  */
void tick_sync(void)
{
  if( dev.local_tm.jiffies_pwm + 80 < 10000  ){
  dev.local_tm.jiffies_pwm = dev.local_tm.jiffies_pwm  +
        80;
  }
  if( dev.local_tm.jiffies_pwm + 80 >= 10000  ){
    dev.evt.sec_need_update=1;
    dev.local_tm.jiffies_pwm = 80 - (10000 - dev.local_tm.
          jiffies_pwm);
  }
  dev.local_tm.jiffies = dev.local_tm.jiffies_pwm;^^I^^I//
        synchronous
}
```

Synchronize details:test point

Enter this code every 100us:

```
/*
 enter this code  every 100us。
*/
void Tick(void)
{
  dev.local_tm.jiffies ++;
  if( dev.local_tm.jiffies % 80 == 0){
   //GPIO A5 for compare.
   HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,1-HAL_GPIO_ReadPin(
          GPIOA,GPIO_PIN_5));
 }
  /* one second passed */
  if(dev.local_tm.jiffies >= 10000 ){
   dev.local_tm.jiffies  = 0;
 }
}
```

Here the slave GPIOA5 has square wave output relative to the clock synchronization line. It's a test point.

TEST Data

Time stamp of serial port printing

After the "report" instruction is sent to the Master, the following code segments are executed:

```
/*
 report the jiffes to the  PC
*/
void report_jiffes(void)
{
  printf("master rj:\t%d\r\n", dev.local_tm.jiffies);
  printf("master rj:\t%d\r\n", dev.local_tm.jiffies);
  printf("master rj_PWM:\t%d\r\n", dev.local_tm.
         jiffies_pwm);
  printf("master rj_PWM:\t%d\r\n", dev.local_tm.
         jiffies_pwm);
 }
```

This code reports the values of jiffies (**local time count**) and jiffies PWM **sync time count** to the PC.

Time stamp of serial port printing

```
MASTER 0 0:0:50   jiffes:0
MASTER 0 0:0:51   jiffes:0
MASTER 0 0:0:52   jiffes:0
MASTER 0 0:0:53   jiffes:0
MASTER 0 0:0:54   jiffes:0
MASTER 0 0:0:55   jiffes:0
MASTER 0 0:0:56   jiffes:0
master rj:       6119
master rj:       6126
master rj_PWM:   6080
master rj_PWM:   6080
slave:rj:80

MASTER 0 0:0:57   jiffes:0
MASTER 0 0:0:58   jiffes:0
```

Figure: report jiffies

After entering the "report" command, output is as shown above (in red circles), combined with the previous code, you can see that:

- In order to print log in the above code, the execution time of printf is about (6126-6119=7)*100us.Here, the printf is a non-blocking, lightweight, full-featured printf for transplantation.
- Slave and Master have transmission delays.

Waveform: Unsynchronized

Let:

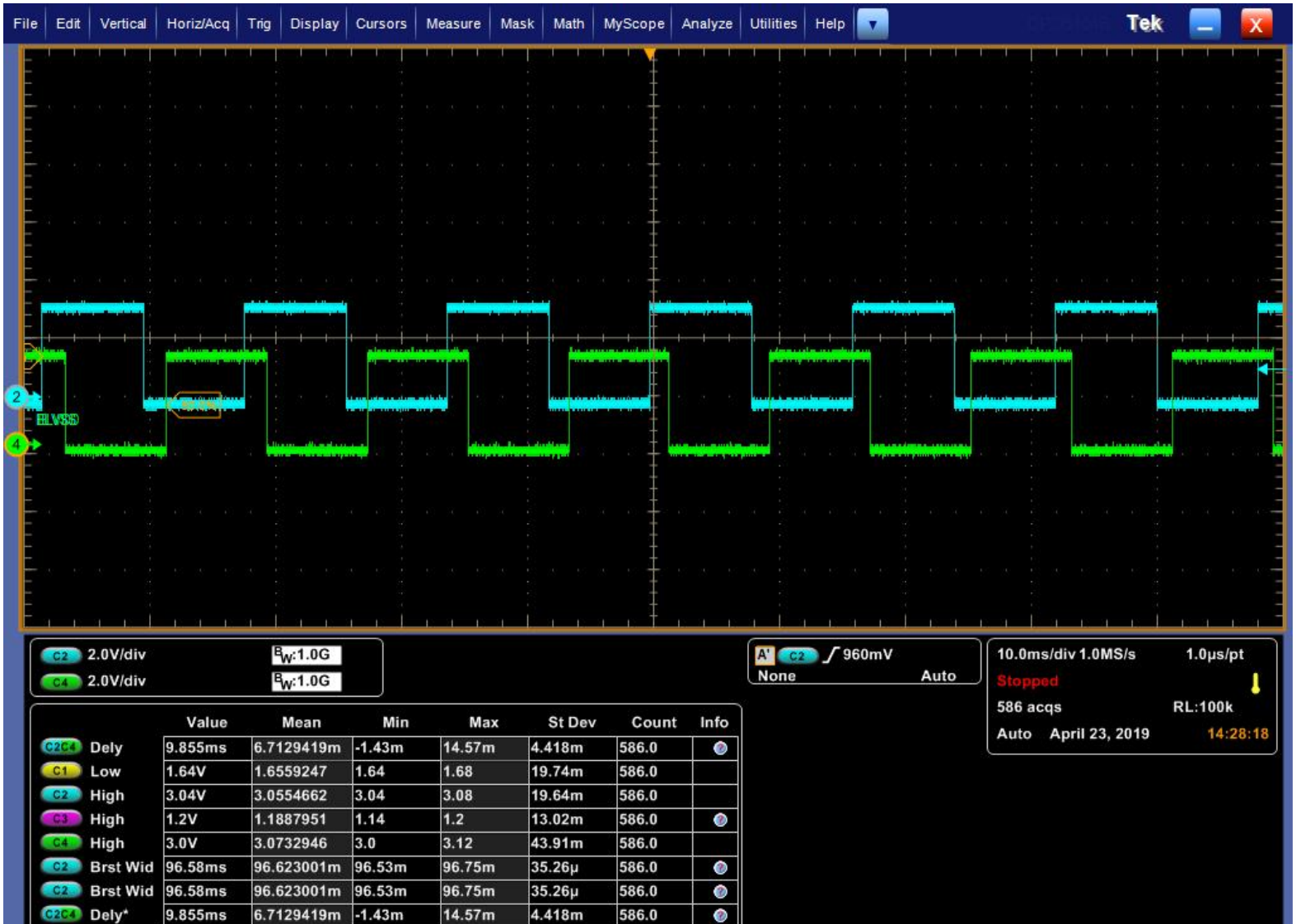
- The local clock inverts the GPIOA5 level state every 8ms. This test channel is marked CH_LOCAL.
- The synchronous line generates a synchronous square wave with a half period of 8ms from the Master. This test channel is marked CH_SYNC.
- In the interrupt processing function of the slave to capture the rising edge of the synchronization line, the clock is not synchronized, that is to say, the following code is commented out:

```
//dev.local_tm.jiffies = dev.local_tm.jiffies_pwm;^^I
        ^^I//synchronize code
```

The following screenshots show the unsynchronized screenshots:

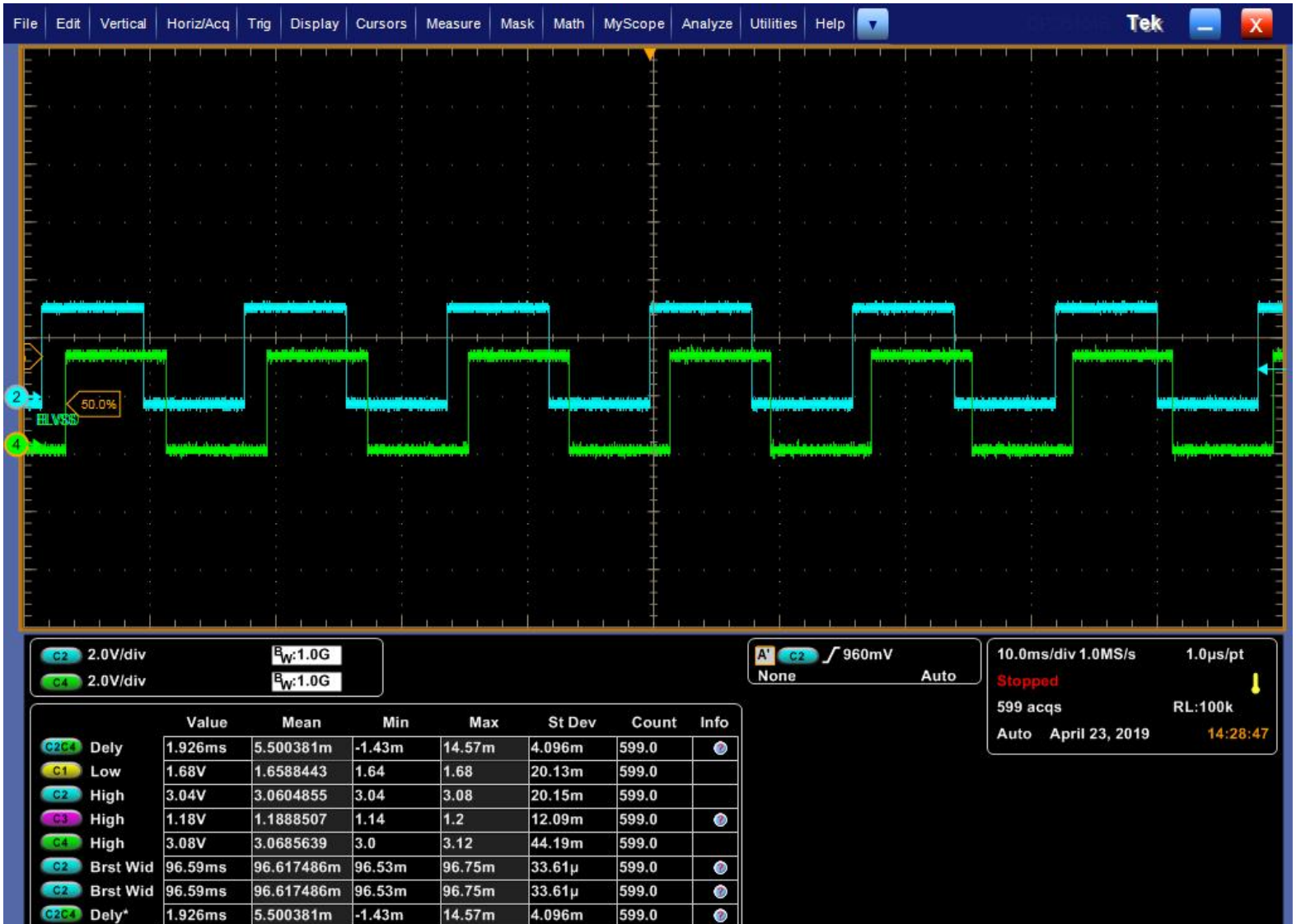
Waveform: Unsynchronized

Unsynchronized waveform 1:



Waveform: Unsynchronized

Unsynchronized waveform 2:

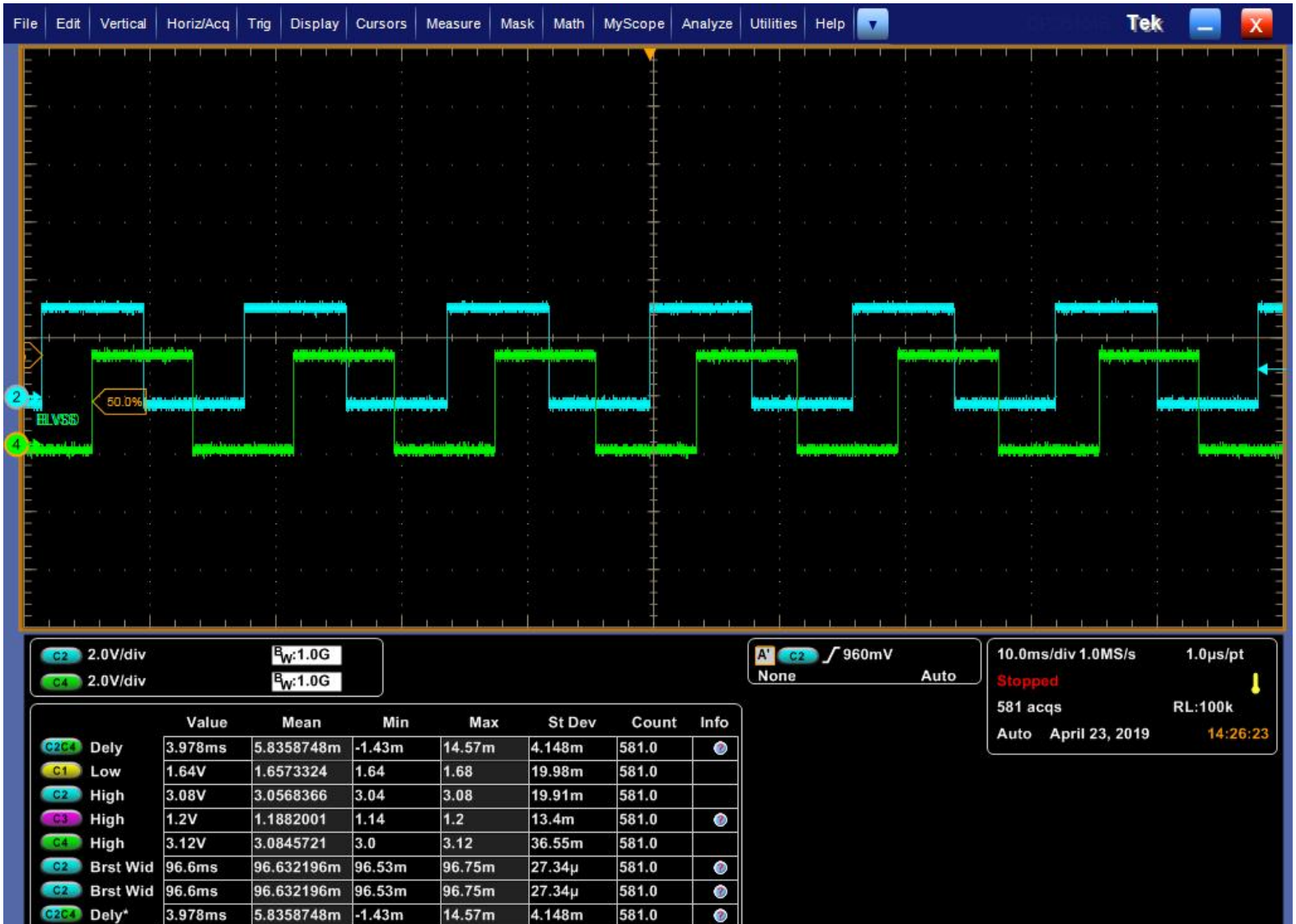


Waveform: Unsynchronized

Unsynchronized waveform 3:

Waveform: Unsynchronized

Unsynchronized waveform 4:



Some explanations of unsynchronized waveforms

Return this document to unsynchronized waveform 1 (17) and turn the page with the right key of the keyboard direction key (from unsynchronized waveform 1 to unsynchronized waveform 4). At this time, the animation effect on the screen can restore the appearance of the oscilloscope at that time.

Because the local clock is not synchronized, each difference accumulates, resulting in such a dynamic effect.

Waveform: Synchronization

Let:

- The local clock inverts the GPIOA5 level state every 8ms. This test channel is marked CH_LOCAL.
- The synchronous line generates a synchronous square wave with a half period of 8ms from the Master. This test channel is marked CH_SYNC.
- In the interrupt processing function of the slave capturing the rising edge of the synchronization line, the clock is synchronized, that is to say, the following code is active:

```
dev.local_tm.jiffies = dev.local_tm.jiffies_pwm;^^I^^I
        //synchronize code
```

The following screenshots show the screenshots in the case of synchronization.

Waveform: Synchronization

synchronized waveform :

Waveform: Synchronization

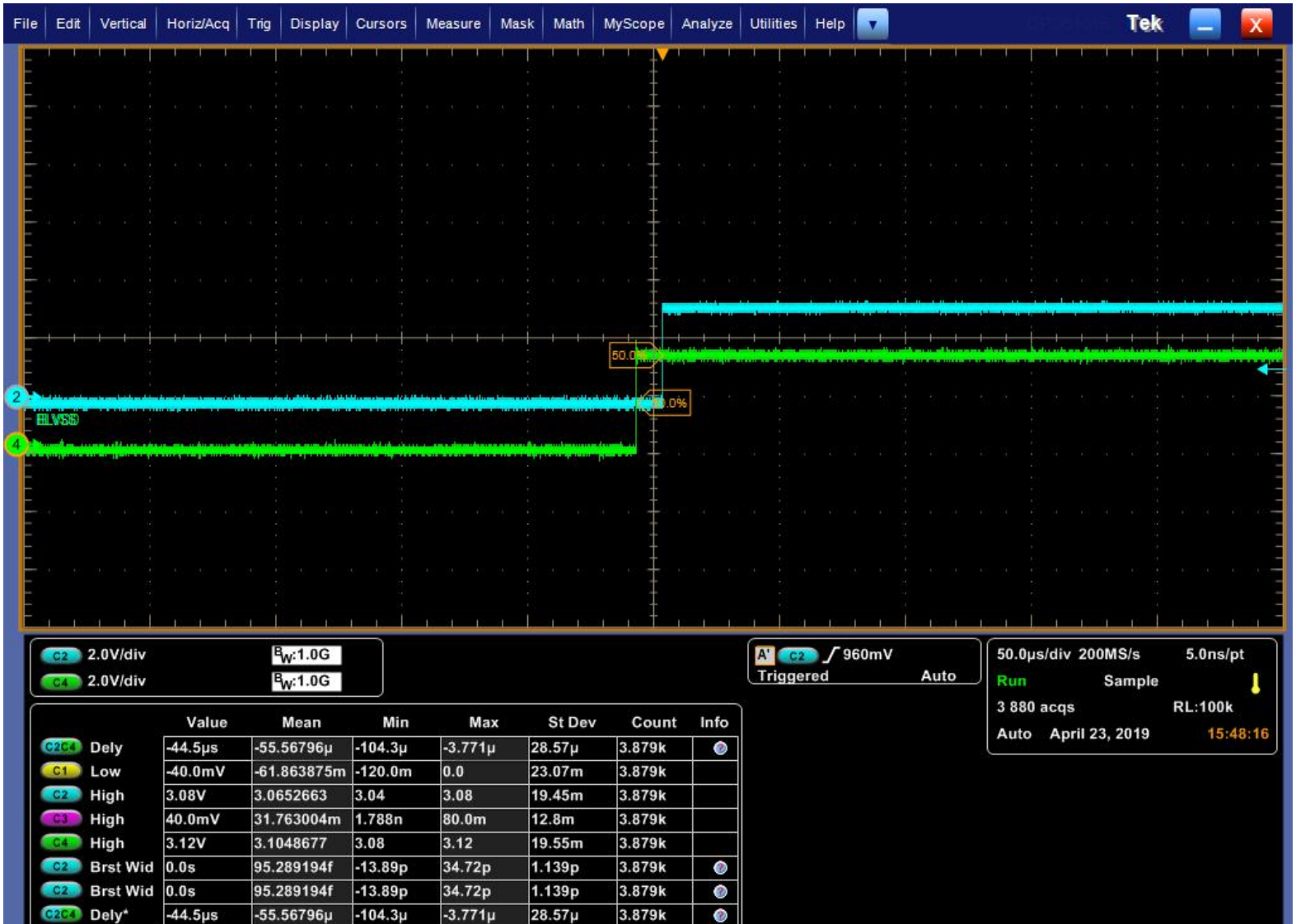
Since the rising edge of each synchronization line is captured by slave computer, synchronization interrupt will be triggered. In interrupt processing, the local clock error of each cycle will be eliminated.

So the synchronized waveform looks fixed and the phase is relatively consistent. But:

If we amplify the size of the oscilloscope, we will find that there are difference. The following are some synchronization delay. The size of the oscilloscope is already 50us. We are concerned about the synchronization time.

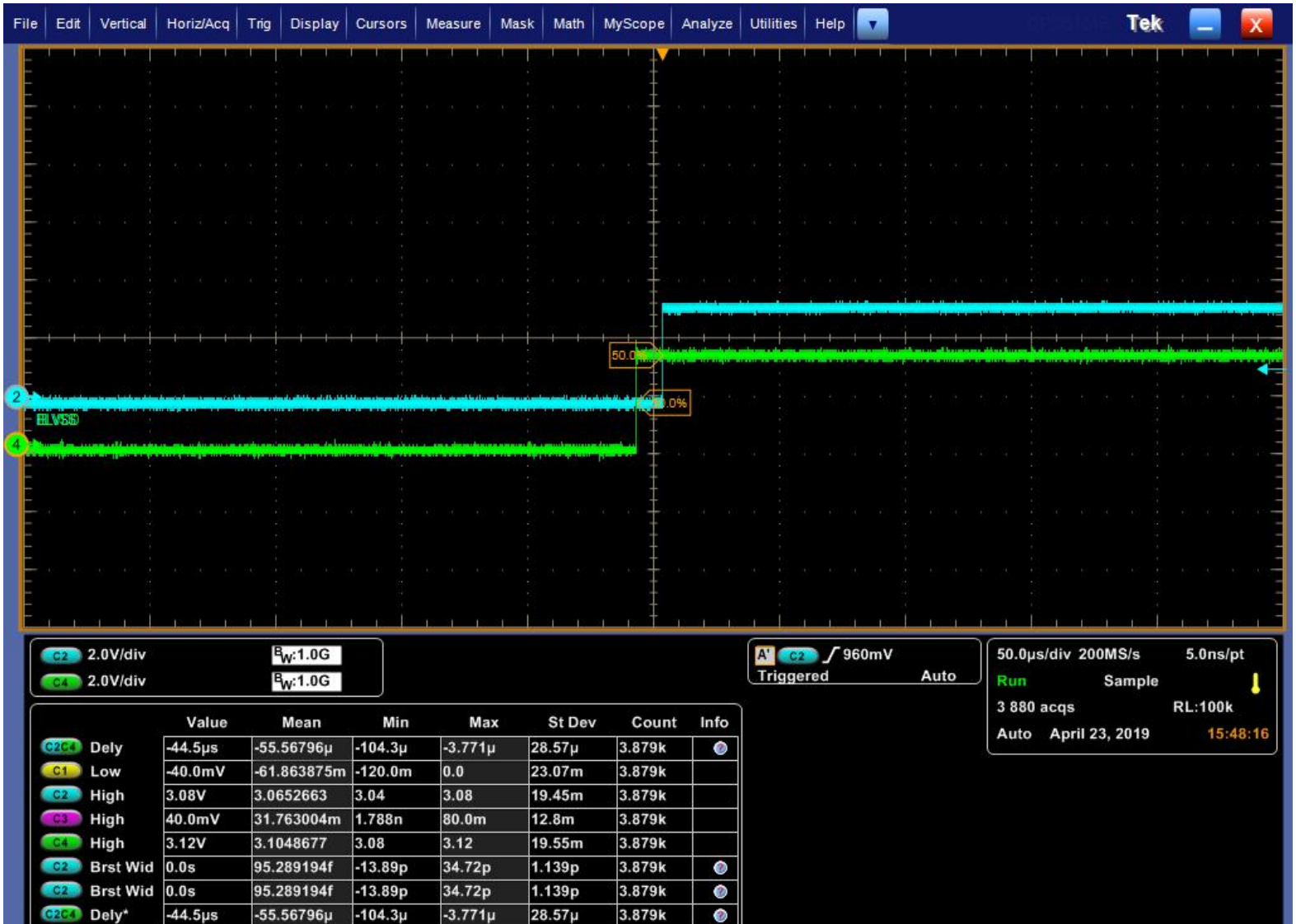
Waveform: Synchronization

Synchronization delay:



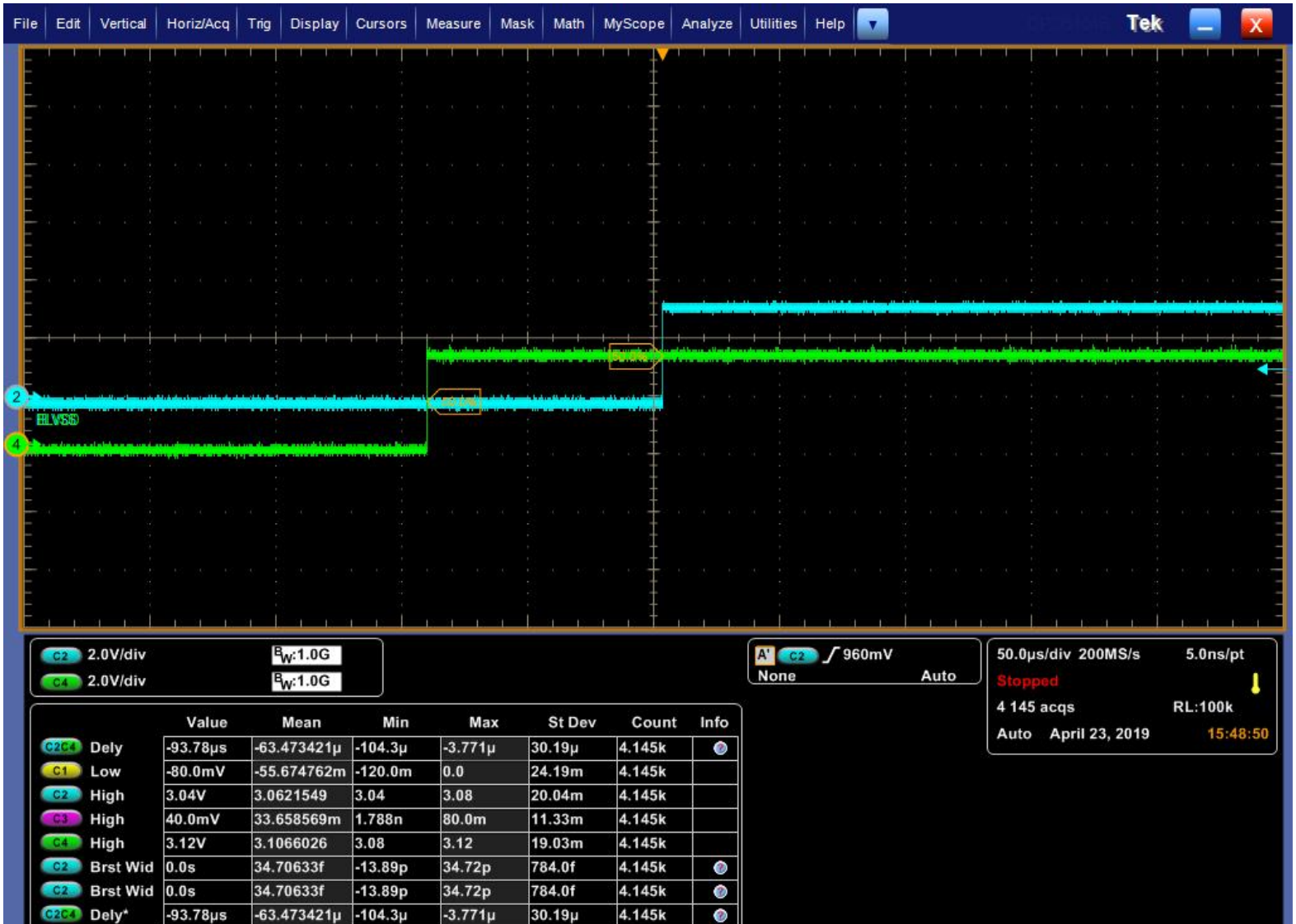
Synchronization delay

Synchronization delay:



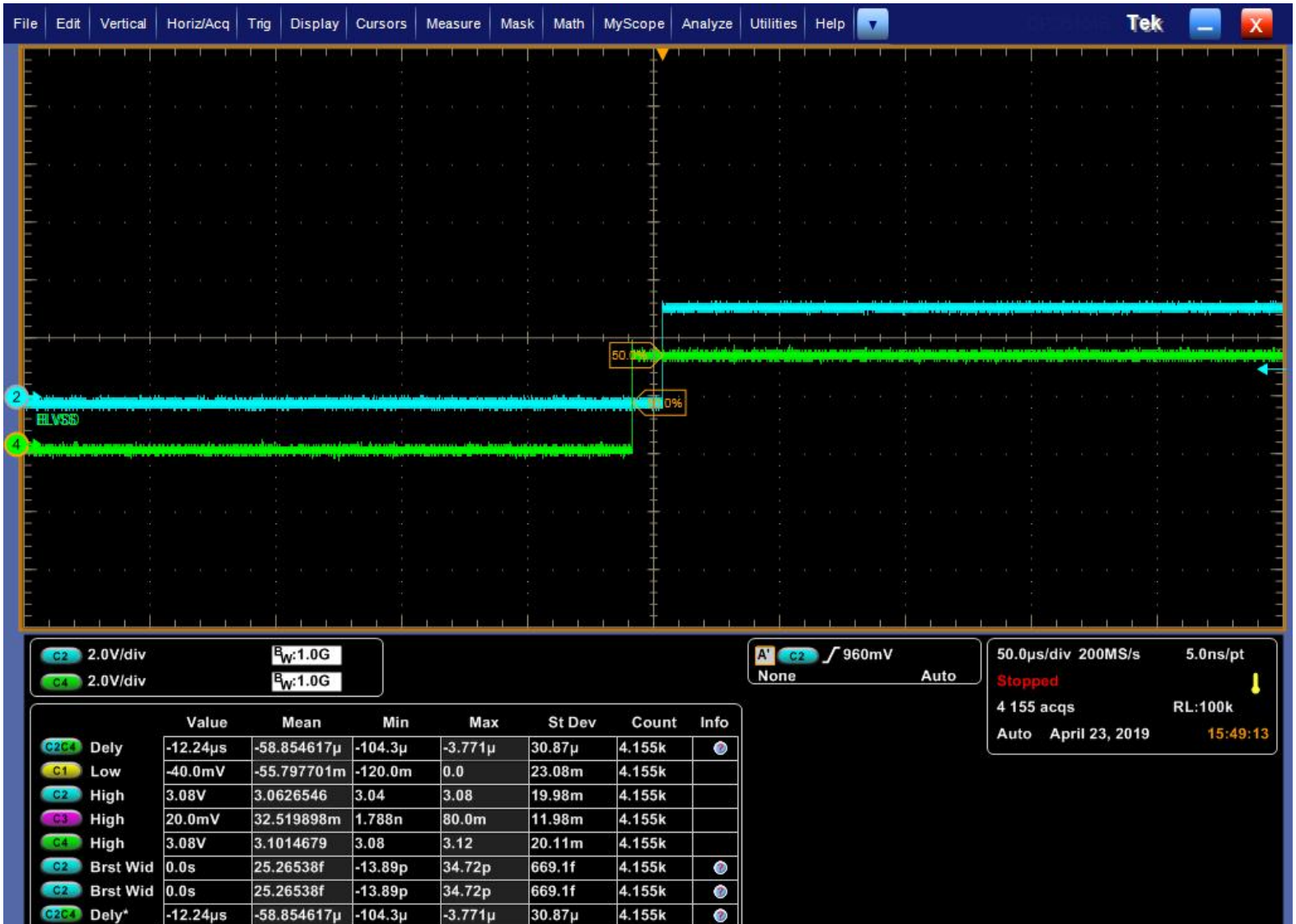
Synchronization delay

Synchronization delay:



Synchronization delay

Synchronization delay:



Synchronization delay

	Value	Mean	Min	Max	St Dev	Count	Info
C2C4 Dely	-12.24µs	-58.854617µ	-104.3µ	-3.771µ	30.87µ	4.155k	

Figure: delay

As can be seen from the figure above, the maximum phase difference between CH_LOCAL and CH_SYNC is -3.771 -(-104.3)= 100.529us when sampling 4.155k times.

summary

Data transmission through serial ports can cause some delays (although there is no blocking at the time of sending, receiving in the interrupt, reading registers directly). The baudrate we used in this example is 115200*2.

Software-only CRC verification will inevitably lead to a certain delay. If it is necessary to verify, the driver of CRC hardware peripherals should be realized in this project.

The error of demo synchronization has been tested many times, among which there are transformed master-slave roles, and the maximum error is less than 500 us. The most common is within 300us.

The test code logic is not complex, real-time processing is in the interrupt context, and STM32F4 uses Cortex M4's Nested Vectored Interrupt Controller (NVIC) real-time is very high, there should be little optimization space.