

# Body Area Network

## Demo Test Report

---

ZZDC SW

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 200mV 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

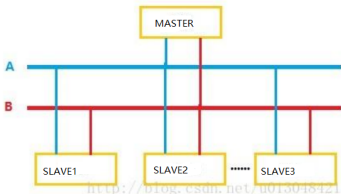


Figure: 485NET

# 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.

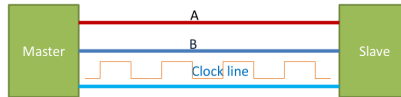


Figure: synchronous

# Our Design

We designed it like this:

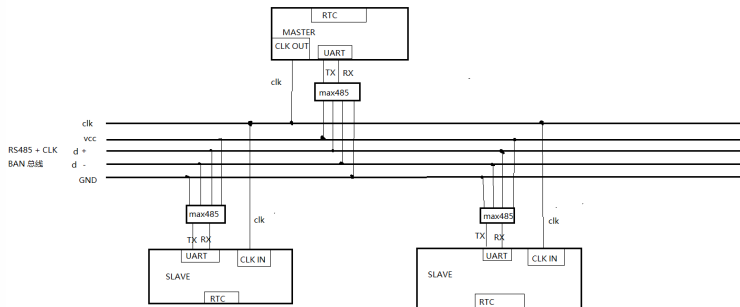


Figure: BAN

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 differential signal .

we 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 captures synchronized 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 MLCC and SLCC start counting at the same time.
- c. Slave calibrates SLCC at each subsequent SYNC\_CLK rising edge.

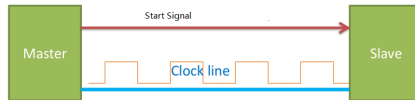


Figure: BAN



# 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 same time (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

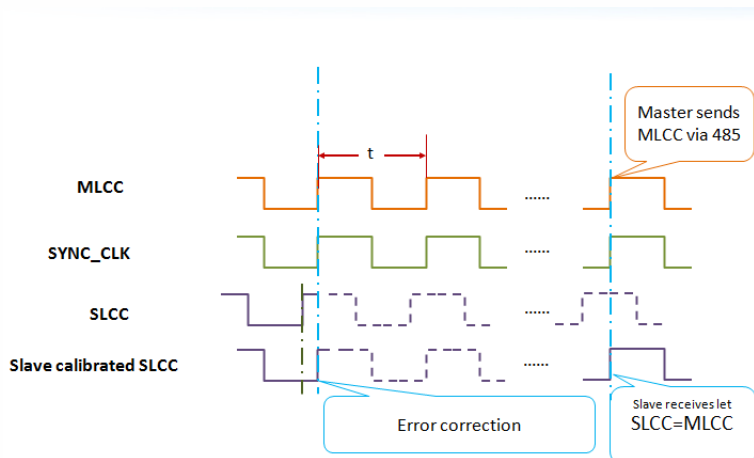


Figure: BAN

# Demo implementation

The following is the demo wiring and structure:

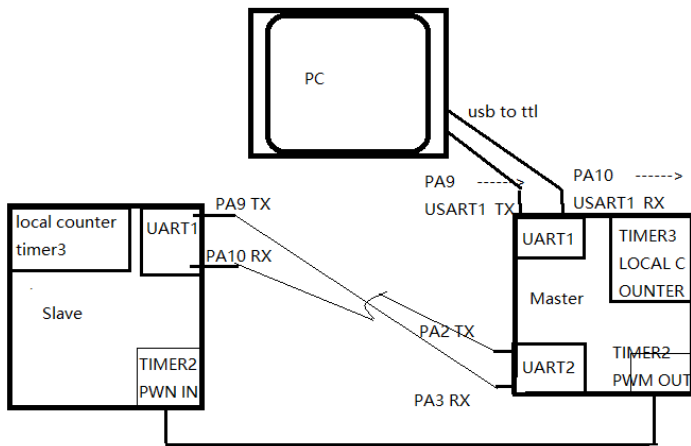


Figure: demo

**workflow**

---

1. Master and Slave Power on。 Peripheral Driver Initialization
2. The Master sends a synchronization command, when ensuring that the slave is online:
  - a.start timer2 to Generating Pulse Square Wave for synchronous。
  - b. start timer3 for MLCC(Master Local Clock Counter).
3. When Slave receives a synchronization command 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.

## Demo Workflow: Synchronize details

local\_tm\_t is the type of time described in Master and Slave code:

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

# Slave Rising Edge Capture Interrupt

tick\_sync is called every 8ms to synchronize the local timestamp:

```
1  /* every 8ms 80*100us */
2  void tick_sync(void)
3  {
4      if( dev.local_tm.jiffies_pwm + 80 < 10000 ){
5          dev.local_tm.jiffies_pwm = dev.local_tm.jiffies_pwm +
              80;
6      }
7      if( dev.local_tm.jiffies_pwm + 80 >= 10000 ){
8          dev.evt.sec_need_update=1;
9          dev.local_tm.jiffies_pwm = 80 - (10000 - dev.local_tm.
              jiffies_pwm);
10     }
11     dev.local_tm.jiffies = dev.local_tm.jiffies_pwm;~~~~~//
        synchronous
12 }
```

## Synchronize details:test point

Enter this code every 100us:

```
1  /*
2   enter this code every 100us.
3  */
4  void Tick(void)
5  {
6      dev.local_tm.jiffies ++;
7      if( dev.local_tm.jiffies % 80 == 0){
8          //GPIO A5 for compare.
9          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,1-HAL_GPIO_ReadPin(
10             GPIOA,GPIO_PIN_5));
11     }
12     /* one second passed */
13     if(dev.local_tm.jiffies >= 10000 ){
14         dev.local_tm.jiffies = 0;
15     }
16 }
```

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



## TEST Data

---

# Waveform: Unsynchronized

Let:

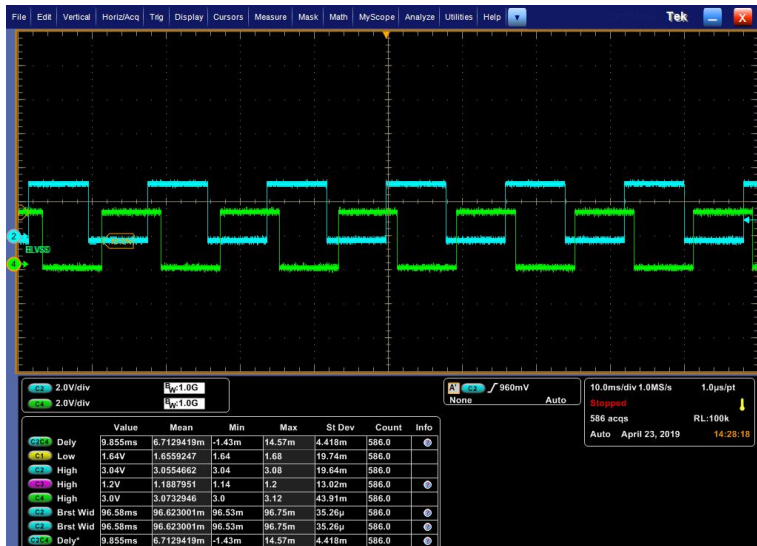
- The local clock inverts the GPIOA5 level state every 8ms. This test channel is marked CH\_LOCAL.
- The synchronous line generates a synchronous square wave with a half period of 8ms from the Master. This test channel is marked CH\_SYNC.
- In the interrupt processing function of the slave to capture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

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

The following screenshots show the unsynchronized screenshots:

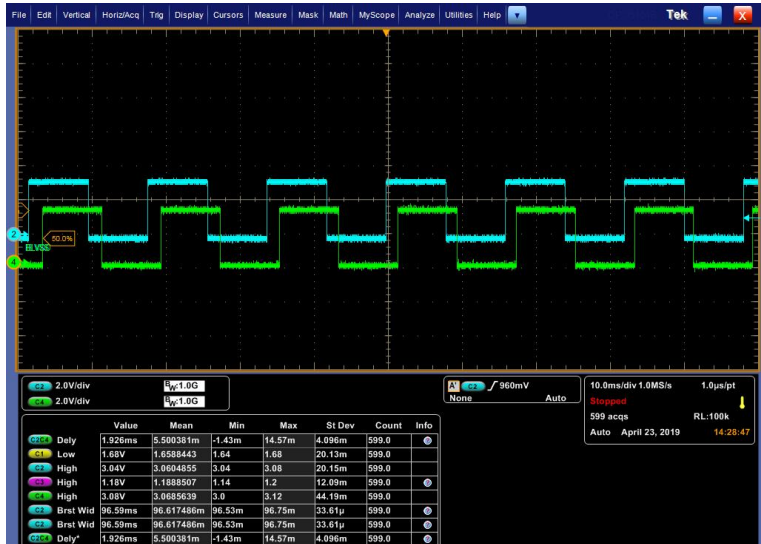
# Waveform: Unsynchronized

Unsynchronized waveform 1:



# Waveform: Unsynchronized

## Unsynchronized waveform 2:



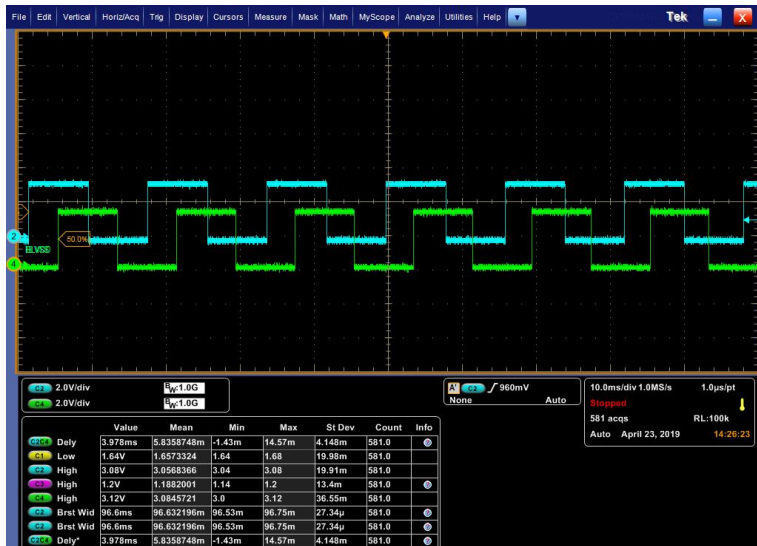
# Waveform: Unsynchronized

Unsynchronized waveform 3:



# Waveform: Unsynchronized

Unsynchronized waveform 4:



## Some explanations of unsynchronized waveforms

Return this document to unsynchronized waveform 1 ( 15) 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:

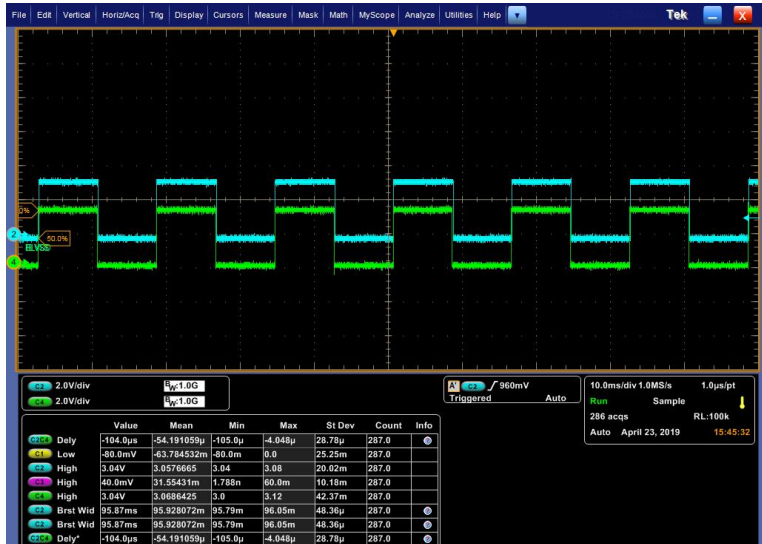
```
1 dev.local_tm.jiffies = dev.local_tm.jiffies_pwm; ^^^  
   //synchronize code
```

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



# Waveform: Synchronization

synchronized waveform :



## Waveform: Synchronization

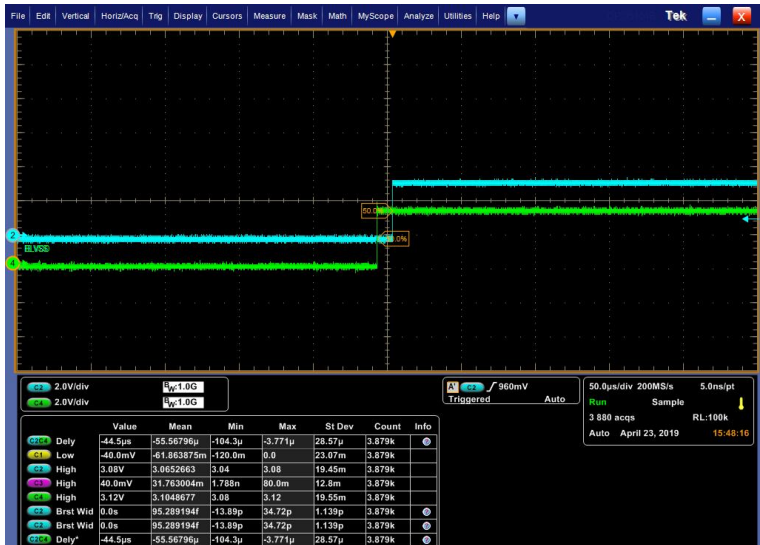
Since the rising edge of each synchronization line is captured by slave computer, synchronization interrupt will be triggered. In interrupt processing, the local clock error of each cycle will be eliminated.

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.

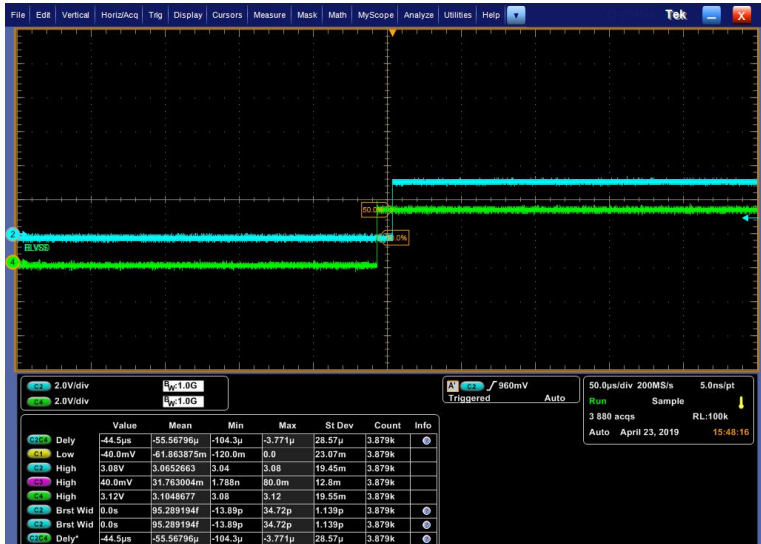
# Waveform: Synchronization

Synchronization delay:



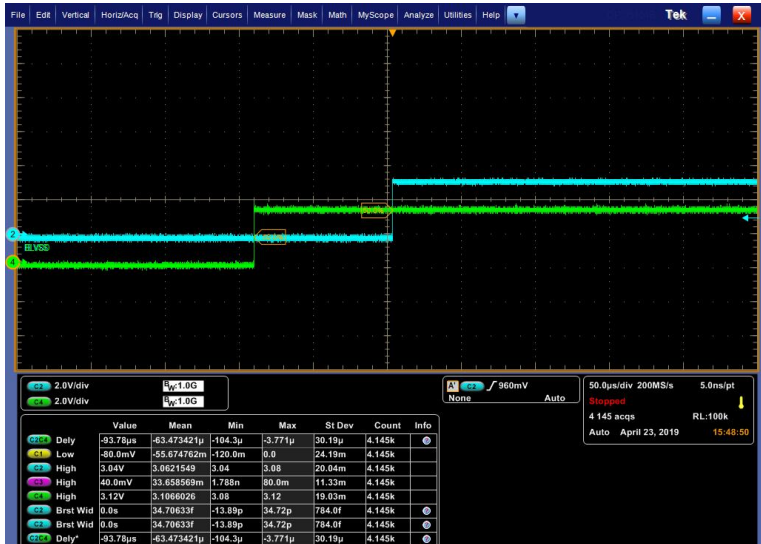
# Synchronization delay

Synchronization delay:



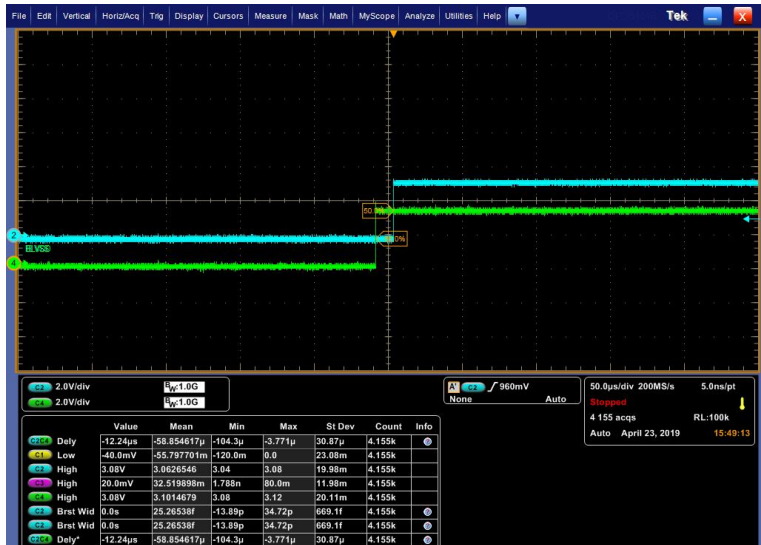
# Synchronization delay

Synchronization delay:



# Synchronization delay

Synchronization delay:



# Synchronization delay



A screenshot of a measurement tool interface showing statistics for a delay measurement. The table has columns for Value, Mean, Min, Max, St Dev, Count, and Info. The 'Value' column shows -12.24μs, 'Mean' shows -58.854617μ, 'Min' shows -104.3μ, 'Max' shows -3.771μ, 'St Dev' shows 30.87μ, and 'Count' shows 4.155k. There is a small icon with the number 0255 and a label 'Dely' on the left.

	Value	Mean	Min	Max	St Dev	Count	Info
0255 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  $-3.771 - (-104.3) = 100.529\mu\text{s}$  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 bps.

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.