**P1060973 FPGA Register Map**

1. **各模块的寄存器列表**

|  |  |  |
| --- | --- | --- |
| **地址** | **地址** | 说明 |
| COUNTER\_ADDR | 0x00000000 | COUNTER address |
| SP1\_ADDR | 0x00010000 | Scratch Pad 1 address |
| SP2\_ADDR | 0x00020000 | Scratch Pad 2 address |
| CLOCK\_ADDR | 0x00030000 | CLOCK module address |
| ILIM\_DAC\_ADDR | 0x00040000 | ILIM address |
| PWR\_IF\_ADDR | 0x00050000 | Power IF address |
| GANTRY\_EMOPS\_ADDR | 0x00050001 | Gantry EMOPS address |
| GANTRY\_96V\_IF\_ADDR | 0x00050002 | GANTRY 96V IF address |
| GANTRY\_BRAKE\_IF\_ADDR | 0x00050003 | GANTRY BRAKE IF address |
| SPD\_DMD\_IF\_ADDR | 0x00050004 | SPD DMD IF address |
| SPD\_EMOPS\_IF\_ADDR | 0x00050005 | SPD EMOPS IF address |
| LIFT\_MOTOR\_SENSOR\_IF\_ADDR | 0x00050006 | LIFT MOTOR SENSOR IF address |
| LIFT\_96V\_IF\_ADDR | 0x00050007 | LIFT 96V IF address |
| STAND\_IF\_ADDR | 0x00050008 | STAND IF address |
| SERVICE\_PENDANT\_IF\_ADDR | 0x00050009 | SERVICE PENDANT IF address |
| MAINS\_LEVEL\_IF\_ADDR | 0x0005000A | MAINS LEVEL IF address |
| EXT\_BRAKE\_IF\_ADDR | 0x0005000B | EXT BRAKE IF address |
| CCHL\_IF\_ADDR | 0x0005000C | CCHL IF address |
| ADCSELMUX\_IF\_ADDR | 0x0005000D | ADCSELMUX IF address |
| DEBUG\_IF\_ADDR | 0x0005000E | DEBUG IF address |
| GPIO\_FREE\_IF\_ADDR | 0x0005000F | GPIO FREE IF address |
| SHUNT\_EN\_CNT\_ADDR | 0x00050010 | SHUNT EN CNT address |
| ADC\_ADDR | 0x00060000 | Analog to Digital converter |
| GANTRY\_MOT\_ADDR | 0x00070000 | Gantry Motor address |
| GANTRY\_BRK1\_ADDR | 0x00071000 | Gantry Brake1 address |
| GANTRY\_BRK2\_ADDR | 0x00072000 | Gantry Brake2 address |
| GANTRY\_BRK3\_ADDR | 0x00073000 | Gantry Brake3 address |
| GANTRY\_BRK1\_RET\_ADDR | 0x00074000 | Gantry Brake1\_ret address |
| GANTRY\_BRK2\_RET\_ADDR | 0x00075000 | Gantry Brake2\_ret address |
| GANTRY\_BRK3\_RET\_ADDR | 0x00076000 | Gantry Brake3\_ret address |
| LIFT\_MOT\_ADDR | 0x00080000 | Lift Motor address |
| MSSB\_STN\_ADDR | 0x00090000 | MSSB address |
| MSSB\_SRV\_ADDR | 0x00091000 | MSSB address |
| EEP\_ADDR | 0x000B0000 | EEPROM address |

1. **时钟频率计数模块OSCILLATOR\_COUNTER**

该模块用于ATP测试项**8.1.5 Test Clock output of U55**。

1. **寄存器列表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **偏移地址** | **名称** | **读写** | **位宽** | **说明** |
| 0x0 | 控制寄存器  CNTRL Register | R/W | 3 bits | control[0]=Start  control[1]=Reset  control[2]=In Progress |
| 0x1 | 计数寄存器  Counter Register | R | 16 bits | 只读，当前计数值 |
| 0x2 | Scratch Pad | R/W | 32 bits | 通用寄存器，可读写 |

1. **详细说明**
   1. 控制寄存器（CNTRLR, 0x0）
      * [0] Start Measurement（写1启动一次测量）
      * [1] Reset Counter（写1复位计数器）
      * [2] Measurement In Progress（只读，测量进行中）
   2. 计数寄存器（COUNTR, 0x1）
      * [15:0] 计数值（只读）
   3. Scratch Pad寄存器（SPR, 0x2）
      * [31:0] 通用数据寄存器（可读写）
2. **示例**

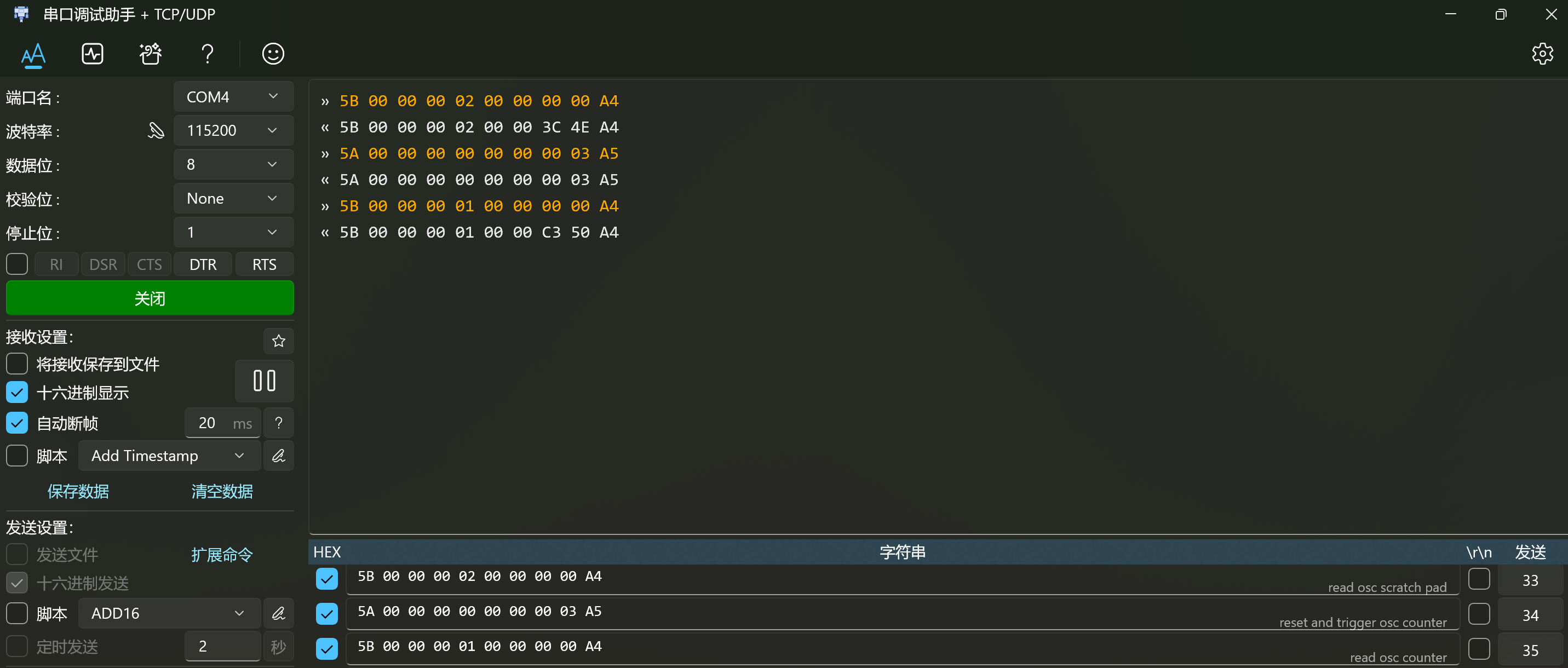
Step-1. OPB Read Scratch Pad Register@(0x00000002), return 32’h00003c4e

Step-2. OPB Write 32’h3 to CNTRL Register@(0x0000000)

Step-3. OPB Read Counter Register @(0x00000001), check return value

Counter Value = 100M/2k = 50,000 (0xC350) (+/ 50）

Freq Value = Counter\_Value \* 2k +/ 0.1M



1. **SCRATCHPAD Register模块**

该模块用于ATP测试项**8.1.4 DSP to FPGA communication Test**

1. 这个模块用于测试上位机和FPGA之间的通信，可访问的寄存器列表如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称/OPB地址** | **访问类型** | **位宽** | **说明** | **复位值** |
| **dev\_sp1**  0x00010000 | **R/W** | **32 bits** | **Scratch Pad 1** | **0x12345678** |
| **dev\_sp2**  0x00020000 | **R/W** | **32 bits** | **Scratch Pad 2** | **0x9abcbeef** |

1. **说明**

* 该模块实现了两个32位通用寄存器（Scratch Pad），支持独立读写。
* 复位时，dev\_sp1 和 dev\_sp2 分别初始化为 0x12345678 和 0x9abcbeef。

1. **示例**

Step-1. OPB Read dev\_sp1 @0x0001\_0000

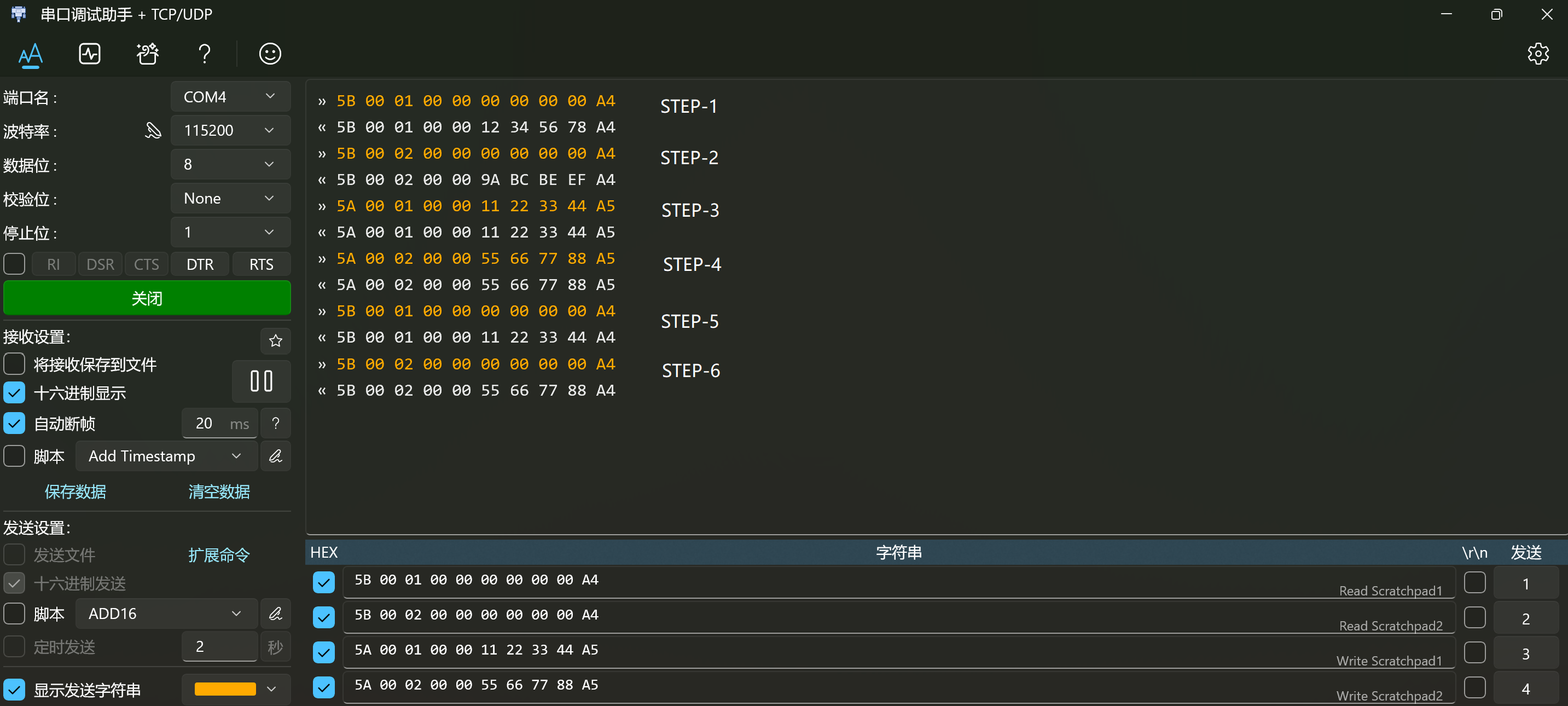
Step-2. OPB Read dev\_sp2 @0x0002\_0000

Step-3. OPB Write dev\_sp1 @0x0001\_0000

Step-4. OPB Write dev\_sp2 @0x0002\_0000

Step-5. OPB Read dev\_sp1 @0x0001\_0000

Step-6. OPB Read dev\_sp2 @0x0002\_0000



1. **时钟生成模块(CLK\_GEN)**

**注意：该模块用于配置内部时钟参数，无需修改！**

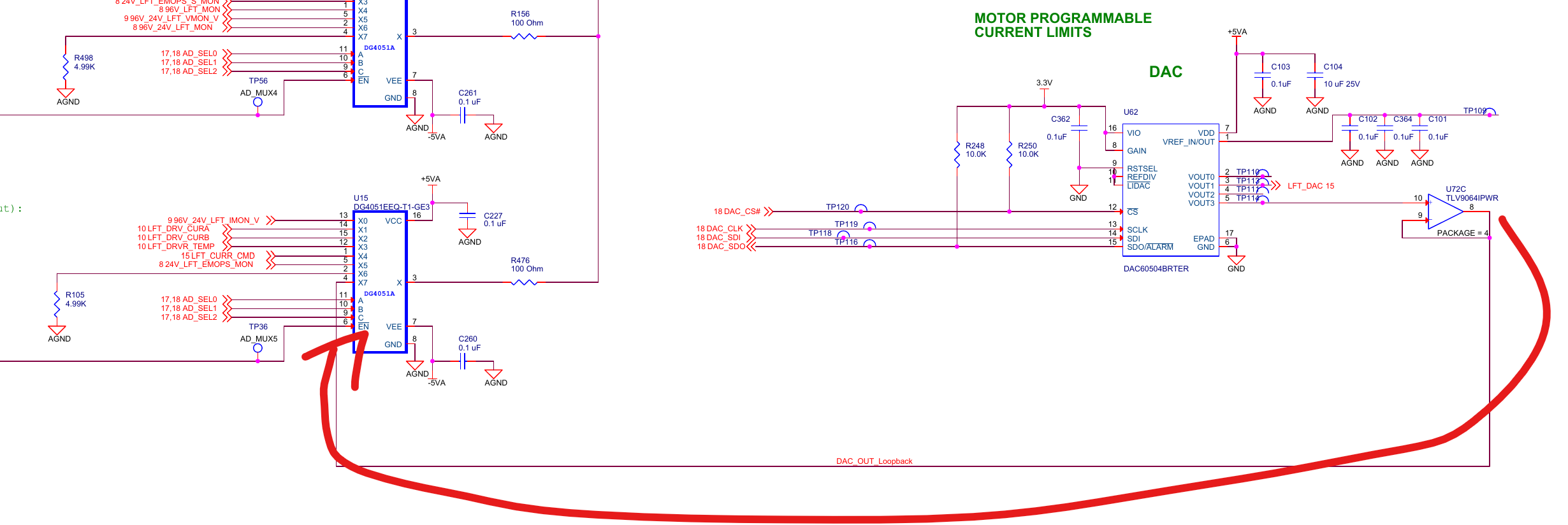
寄存器列表如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **地址宏定义** | **偏移地址** | **名称/功能** | **位宽** | **读写** | **说明** | **复位值** |
| PULSE\_5US\_ADDR | 1 | pulse5us\_div | 16位 | R/W | 5us脉冲分频系数 | 250 |
| PULSE\_50US\_ADDR | 2 | pulse50us\_div | 16位 | R/W | 50us脉冲分频系数 | 2500 |
| PULSE\_500US\_ADDR | 3 | pulse500us\_div | 16位 | R/W | 500us脉冲分频系数 | 25000 |
| PULSE\_100US\_ADDR | 4 | pulse100us\_div | 16位 | R/W | 100us脉冲分频系数 | 5000 |
| CLK\_2MHZ\_ADDR | 5 | clk2meghz\_div | 16位 | R/W | 2MHz时钟分频系数 | 25 |

1. **DAC模块**
2. 该模块用于DACx0504的测试，覆盖ATP测试项**8.1.9 DAC\_OUT signal Test at U62 Pin 5 等。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **地址宏定义** | **地址** | **名称/功能** | **位宽** | **读写** | **说明** |
| DAC0\_ADDR | 0x8 | DAC0数据 | 16bit | R/W | DAC通道0数据 |
| DAC1\_ADDR | 0x9 | DAC1数据 | 16/24 | R/W | DAC通道1数据 |
| DAC2\_ADDR | 0xA | DAC2数据 | 16/24 | R/W | DAC通道2数据 |
| DAC3\_ADDR | 0xB | DAC3数据 | 16/24 | R/W | DAC通道3数据 |

1. **说明**
   1. 写DAC0/1/2/3数据寄存器，DAC的输出对应的电平
   2. Out = Data/65536\*5
2. **举例：DAC-ADC环回测试**



STEP-1. OPB Write DAC data for VOUT3

STEP-2. OPB Write to select DAC\_OUT\_loopback as ADC input

* Measure Voltage @TP70 (4.99V for DAC data = 0xFFFF )
* Measure Voltage @TP70 (2.50V for DAC data = 0x8000 )

STEP-3. OPB Write to reset ADC Reg

STEP-4. OPB Write to Trigger ADC

STEP-5. OPB Read for ADC Status

STEP-6. OPB Read for ADC Result

* + ADC Data = 0Xff51 /65535\*5=4.98V When DAC data = 0xFFFF
  + ADC Data = 0X7FE0 /65535\*5=2.49V When DAC data = 0x8000

****

1. **GPIO模块**

该模块用于读写FPGA GPIO。

1. **寄存器地址与功能列表如下：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 地址（宏定义） | 地址值 | 寄存器名称 | 位宽 | 读写 | 复位值 | 说明 |
| PWR\_IF | 0x00050000 | POWER\_IF\_REG | 32位 | R/W | 0 | 电源接口 |
| GANTRY\_EMOPS | 0x00050001 | GANTRY\_EMOPS\_IF\_REG | 32位 | R/W | 0 | 龙门EMOPS接口 |
| GANTRY\_96V\_IF | 0x00050002 | GANTRY\_96V\_IF\_REG | 32位 | R/W | 0 | 龙门96V接口 |
| GANTRY\_BRAKE | 0x00050003 | GANTRY\_BRAKE\_IF\_REG | 32位 | R/W | 0 | 龙门刹车接口 |
| SPD\_DMD\_IF | 0x00050004 | SPD\_DMD\_IF\_REG | 32位 | R/W | 0 | SPD DMD接口 |
| SPD\_EMOPS\_IF | 0x00050005 | SPD\_EMOPS\_IF\_REG | 32位 | R/W | 0x10 | SPD EMOPS接口 |
| LIFT\_MOTOR\_SENSOR | 0x00050006 | LIFT\_MOTOR\_SENSOR\_IF\_REG | 32位 | R/W | 0x01 | 升降电机传感器接口 |
| LIFT\_96V\_IF | 0x00050007 | LIFT\_96V\_IF\_REG | 32位 | R/W | 0 | 升降96V接口 |
| STAND\_IF | 0x00050008 | STAND\_IF\_REG | 32位 | R/W | 0 | STAND接口 |
| SERVICE\_PENDANT | 0x00050009 | SERVICE\_PENDANT\_IF\_REG | 32位 | R/W | 0 | 服务手柄接口 |
| MAINS\_LEVEL\_IF\_ADDR | 0x0005000A | MAINS\_LEVEL\_IF\_REG | 32位 | R/W | 0 | MAINS LEVEL接口 |
| EXT\_BRAKE\_IF | 0x0005000B | EXT\_BRAKE\_IF\_REG | 32位 | R/W | 0 | 外部刹车接口 |
| CCHL\_IF | 0x0005000C | CCHL\_IF\_REG | 32位 | R/W | 0 | CCHL接口 |
| ADCSELMUX\_IF | 0x0005000D | ADCSELMUX\_IF\_REG | 32位 | R/W | 0xF8 | ADC选择多路复用接口 |
| DEBUG\_IF | 0x0005000E | DEBUG\_IF\_REG | 32位 | R/W | 0 | 调试接口 |
| GPIO\_FREE\_IF | 0x0005000F | GPIO\_FREE\_IF\_REG | 32位 | R/W | 0 | 通用GPIO |
| SHUNT\_EN\_CNT | 0x00050010 | SHUNT\_EN\_CNT\_MAX\_REG | 32位 | R/W | 250000000 | 分流使能计数最大值（2.5s） |

1. **GPIO Output**

// 参考GPIO.v

// GPIO Outputs

assign FAN\_EN               = POWER\_IF\_REG[0];

assign P12V\_ISO\_EN          = POWER\_IF\_REG[1];

assign WD\_TRIG              = POWER\_IF\_REG[2];

assign P24V\_GNT\_EMOPS\_EN    = GANTRY\_EMOPS\_IF\_REG[0];

assign GNT\_EMOPS\_EN         = GANTRY\_EMOPS\_IF\_REG[1];

assign GNT\_MOT\_PWR\_EN       = GANTRY\_96V\_IF\_REG[0];

assign GNT\_BRK\_PWR\_EN       = GANTRY\_96V\_IF\_REG[1];

assign GNT\_SHUNT\_EN         = GANTRY\_96V\_IF\_REG[2];

assign GNT\_BRK\_EXT\_EN       = GANTRY\_BRAKE\_IF\_REG[0];

assign GNT\_BRK\_SW\_MON       = GANTRY\_BRAKE\_IF\_REG[1];

assign GNT\_HW\_EN\_MON        = SPD\_DMD\_IF\_REG[0];

assign LFT\_ST\_DISB\_MON      = SPD\_DMD\_IF\_REG[1];

assign LFT\_HW\_EN\_MON        = SPD\_DMD\_IF\_REG[2];

assign GNT\_ST\_DISB\_MON      = SPD\_DMD\_IF\_REG[3];

assign DMD\_PWR\_OK           = SPD\_DMD\_IF\_REG[4];

assign SPARE\_MON            = SPD\_DMD\_IF\_REG[5];

assign EMOPS\_STAT1          = SPD\_EMOPS\_IF\_REG[0];

assign EMOPS\_STAT2          = SPD\_EMOPS\_IF\_REG[1];

assign LFT\_ROT\_BRK\_RLS      = SPD\_EMOPS\_IF\_REG[2];

assign LAT\_LNG\_BRK\_RLS      = SPD\_EMOPS\_IF\_REG[3];

assign SRVC\_CCH\_GNT\_N       = SPD\_EMOPS\_IF\_REG[4];

assign EM\_CCH\_24V\_EN        = SPD\_EMOPS\_IF\_REG[5];

assign LFT\_HALL\_PWR\_EN\_N    = LIFT\_MOTOR\_SENSOR\_IF\_REG[0];

assign P24V\_LFT\_EMOPS\_EN    = LIFT\_96V\_IF\_REG[0];

assign LFT\_SHUNT\_EN         = LIFT\_96V\_IF\_REG[1];

assign LFT\_EMOPS\_EN         = LIFT\_96V\_IF\_REG[2];

assign LFT\_MOT\_PWR\_EN       = LIFT\_96V\_IF\_REG[3];

assign EXOPS\_GNT\_24V\_EN     = STAND\_IF\_REG[0];

assign PAN\_24V\_SW           = SERVICE\_PENDANT\_IF\_REG[0];

assign EXT\_BRK1\_DRV\_EN      = EXT\_BRAKE\_IF\_REG[0];

assign EXT\_BRK2\_DRV\_EN      = EXT\_BRAKE\_IF\_REG[1];

assign EXT\_BRK3\_DRV\_EN      = EXT\_BRAKE\_IF\_REG[2];

assign LFT\_SER\_CLK          = CCHL\_IF\_REG[0];

assign LFT\_SER\_SYNC         = CCHL\_IF\_REG[1];

assign LFT\_SER\_DATA0        = CCHL\_IF\_REG[2];

assign LFT\_SER\_DATA1        = CCHL\_IF\_REG[3];

assign AD\_SEL0              = ADCSELMUX\_IF\_REG[0];

assign AD\_SEL1              = ADCSELMUX\_IF\_REG[1];

assign AD\_SEL2              = ADCSELMUX\_IF\_REG[2];

assign AD\_MUX1\_N            = (ADCSELMUX\_IF\_REG[7:3] == 5'b11110) ? 1'b0 : 1'b1; // AD\_MUX1\_N is active low

assign AD\_MUX2\_N            = (ADCSELMUX\_IF\_REG[7:3] == 5'b11101) ? 1'b0 : 1'b1; // AD\_MUX2\_N is active low

assign AD\_MUX3\_N            = (ADCSELMUX\_IF\_REG[7:3] == 5'b11011) ? 1'b0 : 1'b1; // AD\_MUX3\_N is active low

assign AD\_MUX4\_N            = (ADCSELMUX\_IF\_REG[7:3] == 5'b10111) ? 1'b0 : 1'b1; // AD\_MUX4\_N is active low

assign AD\_MUX5\_N            = (ADCSELMUX\_IF\_REG[7:3] == 5'b01111) ? 1'b0 : 1'b1; // AD\_MUX5\_N is active low

assign DBUG\_MISO            = DEBUG\_IF\_REG[0];

assign GPIO2                = GPIO\_FREE\_IF\_REG[0];

assign GPIO3                = GPIO\_FREE\_IF\_REG[1];

assign GPIO5                = GPIO\_FREE\_IF\_REG[2];

1. **GPIO Input**

|  |  |
| --- | --- |
| **Input信号** | **寄存器及位** |
| P24V\_GOOD\_N | POWER\_IF\_REG[16] |
| FAN\_FAIL\_N | POWER\_IF\_REG[17] |
| P24V\_GNT\_EMOPS\_PG | GANTRY\_EMOPS\_IF\_REG[16] |
| P12V\_ACT\_DIODE\_ON\_N | GANTRY\_EMOPS\_IF\_REG[17] |
| OC\_V\_GNT\_MOT\_DRV | GANTRY\_96V\_IF\_REG[16] |
| OC\_V\_GNT\_BRK\_DRV | GANTRY\_96V\_IF\_REG[17] |
| GNT\_MOT\_PWR\_FLT\_N | GANTRY\_96V\_IF\_REG[18] |
| GNT\_BRK\_PWR\_FLT\_N | GANTRY\_96V\_IF\_REG[19] |
| GNT\_EMOPS\_OV\_L | GANTRY\_96V\_IF\_REG[20] |
| P5V\_ISO\_MON\_GNT | GANTRY\_96V\_IF\_REG[21] |
| GNT\_SHUNT\_ON | GANTRY\_96V\_IF\_REG[22] |
| P5V\_ISO\_MON\_LFT | GANTRY\_96V\_IF\_REG[23] |
| PWM\_BRK\_EXT\_FAULT | GANTRY\_BRAKE\_IF\_REG[16] |
| GNT\_BRK1\_FB\_N | GANTRY\_BRAKE\_IF\_REG[17] |
| GNT\_BRK2\_FB\_N | GANTRY\_BRAKE\_IF\_REG[18] |
| GNT\_BRK3\_FB\_N | GANTRY\_BRAKE\_IF\_REG[19] |
| SPDIO\_FLT\_N | SPD\_DMD\_IF\_REG[16] |
| SPD2\_24V\_STATUS\_N | SPD\_DMD\_IF\_REG[17] |
| LFT\_ST\_DISB | SPD\_DMD\_IF\_REG[18] |
| LFT\_HW\_EN\_N | SPD\_DMD\_IF\_REG[19] |
| GNT\_ST\_DISB | SPD\_DMD\_IF\_REG[20] |
| GNT\_HW\_EN\_N | SPD\_DMD\_IF\_REG[21] |
| EM\_DOWN\_LIMIT | SPD\_EMOPS\_IF\_REG[16] |
| EM\_UP\_LIMIT | SPD\_EMOPS\_IF\_REG[17] |
| CCH\_LFT\_MOT\_EN\_N | SPD\_EMOPS\_IF\_REG[18] |
| CCH\_LFT\_DWN\_N | SPD\_EMOPS\_IF\_REG[19] |
| CCH\_LAT\_LNG\_FLOAT\_N | SPD\_EMOPS\_IF\_REG[20] |
| EMO\_GOOD\_N | SPD\_EMOPS\_IF\_REG[21] |
| SITE\_24V\_PWR\_OK\_N | SPD\_EMOPS\_IF\_REG[22] |
| SPD\_EMOPS\_FLT\_N | SPD\_EMOPS\_IF\_REG[23] |
| LFT\_HALL\_PWR\_OK | LIFT\_MOTOR\_SENSOR\_IF\_REG[16] |
| LFT\_HALL\_SNS1\_N | LIFT\_MOTOR\_SENSOR\_IF\_REG[17] |
| LFT\_HALL\_SNS2\_N | LIFT\_MOTOR\_SENSOR\_IF\_REG[18] |
| LFT\_HALL\_SNS3\_N | LIFT\_MOTOR\_SENSOR\_IF\_REG[19] |
| P24V\_LFT\_EMOPS\_PG | LIFT\_96V\_IF\_REG[16] |
| OC\_V\_LFT\_MOT\_DRV | LIFT\_96V\_IF\_REG[17] |
| LFT\_SHUNT\_ON | LIFT\_96V\_IF\_REG[18] |
| LFT\_EMOPS\_OV\_L | LIFT\_96V\_IF\_REG[19] |
| LFT\_MOT\_PWR\_FLT\_N | LIFT\_96V\_IF\_REG[20] |
| GNT\_CCW\_LIMIT | STAND\_IF\_REG[16] |
| GNT\_CW\_LIMIT | STAND\_IF\_REG[17] |
| MAINS\_LEVEL\_FB | MAINS\_LEVEL\_IF\_REG[16] |
| LFT\_SERIO\_FLT\_N | CCHL\_IF\_REG[16] |
| LFT\_MTN\_EN\_N | CCHL\_IF\_REG[17] |
| LFT\_SER\_PAGE\_SEL\_N | CCHL\_IF\_REG[18] |
| LFT\_DOWN\_LIMIT | CCHL\_IF\_REG[19] |
| LFT\_UP\_LIMIT | CCHL\_IF\_REG[20] |
| DBUG\_MOSI | DEBUG\_IF\_REG[16] |
| DBUG\_CS\_N | DEBUG\_IF\_REG[17] |
| DBUG\_SCLK | DEBUG\_IF\_REG[18] |
| DBUG\_ACTIVE | DEBUG\_IF\_REG[19] |

1. **示例：读写Power相关GPIO**

如下命令使能FAN\_EN和P12V\_ISO\_EN

5A 00 05 00 00 00 00 00 03 A5

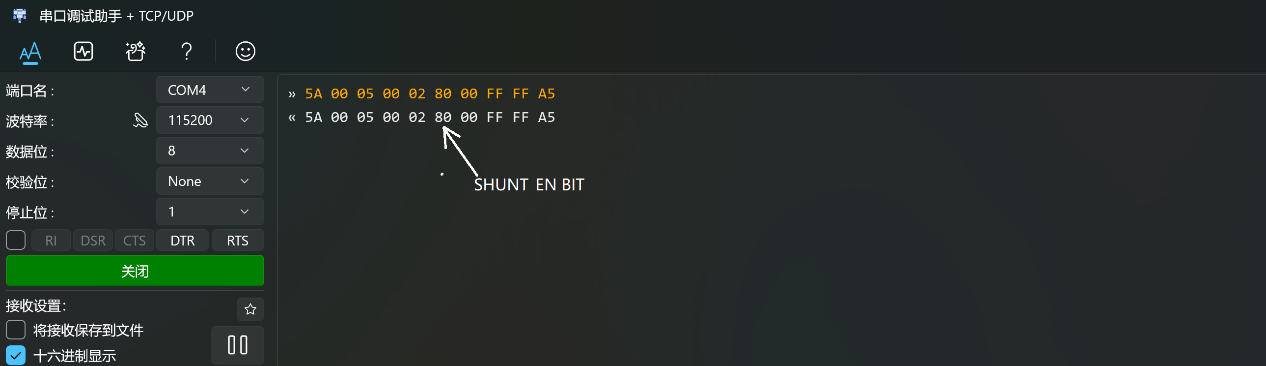
如下命令读取FAN\_EN和P12V\_ISO\_EN，P24V\_GOOD\_N, FAN\_FAIL\_N

5B 00 05 00 00 00 00 00 00 A4

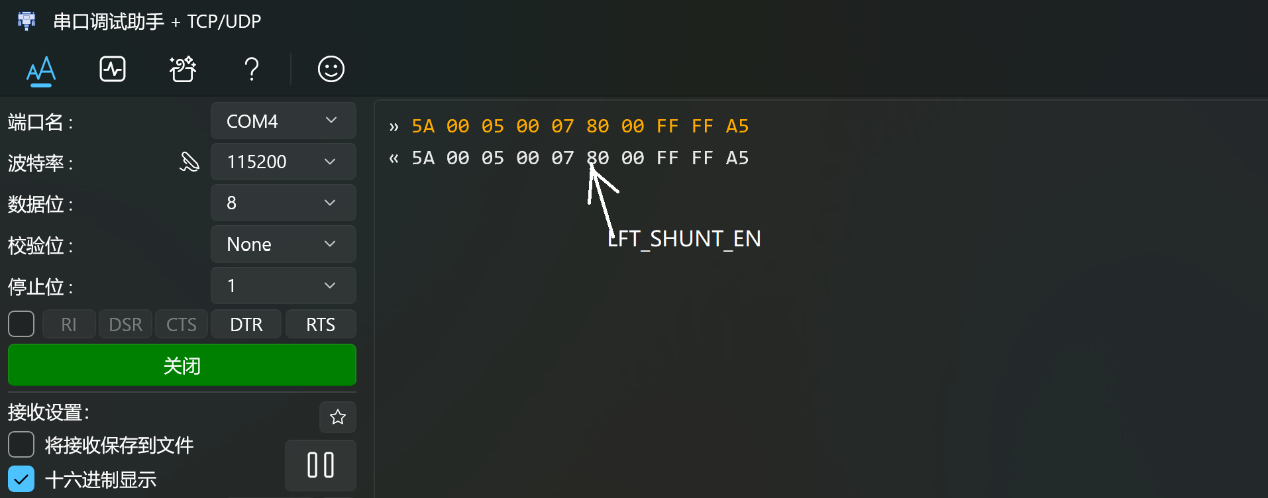
查找对应的bit得到GPIO的状态。

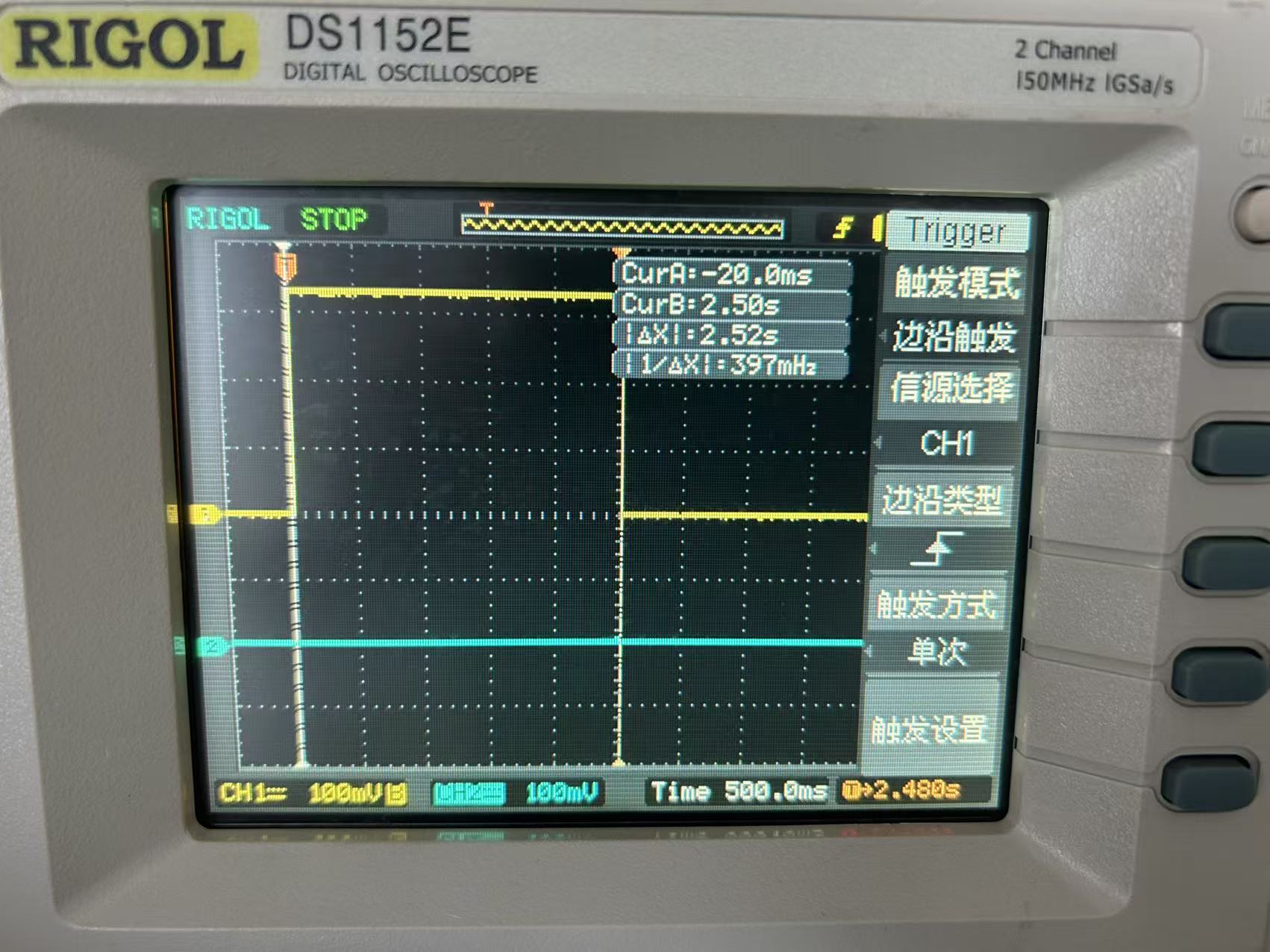
1. **特例：SHUNT\_EN**

OPB\_DATA[31]=1将使能SHUNT测试，GNT\_SHUNT\_EN输出只持续2.5s



类似的方法测试LFT\_SHUNT\_EN，命令如下：





1. **ADC模块**

该模块用于ADC ADS8864的测试, 覆盖ATP测试项**8.6.21-8.6.26等。**

1. **寄存器定义如下**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **地址宏定义** | **地址范围** | **名称/功能** | **位宽** | **读写** | **说明** | **复位值** |
| D\_RAM\_ADDR | 0x000-0x3FF | 数据RAM | 16位 | R | 采集数据存储区 | 0 |
| CNTRL\_ADDR | 0x800 | 控制寄存器 | 3位 | R/W | control[1]: start, control[2]: reset | 0 |
| CLK\_DIV\_AQ\_ADDR | 0x802 | 采样时钟分频 | 16位 | R/W | 采样时钟分频系数 | 250 |
| CLK\_DIV\_SD\_ADDR | 0x804 | 串行时钟分频 | 16位 | R/W | 串行数据时钟分频系数 | 3 |
| D\_LENGTH\_ADDR | 0x806 | 采样长度 | 10位 | R/W | 采集数据点数, Max=1024 | 5 |
| STATUS\_ADDR | 0x808 | 状态寄存器 | 4位+1 | R | status[0]:busy  [1]:done  [2]:aq timeout  [3]:busy timeou  [31]:ADC\_SDOUT | 0 |
| STATE\_ADDR | 0x80A | 状态机状态 | 5位 | R | 当前状态机状态 | 0 |
| SP\_ADDR | 0x900 | SP寄存器 | 16位 | R | 调试/辅助寄存器 | 0x1122 |

1. **说明**

* **数据RAM**：0x000~0x3FF，采集到的ADC数据，16位宽，只读。
* **控制寄存器**：写入可启动采样、复位模块，读出当前控制状态。
* **分频寄存器**：可配置采样时钟和串行数据时钟分频系数。**无特殊需求，不用修改！**
* **采样长度**：设置一次采集的数据点数。
* **状态寄存器**：包含采集状态、完成标志、超时等信息。
* **状态机状态**：可读出当前状态机的状态码，用于辅助调试。
* **SP寄存器**：辅助/调试用途。

1. **举例**

STEP1. 配置ADC SEL\_MEX，选择哪一路采样信号，参考Section-6.GPIO模块

STEP2. 复位ADC寄存器

STEP3. 配置采样长度

STEP4. 触发ADC采数

STEP5. 读ADC状态

STEP6. 读ADC结果



1. **Gantry Motor模块**

该模块用于Gantry 3相电机驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

**举例1 – 测试模式0：**

STEP-1. OPB WRITE配置测试模式0

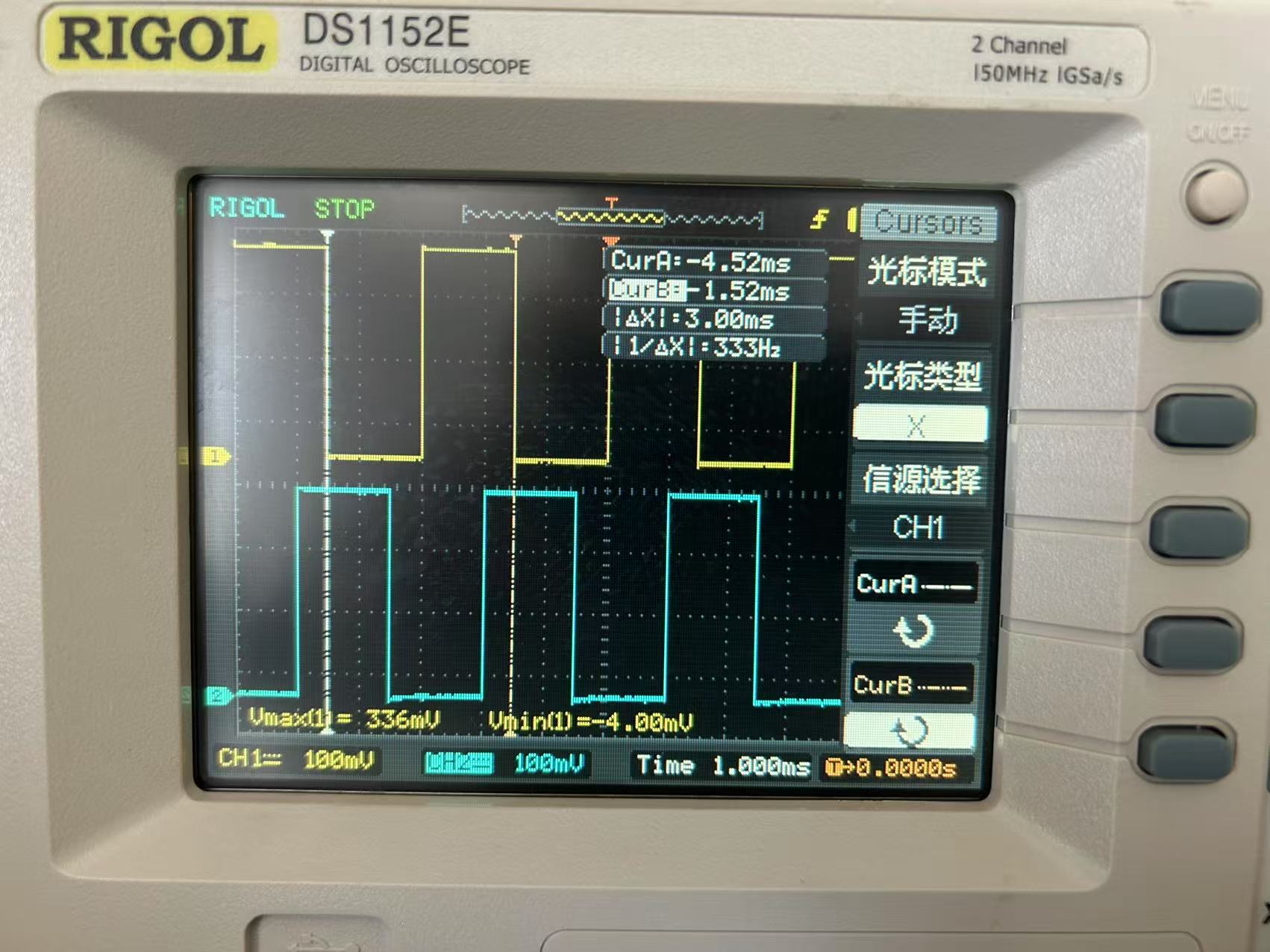
* PWM输出由START和STOP消息控制

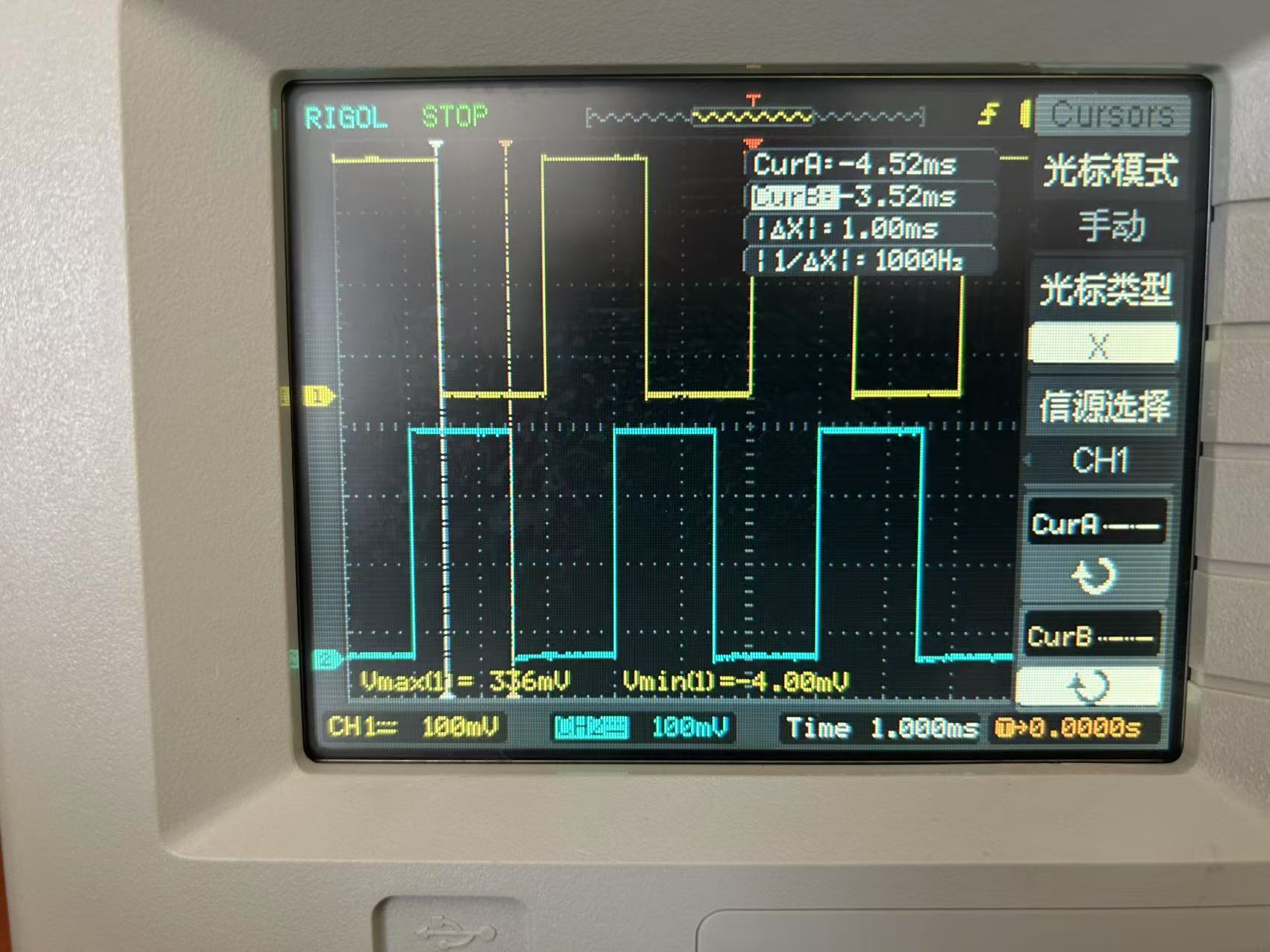
STEP-2. OPB WRITE发送START命令，触发PWM输出

STEP-3. OPB WRITE 发送STOP命令，关闭PWM输出



PWM输出波形（A/B两相，频率333.3Hz,相差120度）





**举例2– 测试模式1：**

STEP-1. OPB WRITE配置测试模式1

* PWM输出由START和STOP消息控制
* PWM输出100ms后将自动关断
* OC\_V\_GNT\_MOT\_DRV也会触发关断

STEP-2. OPB WRITE Dummy PWM Param1(不能省略!)

STEP-3. OPB WRITE Dummy PWM Param2(不能省略!)

STEP-4. OPB WRITE Dummy PWM Param3(不能省略!)

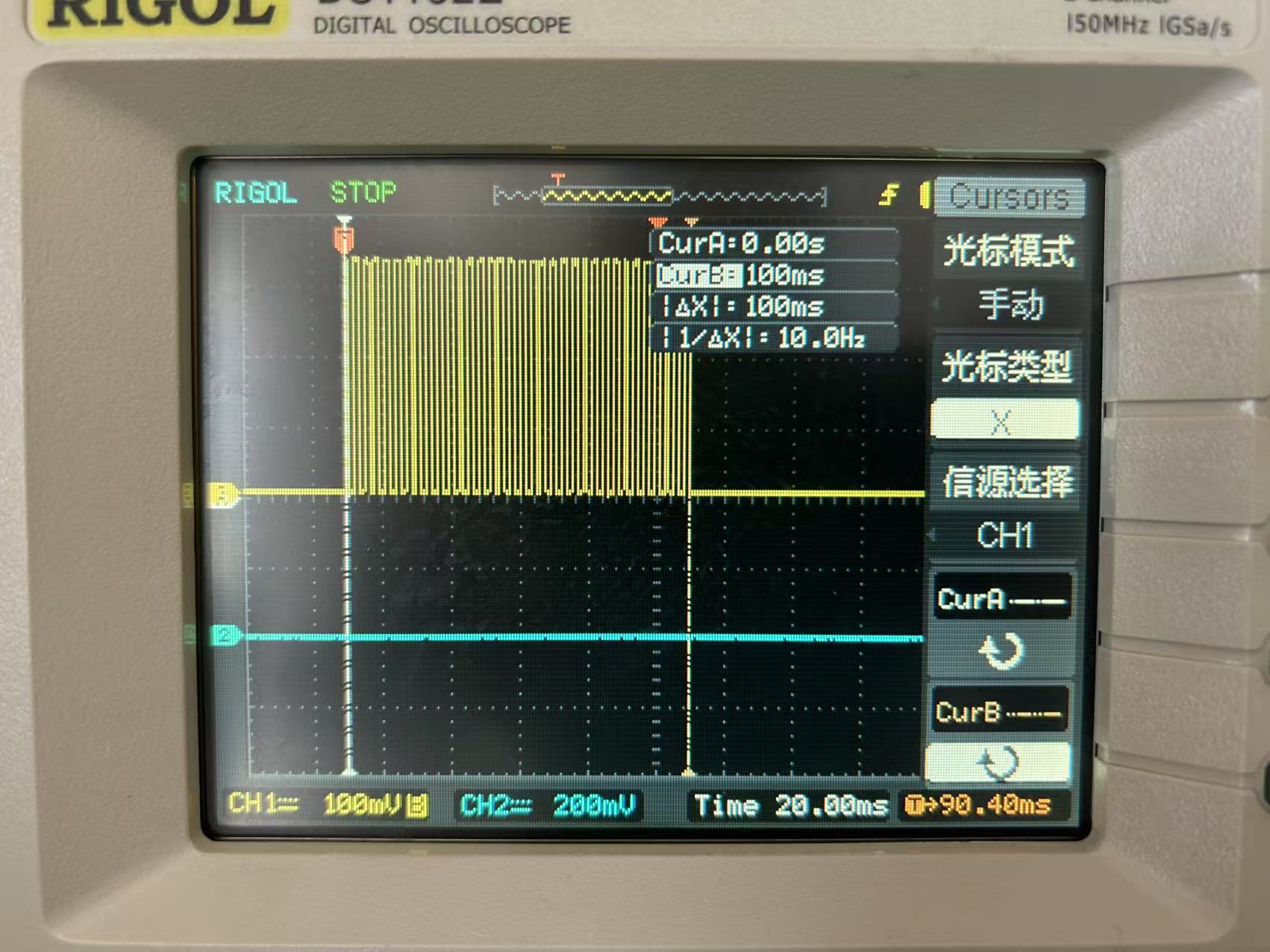
STEP-5. OPB WRITE发送START命令，触发PWM输出

STEP-6. OPB Read Status Register @offset 0x06

* Bit-0: 1 indicates Motor Overcurrent was triggered



测试GNT\_PWM\_PHA\_HI输出波形如下图：



1. **Gantry BRK1模块**

该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

示例1. 测试模式0

STEP-1. OPB WRITE配置测试模式0

* 5A 00 07 10 00 00 0F A0 03 A5

STEP-2. OPB WRITE DUMMY PARAM1 (不能省略!)

STEP-3. OPB WRITE DUMMY PARAM2 (不能省略!)

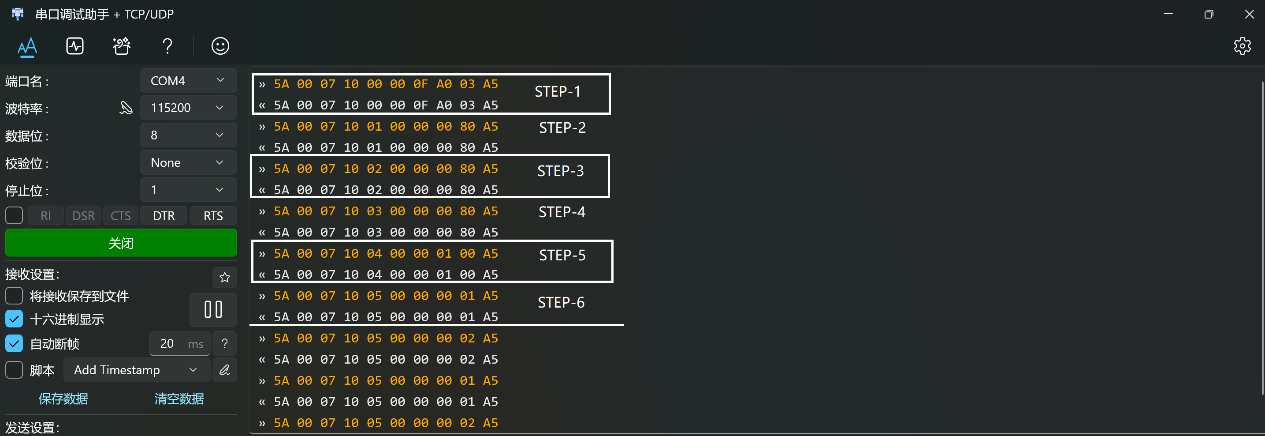
STEP-4. OPB WRITE DUMMY PARAM3 (不能省略!)

STEP-5. OPB WRITE配置BRK占空比，最大值为311, PWM Freq is fixed as 40kHz

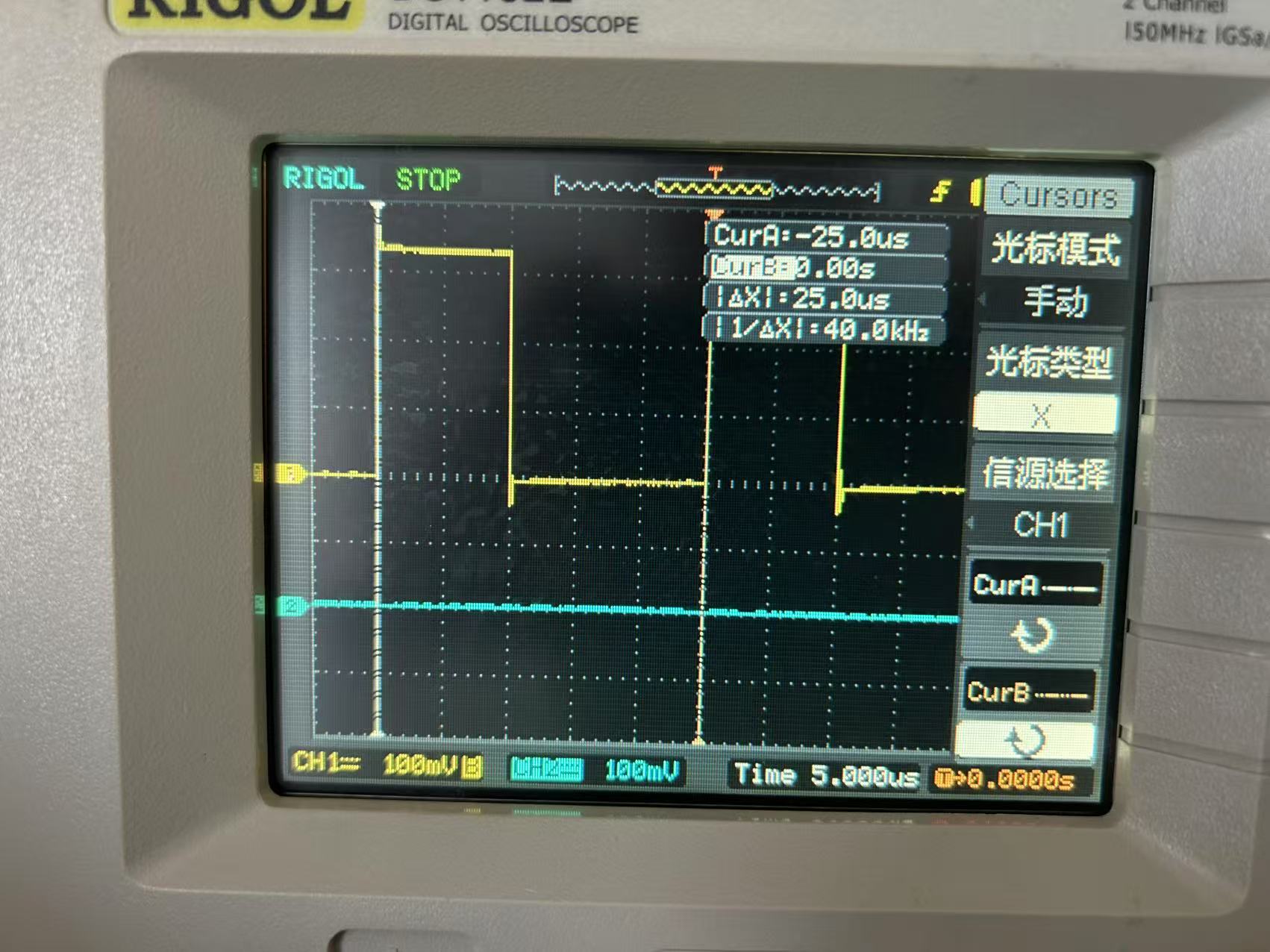
STEP-6. OPB WRITE发送START命令，触发PWM输出

STEP-7. OPB WRITE发送STOP命令，触发PWM输出

* 5A 00 07 10 05 00 00 00 02 A5



观察GNT\_BRK1\_PWM\_HI输出如下图，调整占空比，可以看到输出波形随着变化。



示例1. 测试模式1

STEP-1. OPB WRITE配置测试模式1，和测试时长(最大为5000\*25us=125ms)

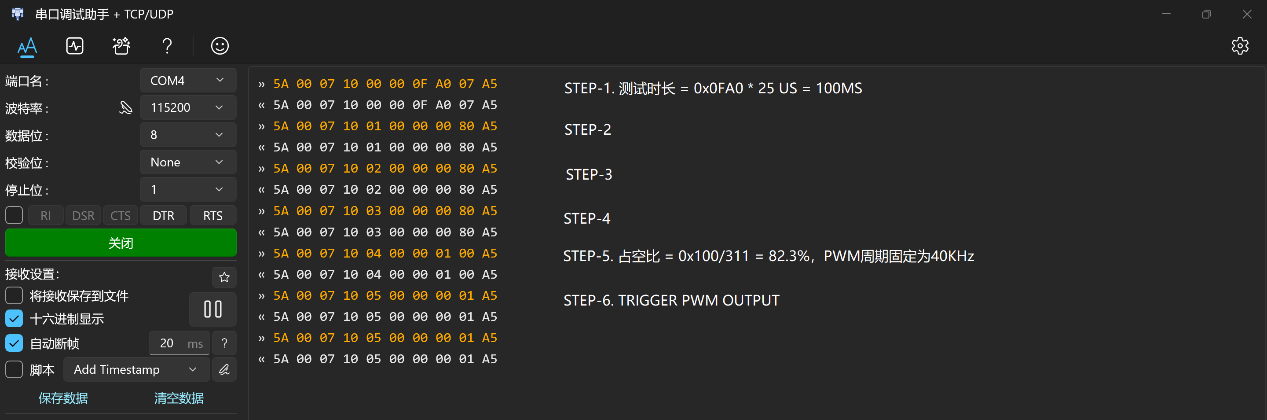
STEP-2. OPB WRITE DUMMY PARAM1 (不能省略!)

STEP-3. OPB WRITE DUMMY PARAM2 (不能省略!)

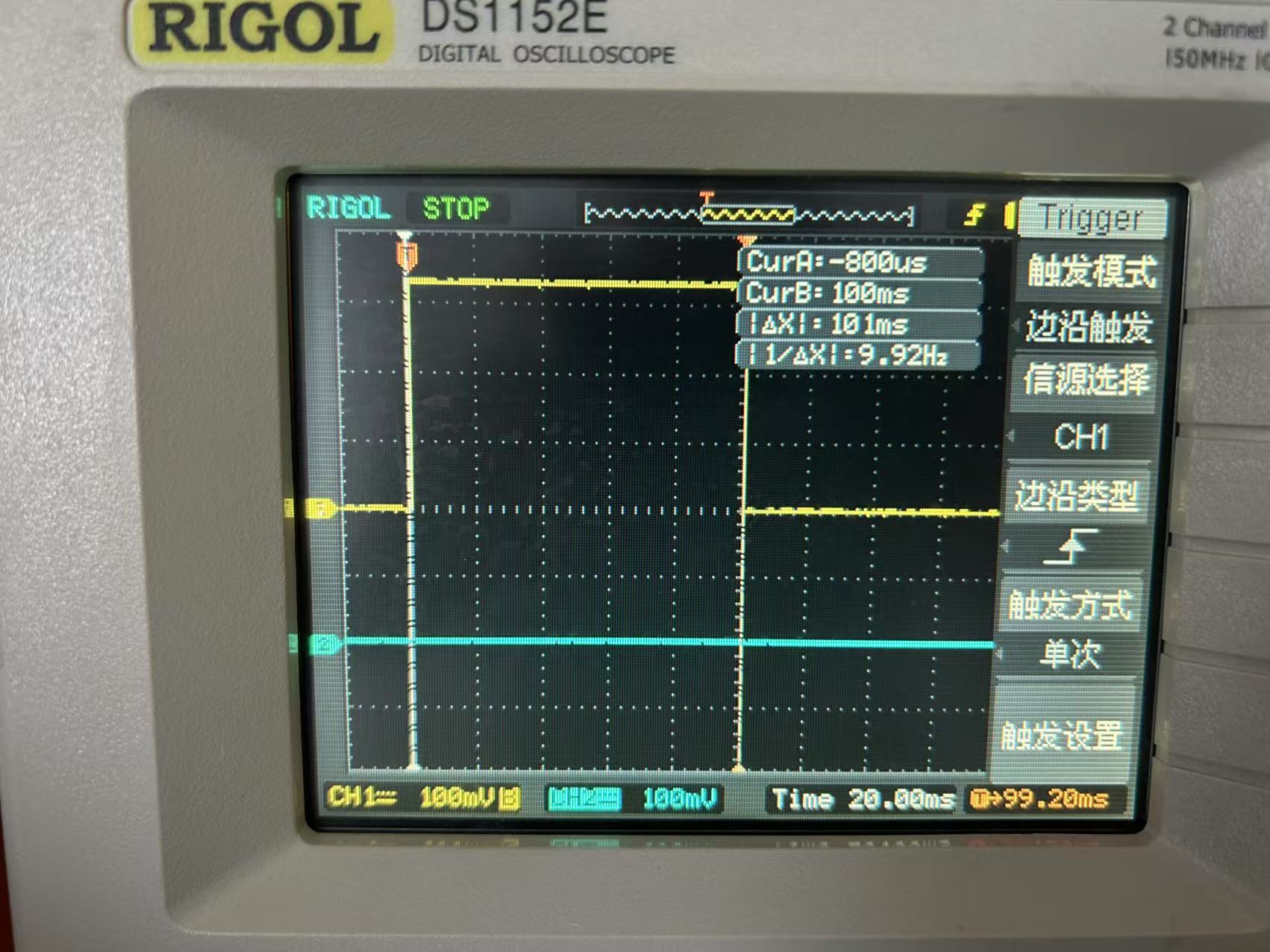
STEP-4. OPB WRITE DUMMY PARAM3 (不能省略!)

STEP-5. OPB WRITE配置BRK占空比，最大值为311

STEP-6. OPB WRITE发送START命令，触发PWM输出



观察GNT\_BRK1\_PWM\_HI输出如下图，受限于示波器分辨率，PWM\_HI输出占空比看起来是100%



1. **Gantry BRK2模块**

该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

参考9.Gantry BRK1模块，注意寄存器基地址不同

1. **Gantry BRK3模块**

该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

参考9.Gantry BRK1模块，注意寄存器基地址不同

1. **Gantry BRK1\_RET模块**

该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

参考9.Gantry BRK1模块，注意寄存器基地址不同

1. **Gantry BRK2\_RET模块**

该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

参考9.Gantry BRK1模块，注意寄存器基地址不同

1. **Gantry BRK3\_RET模块**

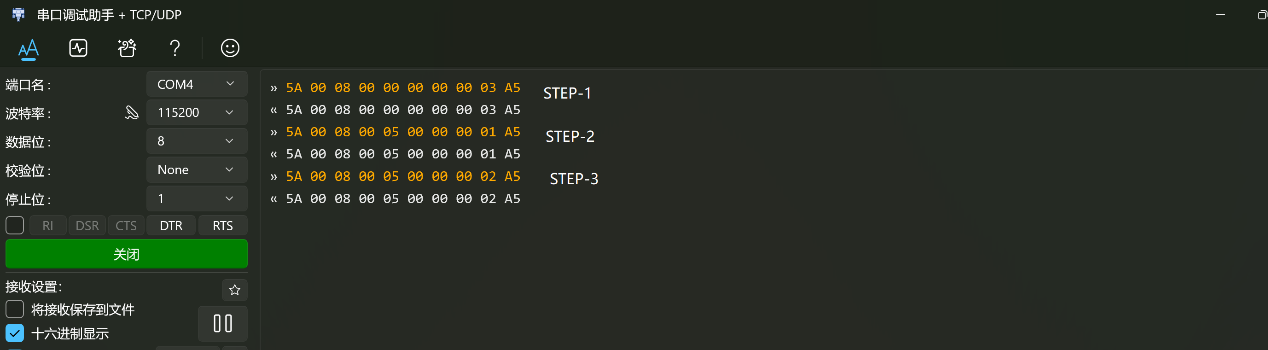
该模块用于Gantry Brake驱动的测试, 覆盖ATP测试项**8.9 Gantry Motor Test**

参考9.Gantry BRK1模块，注意寄存器基地址不同

1. **LIFT Motor模块**

该模块用于LIFT 3相电机驱动的测试, 覆盖ATP测试项**8.10 LIFT Motor Test**

参考8.Gantry Motor模块,注意两者的地址不一样，测试命令如下：



1. **MSSB STN模块**

该模块用于MSSB STN(U86)的测试, 覆盖ATP测试项**8.4.10-8.4.12**

1. **寄存器列表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **偏移地址** | **名称/功能** | **位宽** | **读写** | **说明** |
| 0x0 | mssb\_config | 32位 | R/W | [31]:trans\_start，写1启动传输，自动清零；[19:0]:trans\_bytes，传输字节数 |
| 0x1 | mssb\_status | 32位 | R | [1:0]：00=busy，01=完成无错，11=完成有错 |
| 0x2 | mssb\_recv\_bytes | 32位 | R | [19:0]: 实际接收字节数 |
| 0x3 | mssb\_error\_bytes | 32位 | R | [19:0]: 错误字节数 |

1. **说明**
   1. **mssb\_config**

* [31] trans\_start：写1启动传输，自动清零
* [19:0] trans\_bytes：本次要传输的字节数（最大1M，默认10K）
  1. **mssb\_status**
* [1:0] 状态：00=busy，01=完成无错，11=完成有错
  1. **mssb\_recv\_bytes**
* [19:0] 实际接收字节数
  1. **mssb\_error\_bytes**
* [19:0] 错误字节数（接收数据与预期不符的字节数）

1. **举例**

参考MSSB SRV模块测试

1. MSSB SRV模块

该模块用于MSSB SRV(U98)的测试, 覆盖ATP测试项**8.4.10-8.4.12**

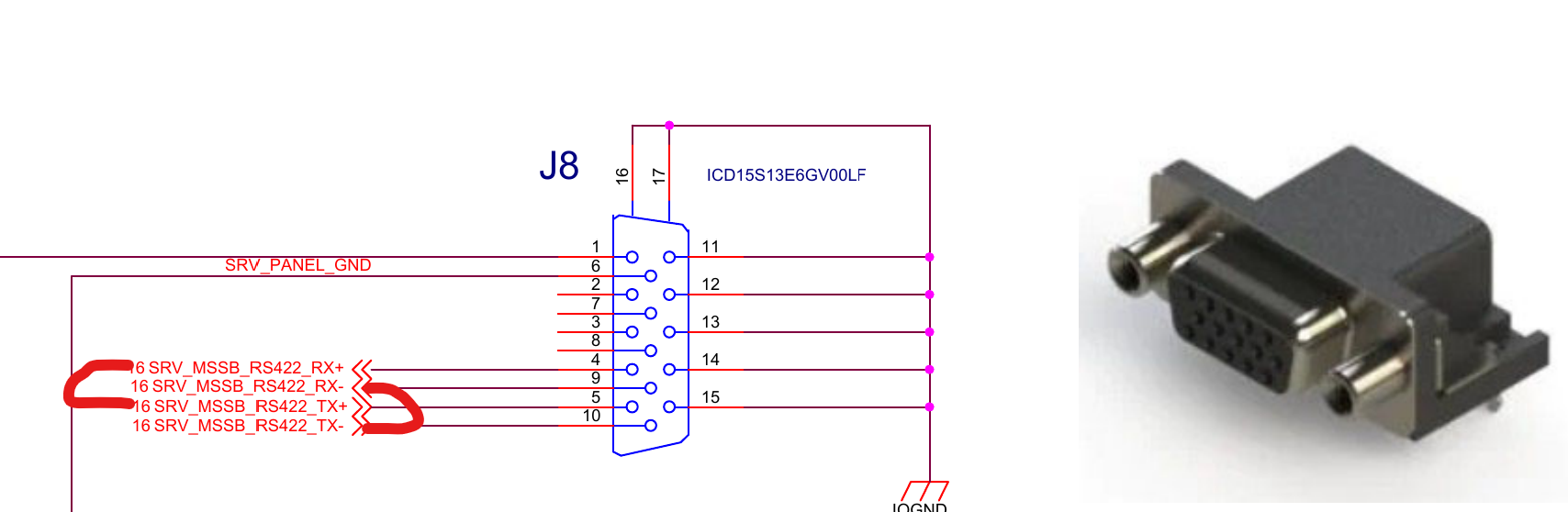
参考MSSB STN模块的测试，注意两者的基地址不同：

`define MSSB\_STN\_ADDR           32'h00090000        /\* MSSB address \*/

`define MSSB\_SRV\_ADDR           32'h00091000        /\* MSSB address \*/

**举例**

* Connect RS422\_RX+ to RS422\_TX+, pin-4 to pin-5
* Connect RS422\_RX- to RS422\_TX-, pin-9 to pin-10

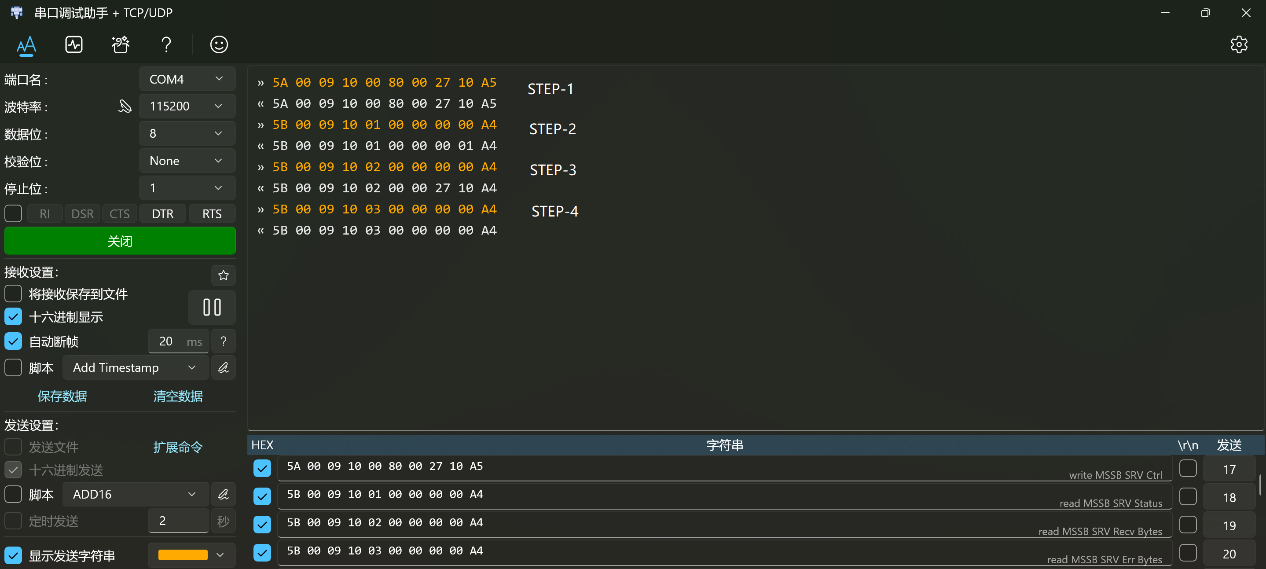


STEP-1. OPB Write to trigger MSSB Test, TRANS\_BYTES = 0x2710 (10K)

STEP-2. OPB Read for MSSB Status, the return value shall be 0x01

STEP-3. OPB Read for MSSB Recv Bytes, it shall be 0x2701 (10K)

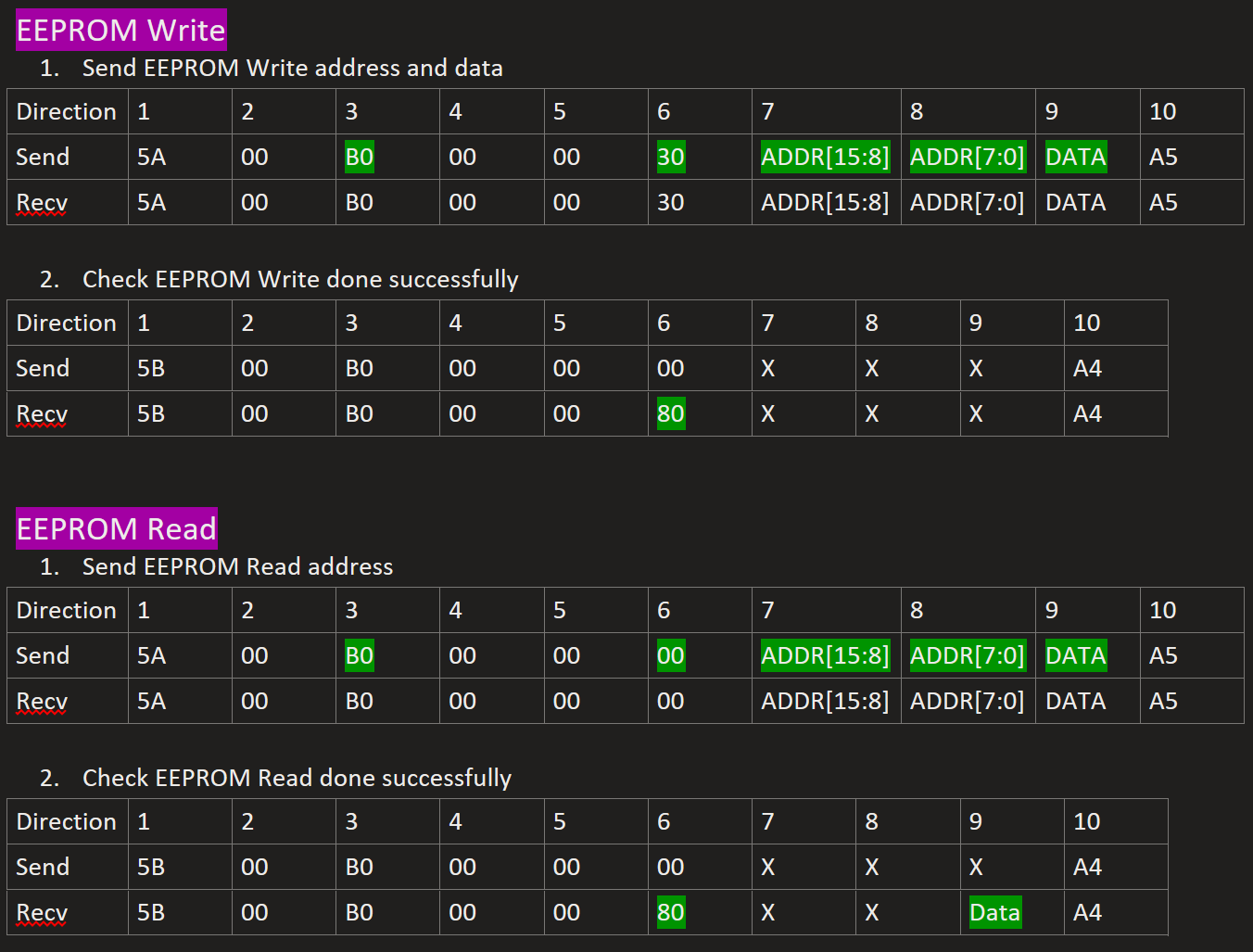
STEP-4. OPB Read for MSSB Err Bytes, it shall be 0



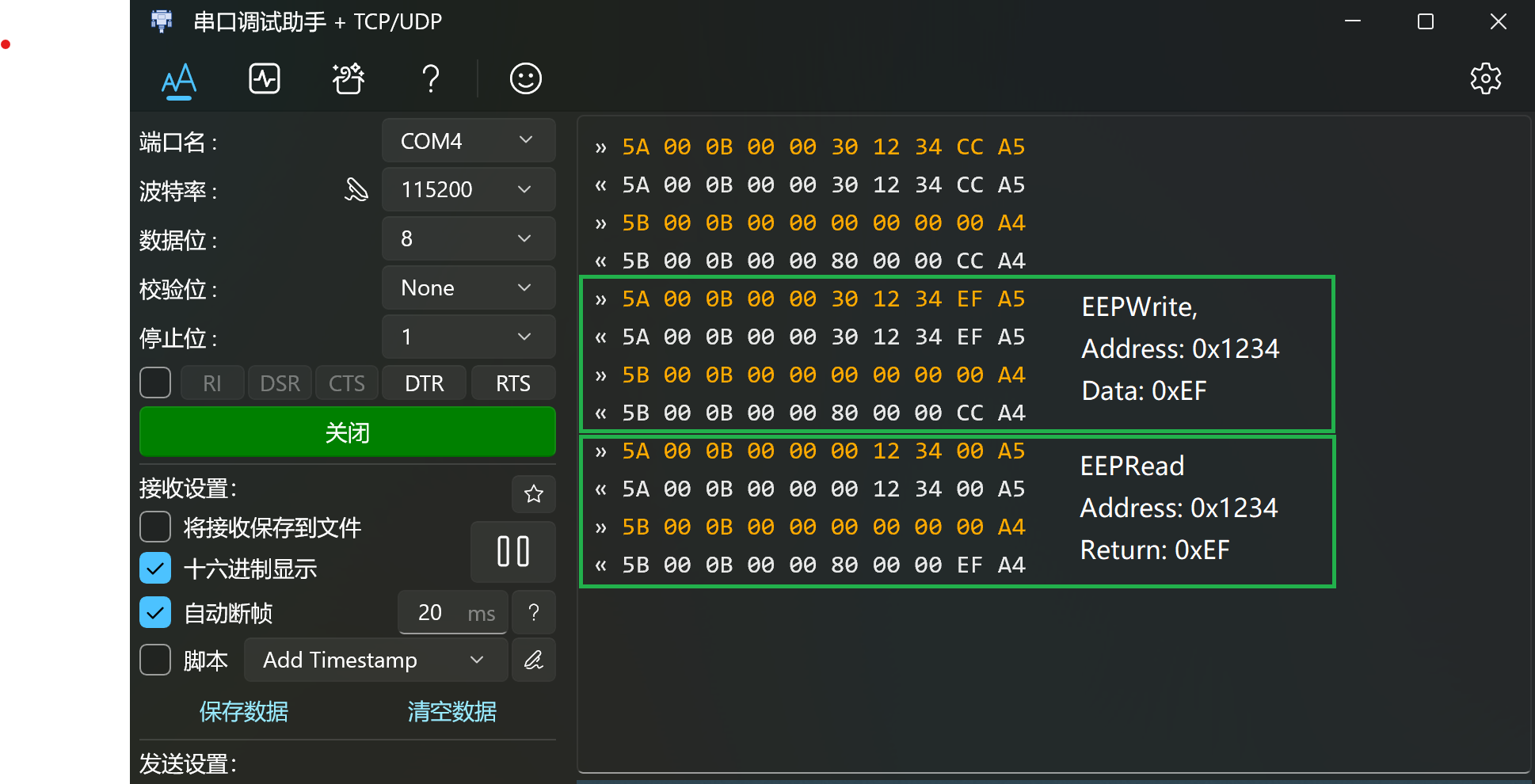
1. EEPROM模块

该模块用于EEPROM(U65)的测试。

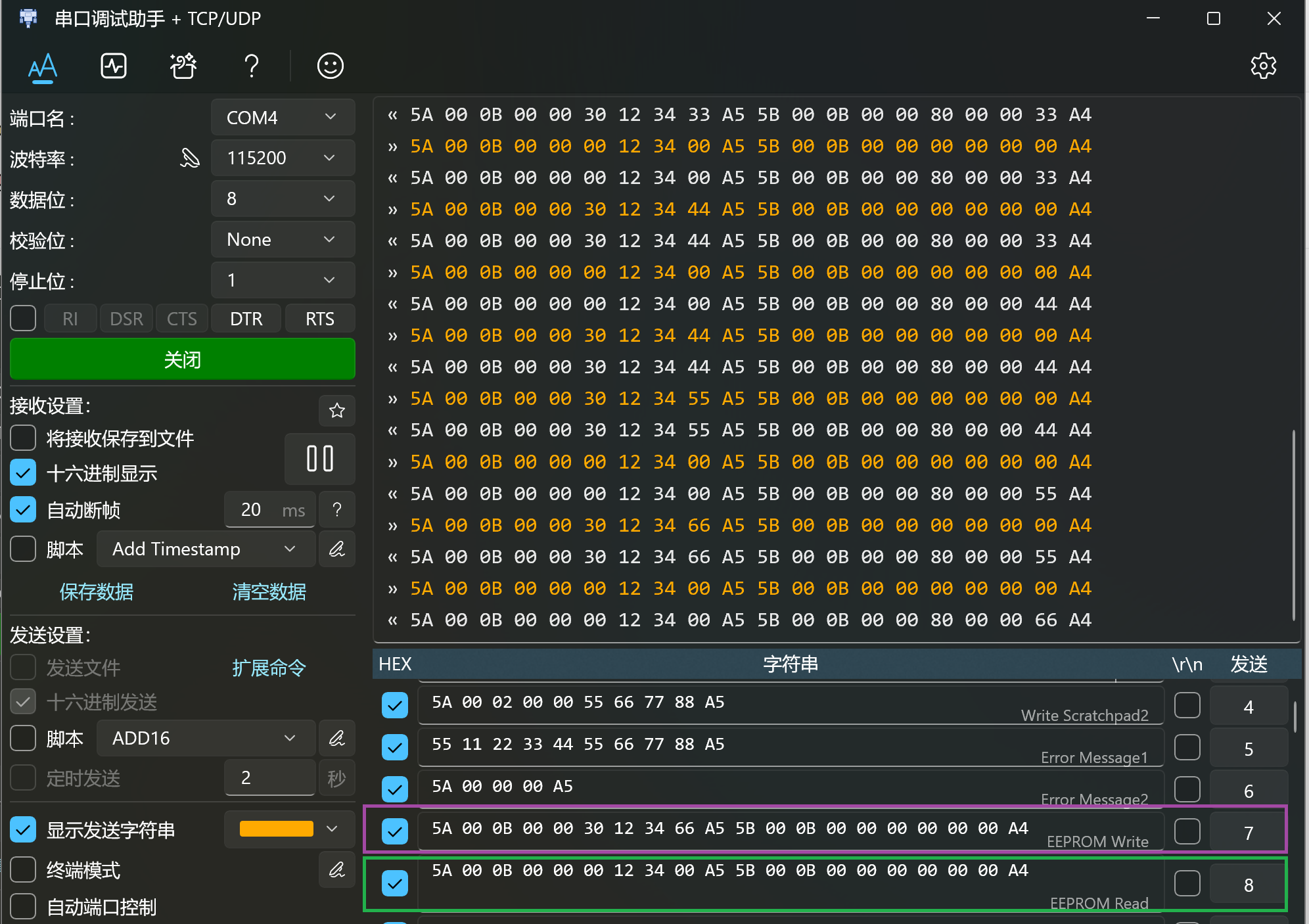




**Example**



为了节省测试时间，两条命令可以组装成一条命令，如下：



写了一段python测试EEPROM的读写时间，见eeprom\_test.py。先写入数据到一段EEPROM，然后读出并进行校验。注意：EEPROM 25XX640A的最大存储空间是8192 bytes. 测试结果如下：

|  |  |
| --- | --- |
| 数据长度 | 读写时间(second) |
| 128 | 1.32 |
| 1024 | 10.53 |
| 4096 | 42.32 |
| 8192 | 84.65 |

