# P1060966 HW&APP FPGA Register Map

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **作者** | **日期** | **描述** |
| 0.1 | XYL | 2025/8/1 | 基于原ATP的初始版本 |
|  |  |  |  |

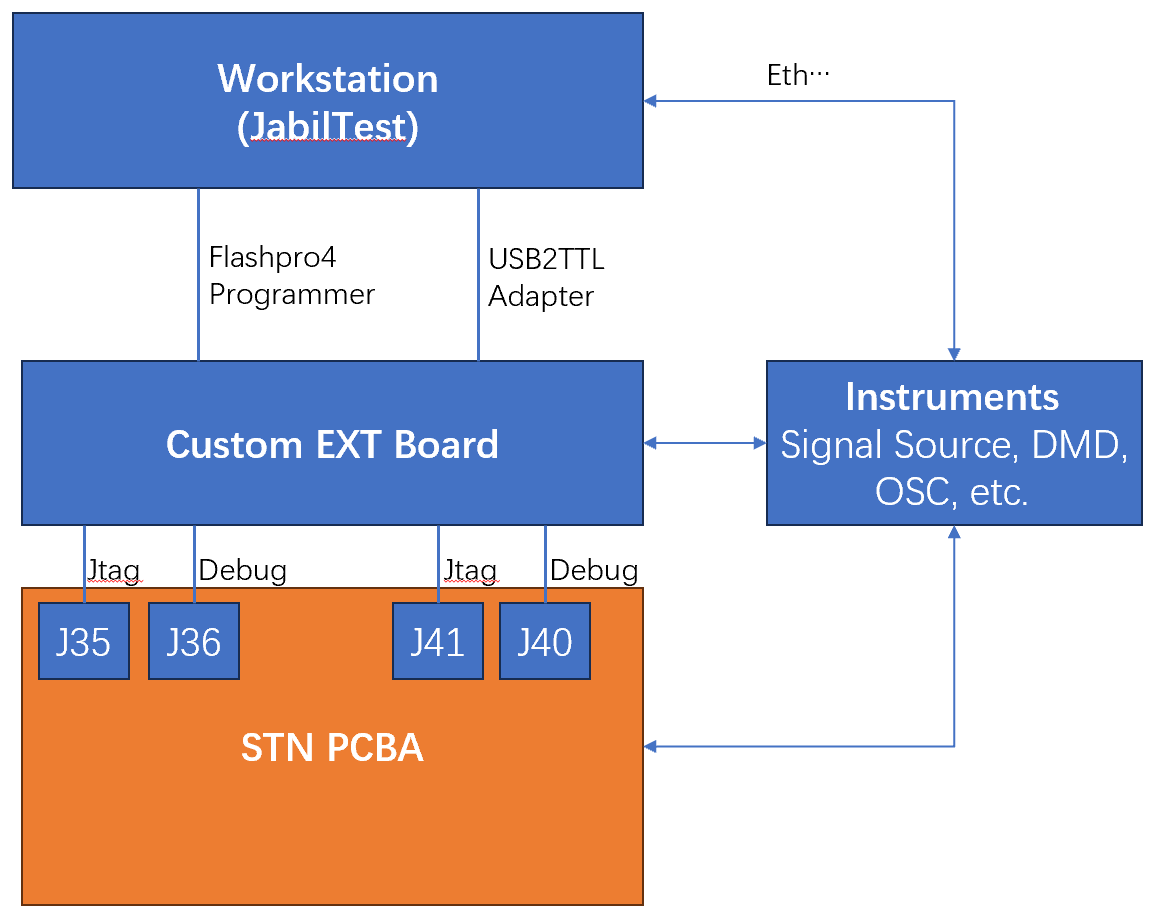
## **概要**

Stand HW FPGA和APP FPGA是完全独立的，各自维护一张寄存器表。

## **FPGA设计**

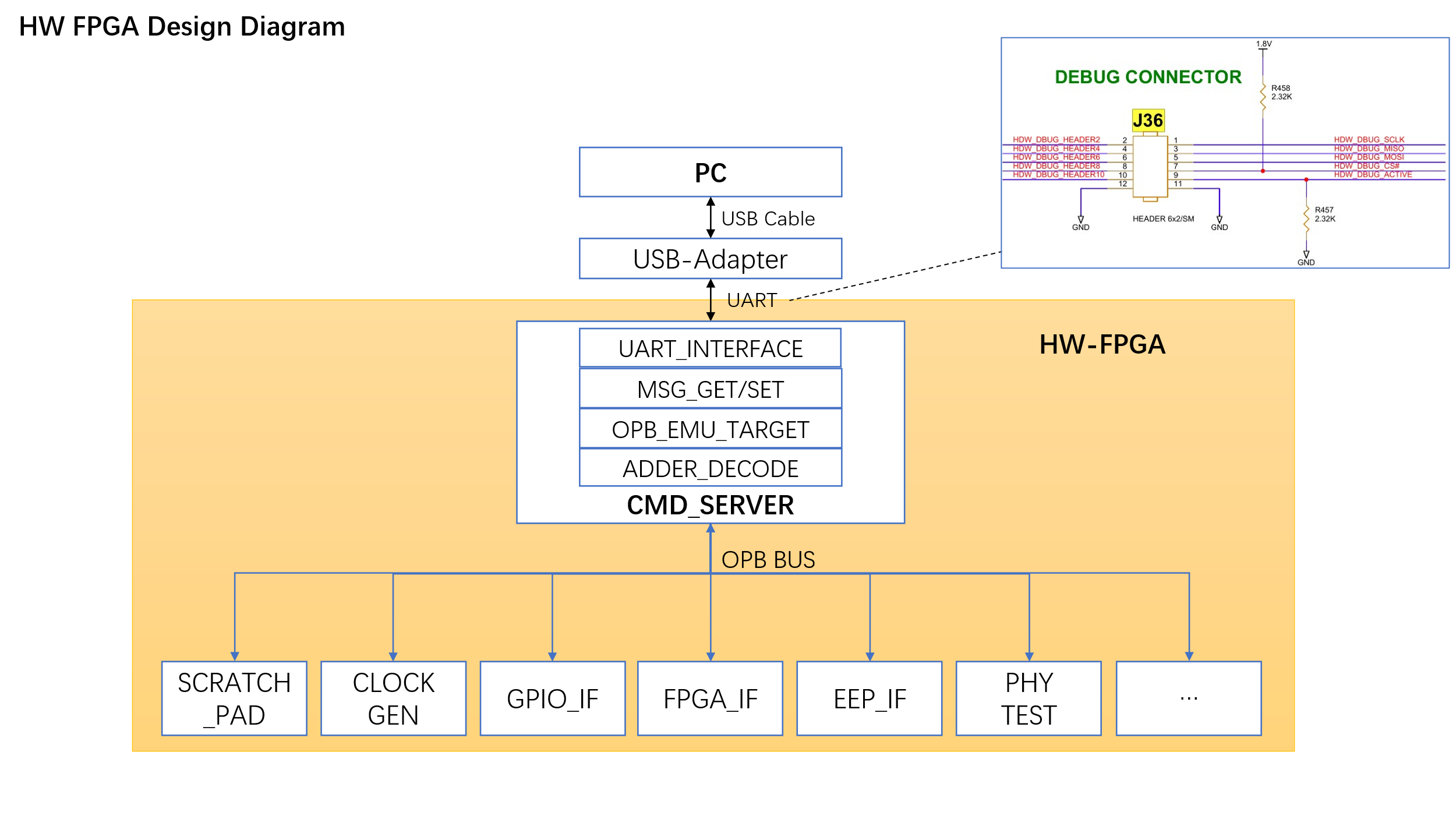
### **系统示意图**

STN PCBA测试系统示意图如下，上位机通过Debug connector J40，J36分别和APP FPGA、HW FPGA通信。



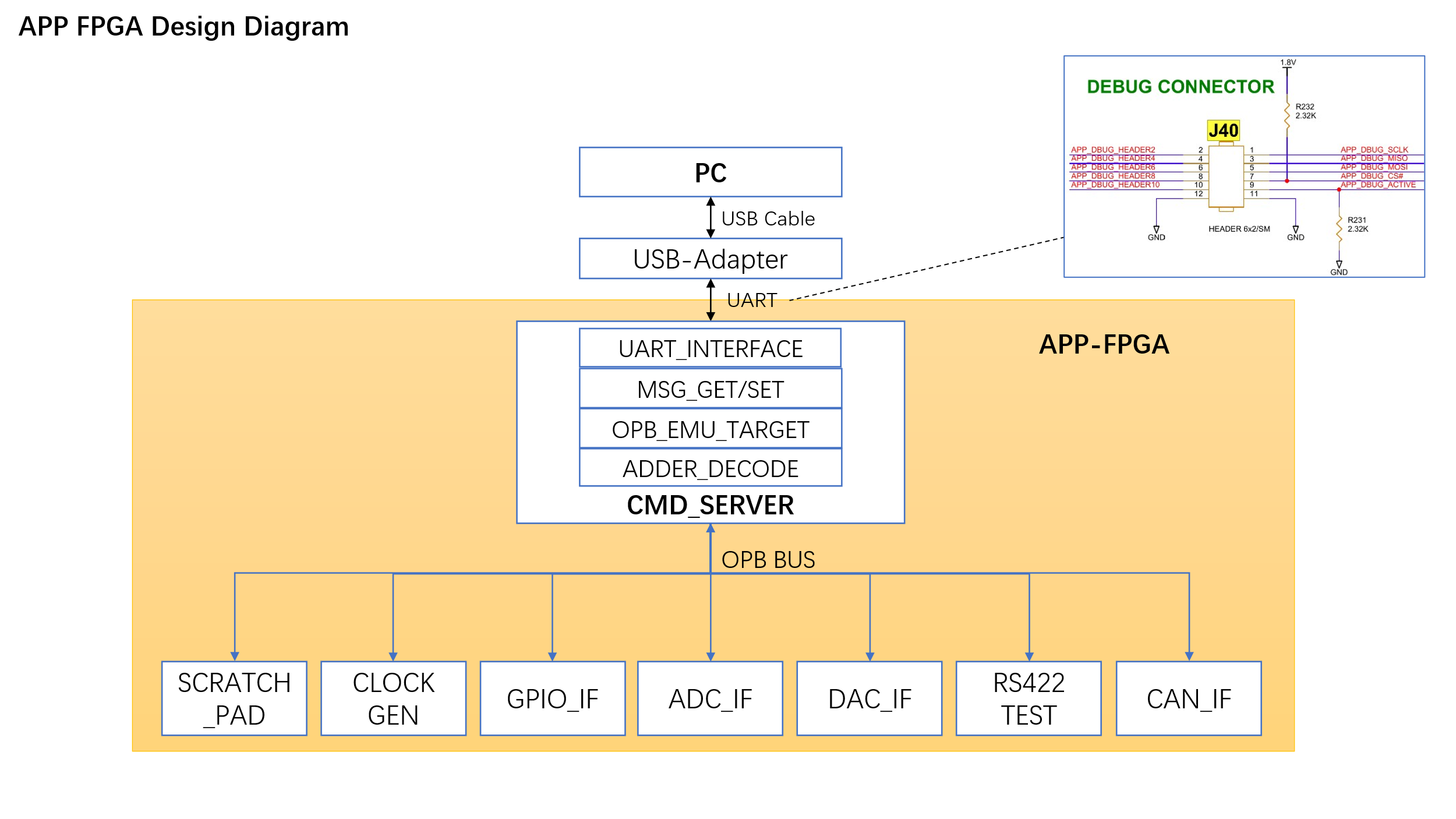
### **HW FPGA模块设计**

HW FPGA模块设计如下图，CMD\_SERVER负责和上位机通信，并将上位机的各种命令转换为OPB总线读、写操作；负责功能测试的子模块挂载到OPB总线，响应命令并执行测试任务，然后返回结果。J36-P1为UART-RXD，J36-P2为UART-TXD，然后通过USB-TTL1.8V Adapter和上位机通信。



### **APP FPGA模块设计**

APP FPGA模块设计和HW FPGA类似，如下图。J40-P1为UART-RXD，J40-P2为UART-TXD，然后通过USB-TTL1.8V Adapter和上位机通信。



### **通信方式**

APP和HW FPGA均采用UART和上位机通信，串口配置如下：

**$baudRate = 115200 bps; $byteSize = 8; $parity = "N"; $stopBits = 1;**

下载APP FPGA镜像后，D42 (HEARTBEAT LED) 会间隔1s闪烁，打开串口调试助手并配置正确后，每隔1s会收到FPGA发送的ping消息**“5A AA BB CC DD 11 22 33 44 A5”**，如下图所示。

**注意：**上位机成功发送任何一条格式正确的消息指令，FPGA会立即响应，并终止ping消息的发送。

****

## **HW-FPGA寄存器列表**

|  |  |  |
| --- | --- | --- |
| **地址** | **地址** | **说明** |
| SCRATCHPAD | 0x00000000 | FPGA型号，版本，发布日期等；读写测试寄存器 |
| CLOCK GEN | 0x00010000 | 配置内部时钟，默认不用修改 |
| OSC COUNTER1 | 0x00020000 | 100M时钟频率测试 |
| OSC COUNTER2 | 0x00021000 | 50M时钟频率测试 |
| GPIO | 0x00030000 | GPIO输入/输出，包括通用GPIO和Switches等 |
| FPGA IF | 0x00040000 | HW和APP FPGA之间的接口信号互联测试 |
| EEPROM | 0x00050000 | EEPROM读写测试 |
| PHY | 0x00060000 | PHY通信模块测试 |
|  |  |  |

### **SCRATCHPAD子模块**

* + 1. **寄存器列表**

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：

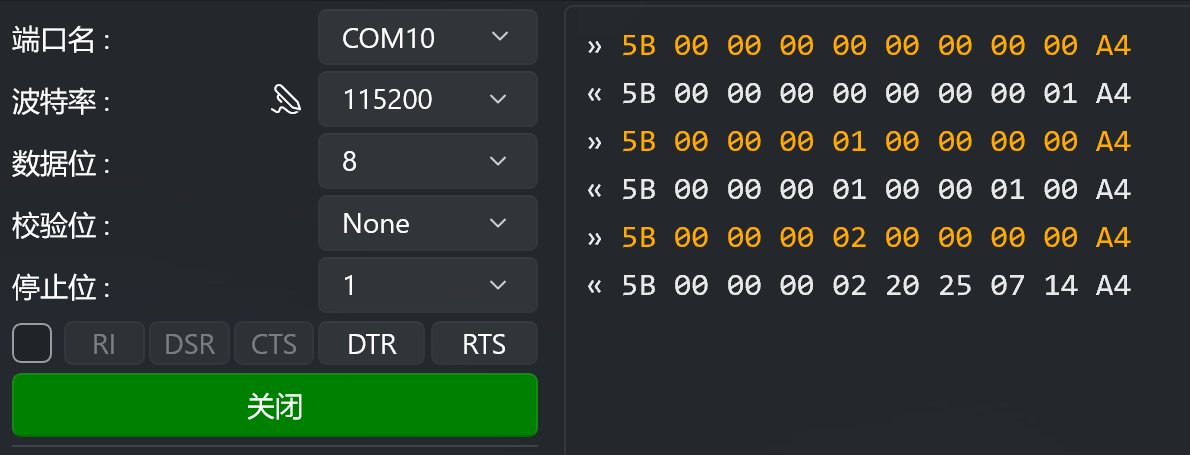
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **偏移地址** | **读写** | **位宽** | **说明** | **复位值** |
| **Version** | **0** | **R** | **32位** | **FW Version** | **0x0000\_0001** |
| **ID** | **1** | **R** | **32位** | **FPGA ID** | **0x0000\_0050** |
| **Date** | **2** | **R** | **32位** | **Build Date** | **0x2025\_0714** |
| **dev\_sp1** | **3** | **R/W** | **32位** | **Scratch Pad 1** | **0x12345678** |
| **dev\_sp2** | **4** | **R/W** | **32位** | **Scratch Pad 2** | **0x9abcbeef** |

* + 1. **说明**
* 该模块定义了一些只读寄存器包含FPGA型号，版本，发布日期等
* 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 **Version** @0x0000\_0000

Step-2. OPB Read **FPGA\_ID** @0x0000\_0001

Step-3. OPB Read **Build\_Date** @0x0000\_0002



Step-5. OPB Read **dev\_sp1** @0x0000\_0003

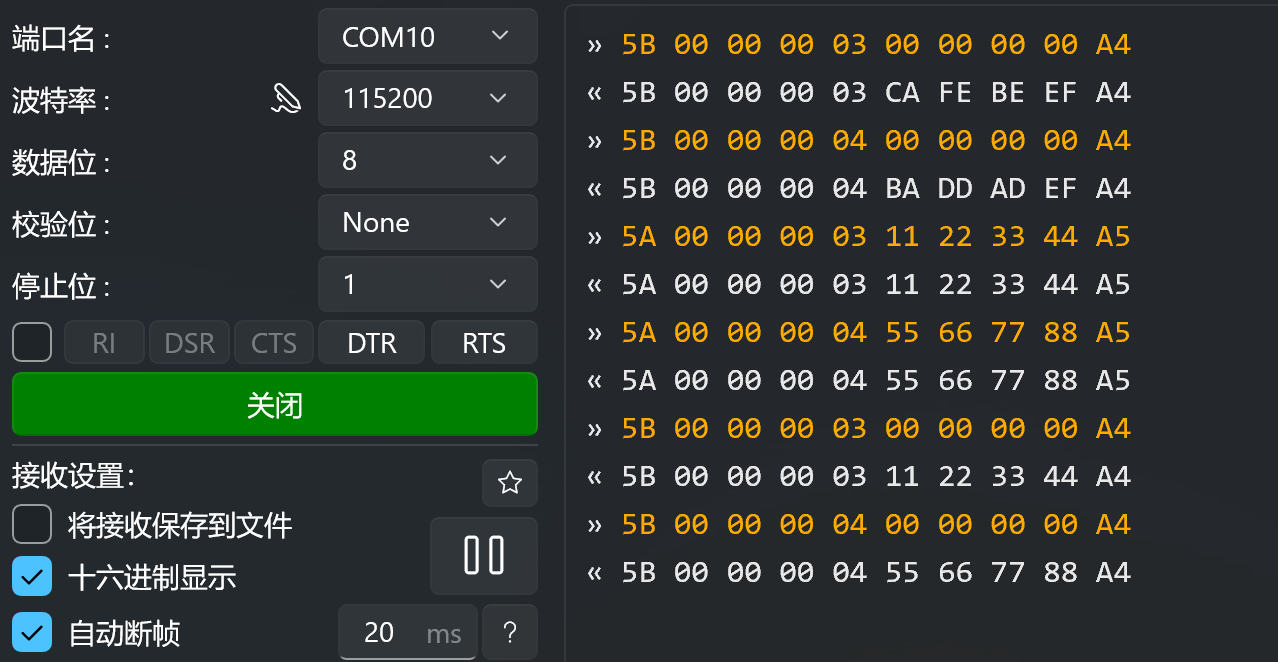
Step-6. OPB Read **dev\_sp2** @0x0000\_0004

Step-4. OPB Write **dev\_sp1** @0x0000\_0003

Step-4. OPB Write **dev\_sp2** @0x0000\_0004

Step-5. OPB Read **dev\_sp1** @0x0000\_0003

Step-6. OPB Read **dev\_sp2** @0x0000\_0004



### **CLOCK GEN子模块**

**注意：该模块用于配置内部时钟参数，无需修改！**

* + 1. 寄存器列表如下：

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

### **OSC COUNTER1子模块**

该模块用于测试U61 (100MHz Oscollator)。

注意：STN ATP没有这项测试要求。

* + 1. **寄存器列表**

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

* + 1. **详细说明**
* 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）
* 计数寄存器（COUNTR, 0x1）
  + [15:0] 计数值（只读）
* Scratch Pad寄存器（SPR, 0x2）
  + [31:0] 通用数据寄存器（可读写）
    1. **示例**

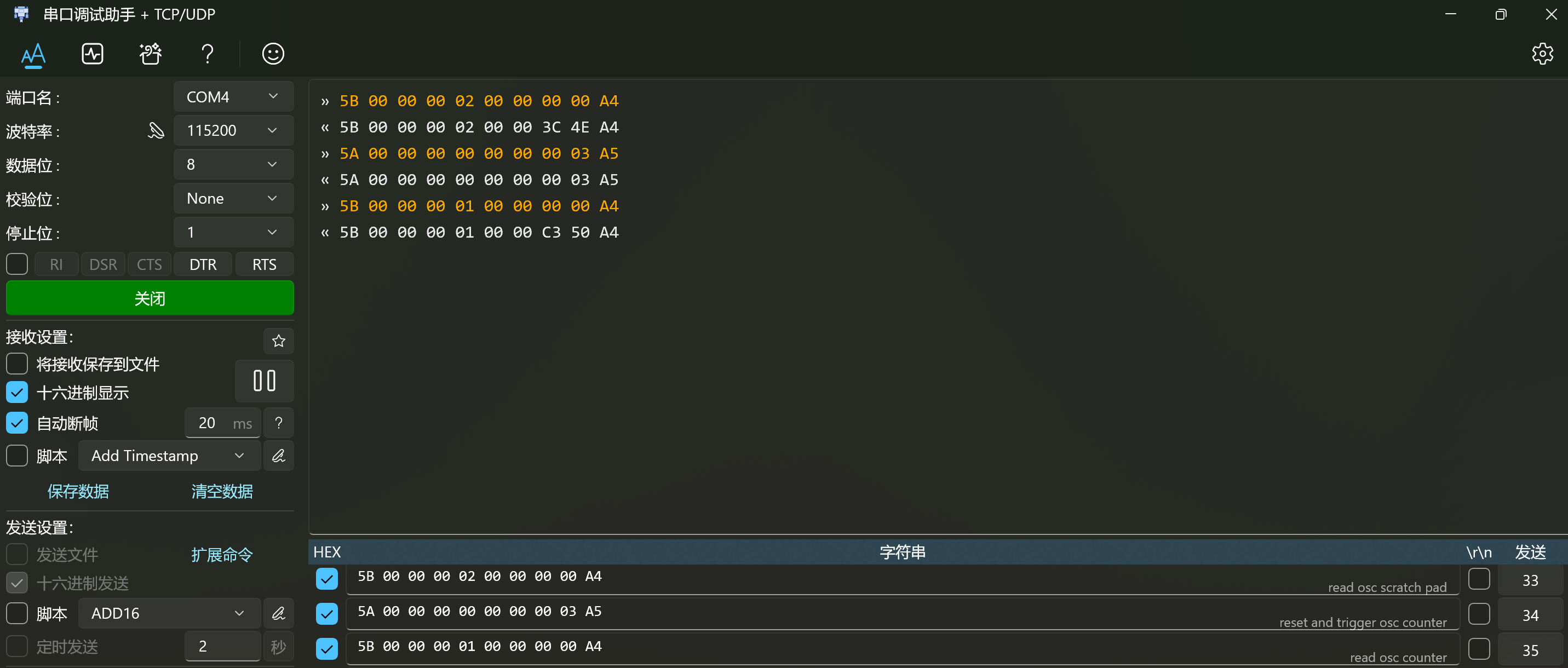
Step-1. OPB Read Scratch Pad Register@(0x0002\_0002), return 32’h00003c4e

Step-2. OPB Write 32’h3 to CNTRL Register@(0x0002\_0000)

Step-3. OPB Read Counter Register @(0x0002\_0001), check return value

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

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



### **OSC COUNTER2子模块**

该模块用于测试U48 (50MHz Oscillator)。

注意：STN ATP没有这项测试要求。

* + 1. **寄存器列表**

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

* + 1. **详细说明**
* 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）
* 计数寄存器（COUNTR, 0x1）
  + [15:0] 计数值（只读）
* Scratch Pad寄存器（SPR, 0x2）
  + [31:0] 通用数据寄存器（可读写）
    1. **示例**

参考3.3示例

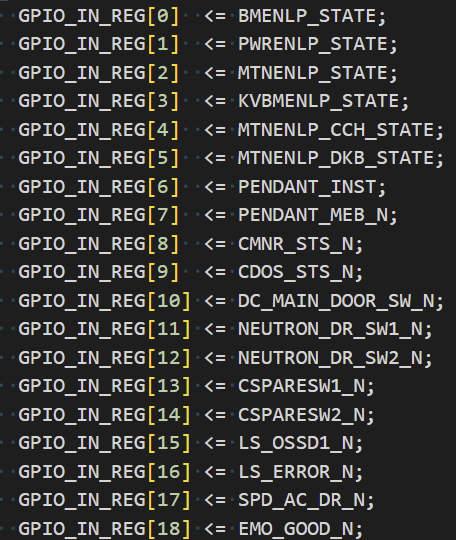
### **GPIO子模块**

该模块用于读写FPGA GPIO。

* + 1. **寄存器列表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 地址（宏定义） | 偏移地址 | 读写 | 位宽 | 说明 | 复位值 |
| GPIO\_IN\_ADDR | 0x0 | R | 32位 | 通用GPIO输入 | NA |
| SWITCH\_IN\_ADDR | 0x1 | R | 32位 | 编码开关输入 | NA |
| GPIO\_OUT\_ADDR | 0x2 | R/W | 32位 | 通过GPIO输出 | 32’h0000\_0000 |
| TP\_OUT\_ADDR | 0x3 | R/W | 32位 | 测试点输出 | 32’h0000\_0000 |

* + 1. **GPIO IN**

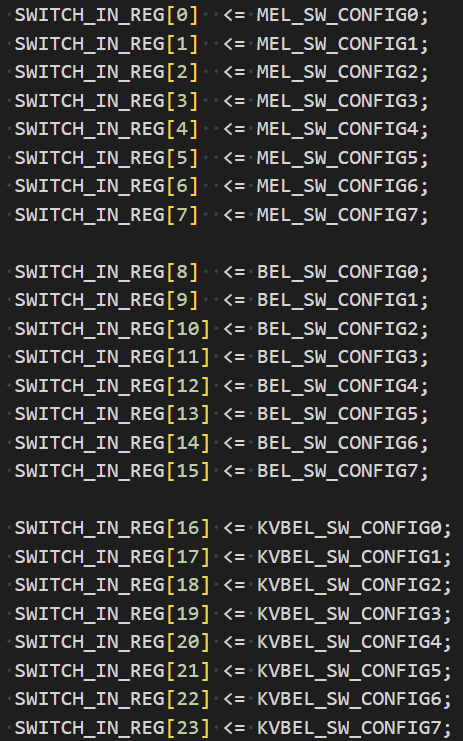


如下OPB\_Read(0x00030000)命令读取GPIO\_IN\_REG，即

5B 00 03 00 00 00 00 00 A4

示例如下：

* + 1. **SWITCH IN**

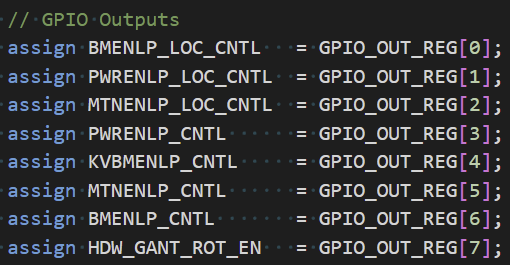


如下OPB\_Read(0x00030001)命令读取GPIO\_IN\_REG，即

5B 00 03 00 01 00 00 00 00 A4

示例如下：

* + 1. **GPIO OUT**

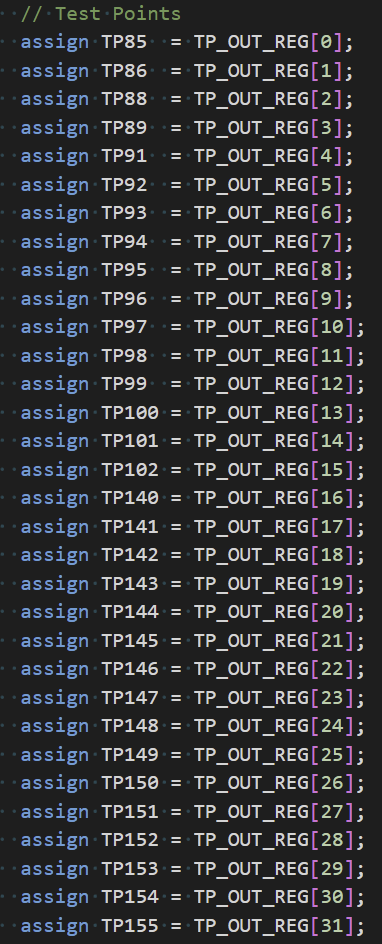


如下OPB\_Write(0x00030002, 0xFF)命令设置8个GPIO Outputs为1：

5A 00 03 00 02 00 00 00 FF A5

示例如下：

* + 1. **TESTPOINTS OUT**



如下OPB\_Write(0x00030003, 0xFFFFFFFF)命令设置所有TestPoints为1：

5A 00 03 00 03 FF FF FF FF A5

示例如下：

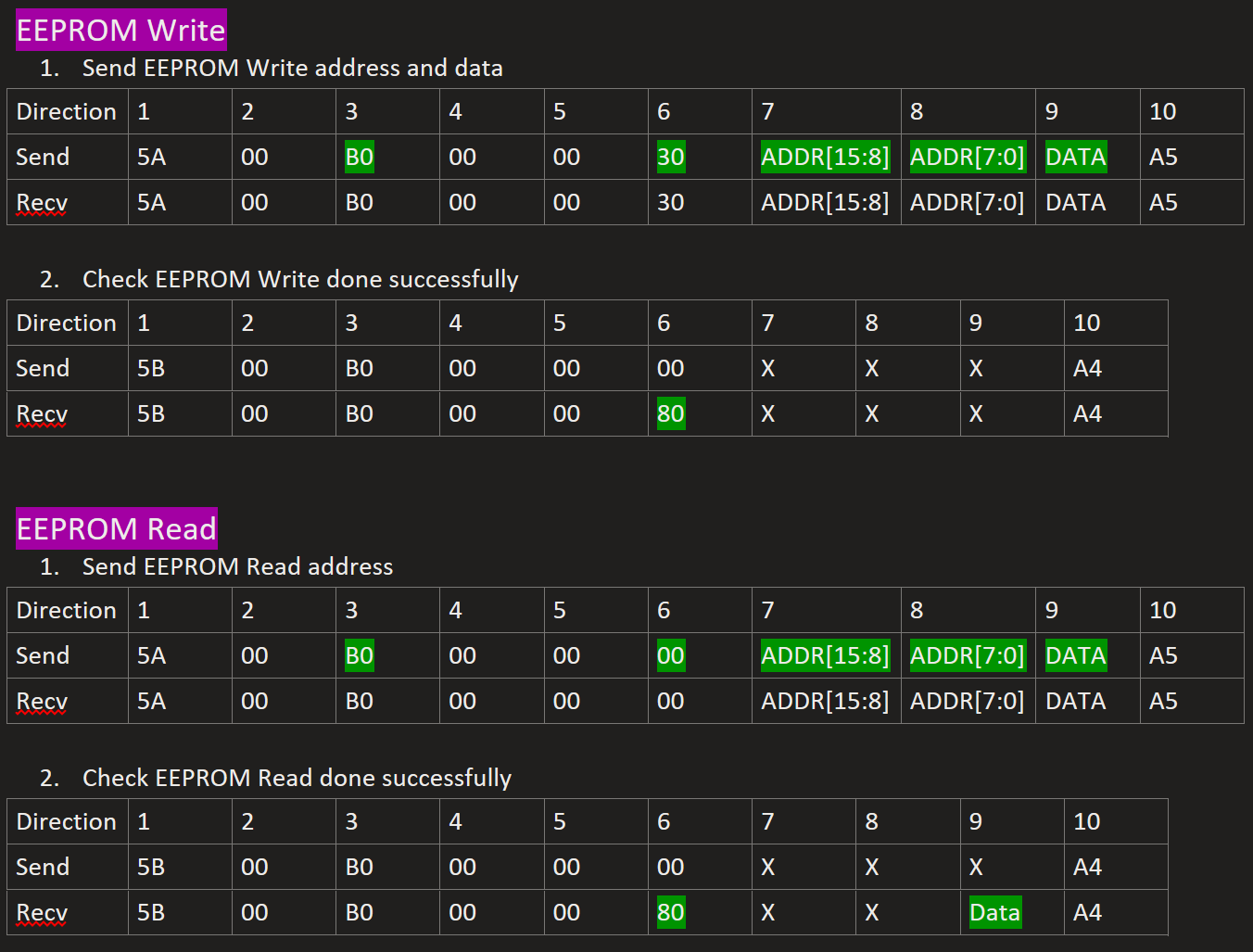
### **FPGA IF子模块 (TODO)**

### **EEPROM子模块**

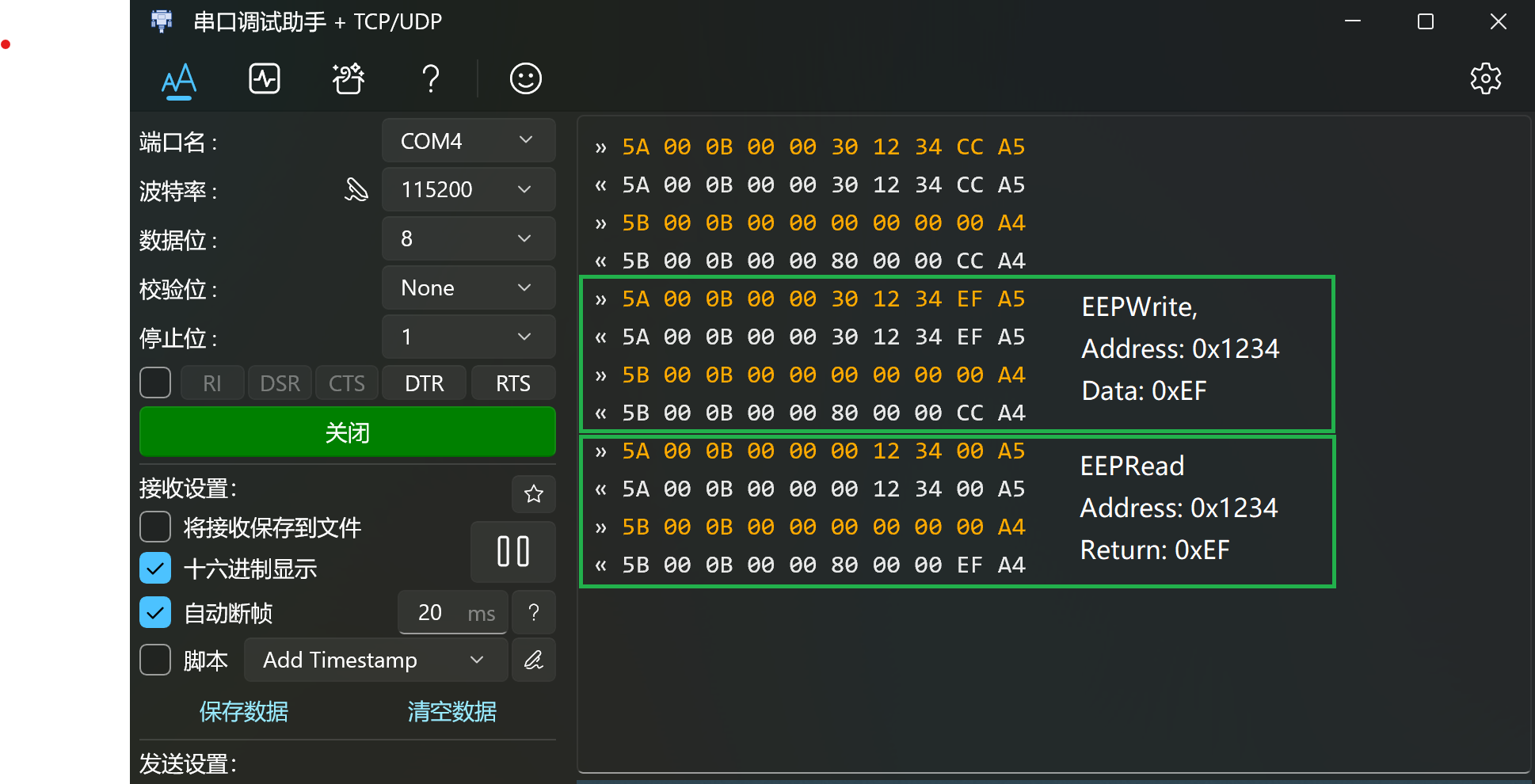
该模块用于EEPROM(U46)的测试。

注意：OPB\_ADDR应该修改为32’H0005\_0000

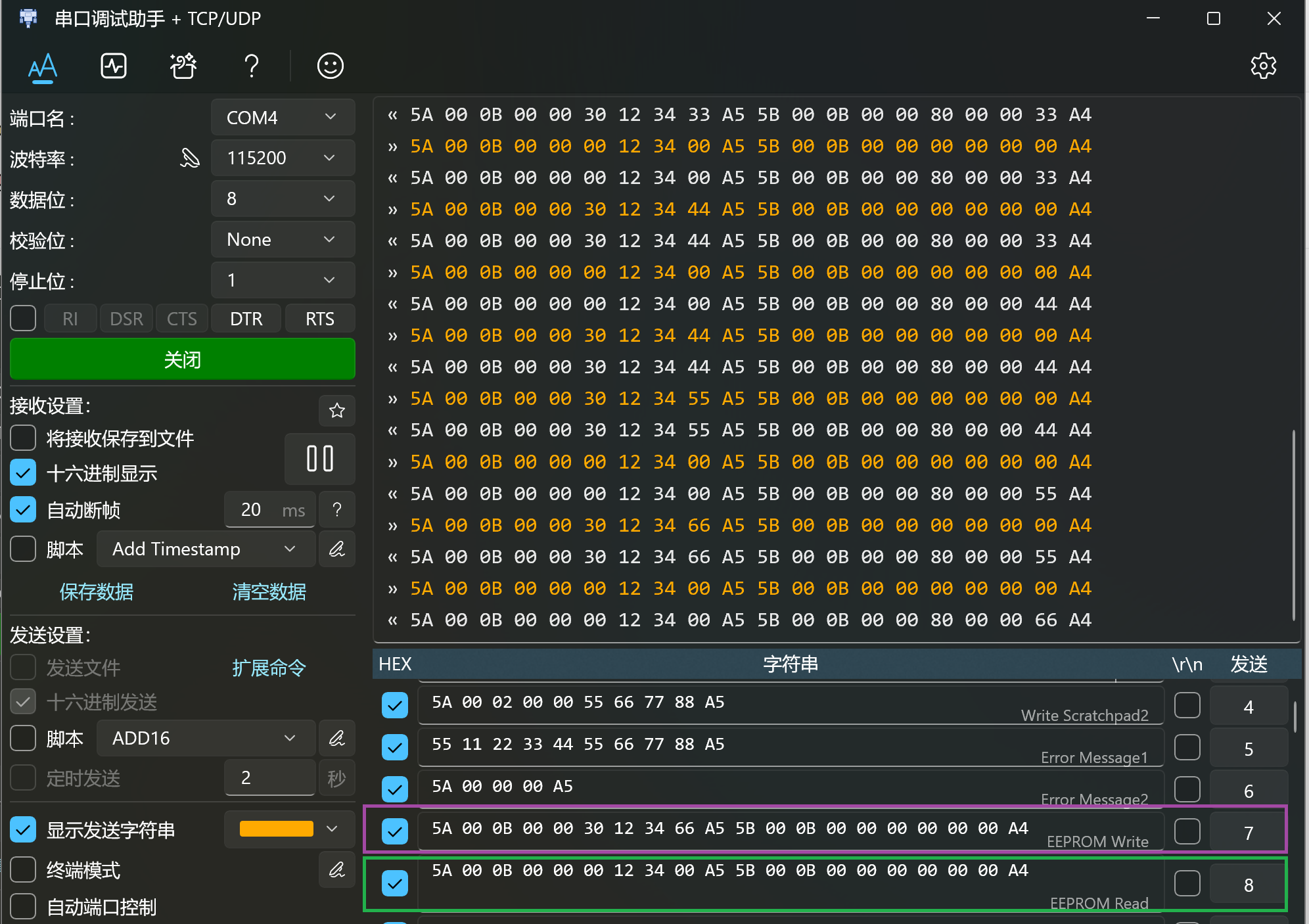




**Example**

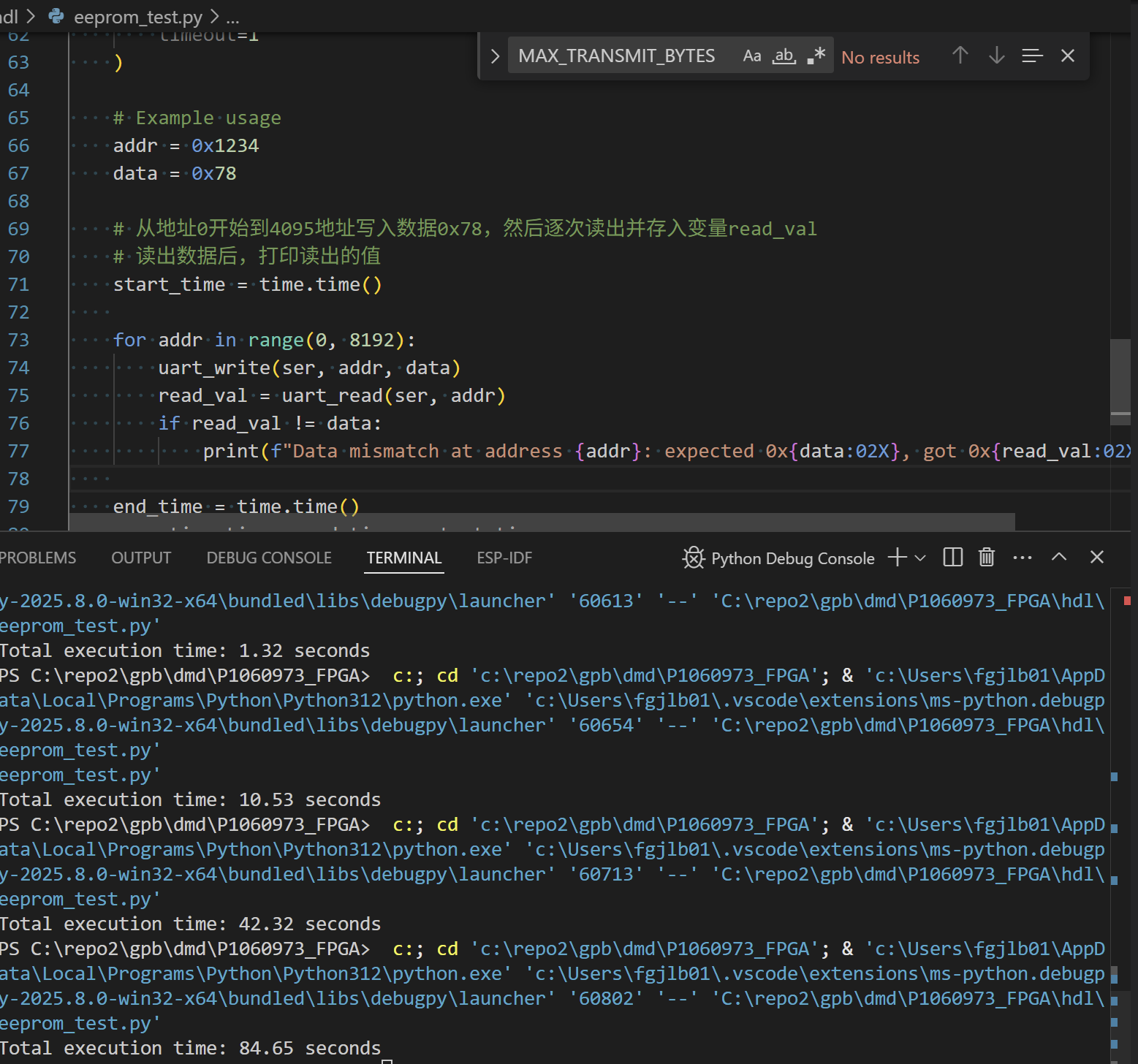


为了节省测试时间，两条命令可以组装成一条命令，如下：



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 |



### **PHY子模块 (TODO)**

## **APP-FPGA寄存器列表**

|  |  |  |
| --- | --- | --- |
| **地址** | **地址** | **说明** |
| SCRATCHPAD | 0x00000000 | FPGA型号，版本，发布日期等；读写测试寄存器 |
| CLOCK GEN | 0x00010000 | 配置内部时钟，默认不用修改 |
| OSC COUNTER | 0x00020000 | 100M/50M时钟频率测试 |
| GPIO | 0x00030000 | GPIO输入/输出，包括通用GPIO和Switches等 |
| FPGA IF | 0x00040000 | HW和APP FPGA之间的接口信号互联测试 |
| ADC | 0x00050000 | ADC ADS8864I相关测试 |
| DAC | 0x00060000 | DAC60504相关测试 |
| RS422 | 0x00070000 | RS422/485通信测试 |
| CAN | 0x00080000 | CAN BUS通信测试 |

### **SCRATCHPAD子模块**

这个模块用于测试上位机和FPGA之间的通信。

* + 1. **寄存器列表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **偏移地址** | **读写** | **位宽** | **说明** | **复位值** |
| **Version** | **0** | **R** | **32位** | **FW Version** | **0x0000\_0001** |
| **ID** | **1** | **R** | **32位** | **FPGA ID** | **0x0000\_0100** |
| **Date** | **2** | **R** | **32位** | **Build Date** | **0x2025\_0714**  **(随版本变化)** |
| **dev\_sp1** | **3** | **R/W** | **32位** | **Scratch Pad 1** | **0x12345678** |
| **dev\_sp2** | **4** | **R/W** | **32位** | **Scratch Pad 2** | **0x9abcbeef** |

* + 1. **说明**
* 该模块定义了一些只读寄存器包含FPGA型号，版本，发布日期等
* 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**

参考HW FPGA测试章节3.1.

### **CLOCK GEN子模块**

**注意：该模块用于配置内部时钟参数，无需修改！**

* + 1. **寄存器列表**

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

### **OSC COUNTER子模块**

该模块用于测试U58 (100MHz Oscollator)。

注意：STN ATP没有这项测试要求。

* + 1. **寄存器列表**

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

* + 1. **详细说明**
* 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）
* 计数寄存器（COUNTR, 0x1）
  + [15:0] 计数值（只读）
* Scratch Pad寄存器（SPR, 0x2）
  + [31:0] 通用数据寄存器（可读写）
    1. **示例**

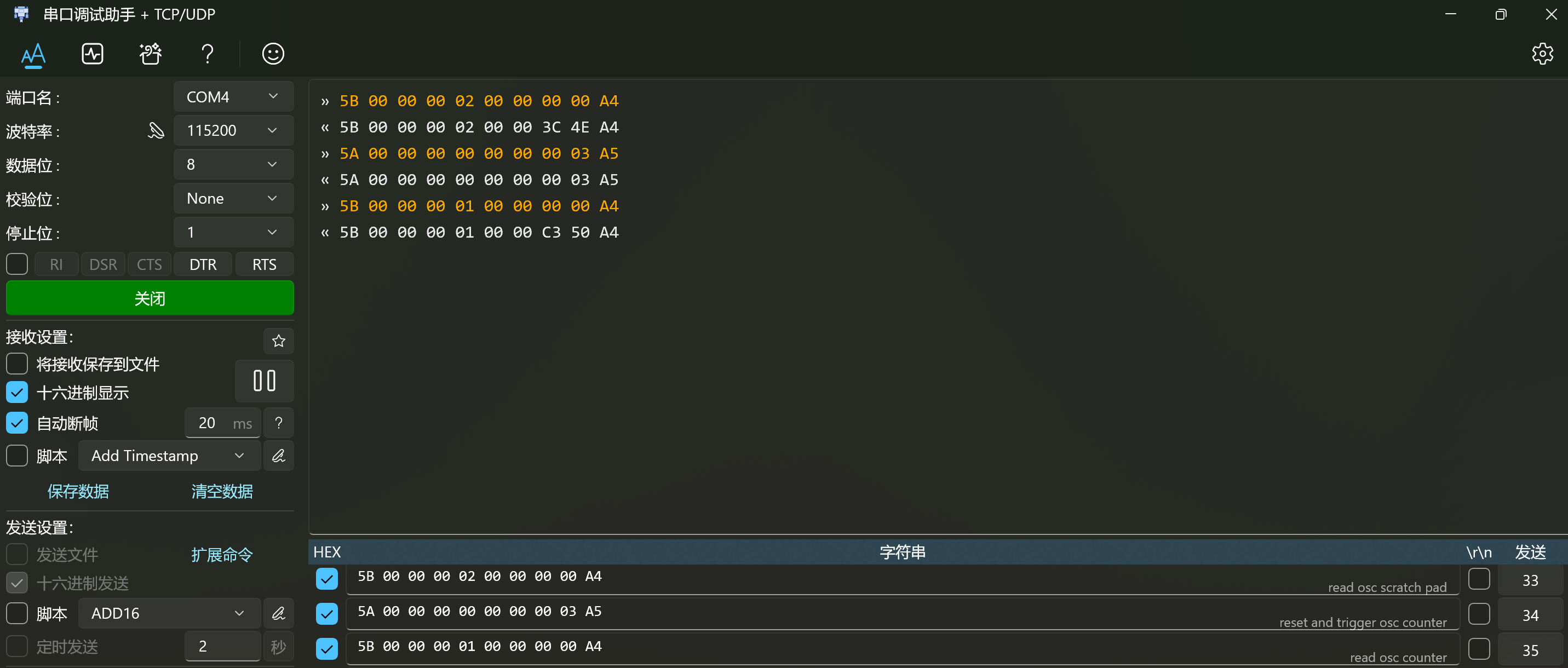
Step-1. OPB Read Scratch Pad Register@(0x00020002), return 32’h00003c4e

Step-2. OPB Write 32’h3 to CNTRL Register@(0x00020000)

Step-3. OPB Read Counter Register @(0x00020001), check return value

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

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



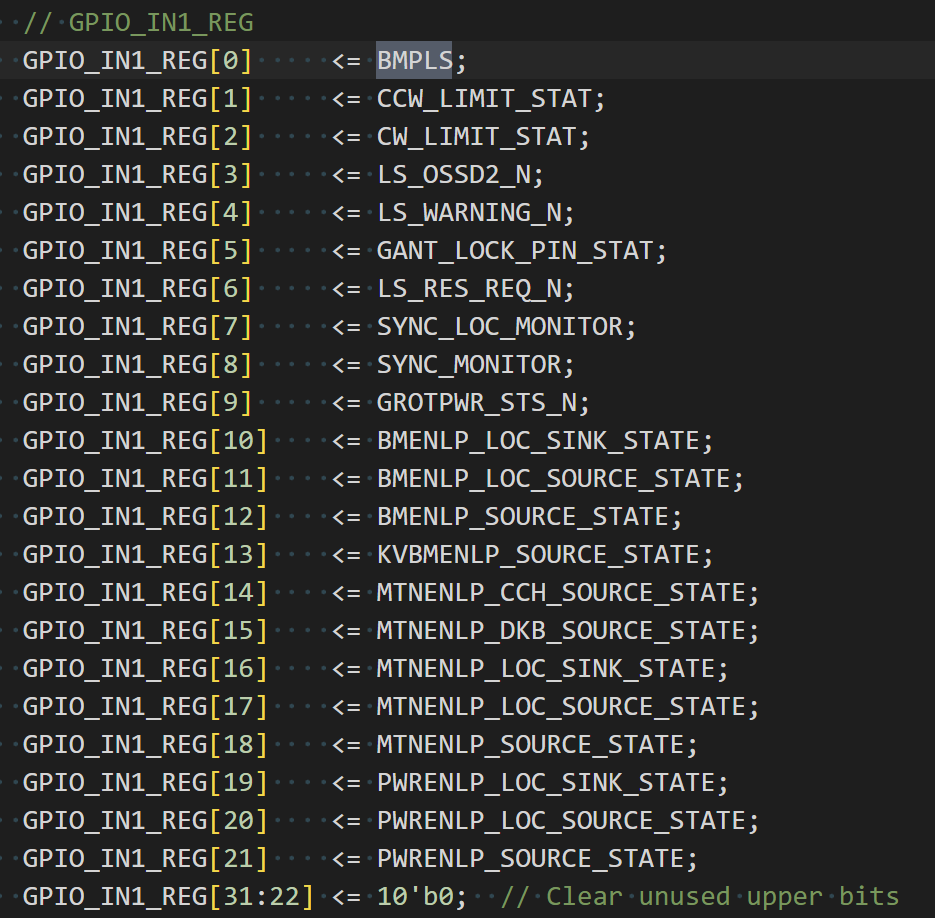
### **GPIO子模块**

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） | 偏移地址 | 读写 | 位宽 | 说明 | 复位值 |
| GPIO\_IN1\_ADDR | 0x0 | R | 32位 | 通用GPIO输入 | NA |
| GPIO\_IN2\_ADDR | 0x1 | R | 32位 | 通用GPIO输入 | NA |
| GPIO\_OUT1\_ADDR | 0x2 | R/W | 32位 | 通过GPIO输出 | 32’h0000\_0000 |
| GPIO\_OUT2\_ADDR | 0x3 | R/W | 32位 | 通过GPIO输出 | 32’h0000\_0000 |
| TP\_IN1\_ADDR | 0x4 | R | 32位 | 测试点输出 | NA |
| TP\_IN2\_ADDR | 0x5 | R | 32位 | 测试点输出 | NA |

* + 1. **GPIO\_IN1**

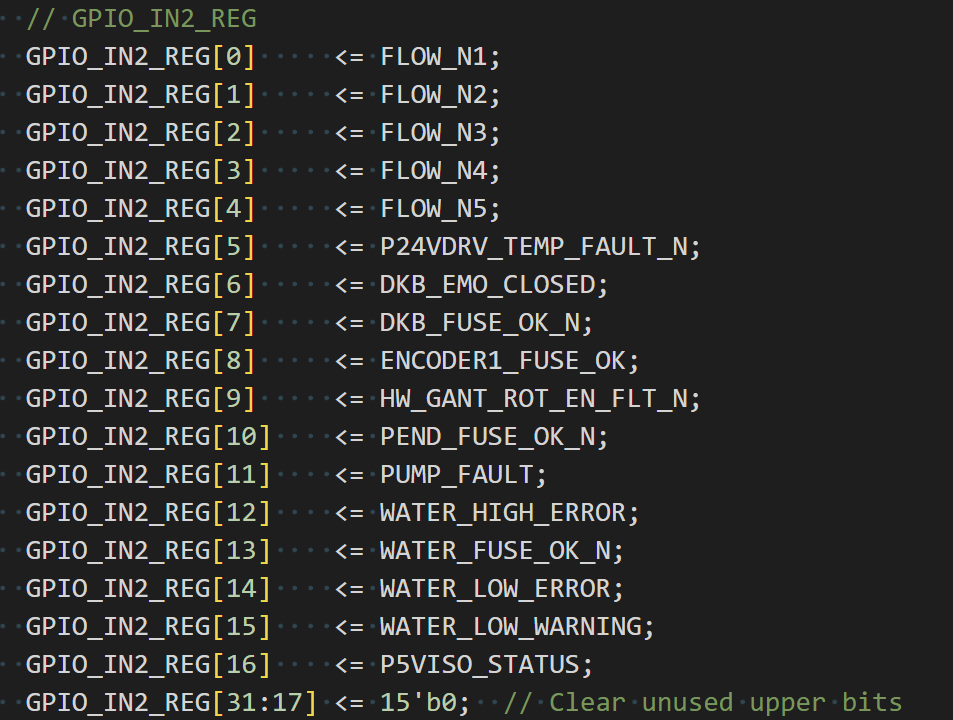
****

如下OPB\_Read(0x00030000)命令读取GPIO\_IN1，即

**5B 00 03 00 00 00 00 00 A4**

示例如下：

* + 1. **GPIO\_IN2**

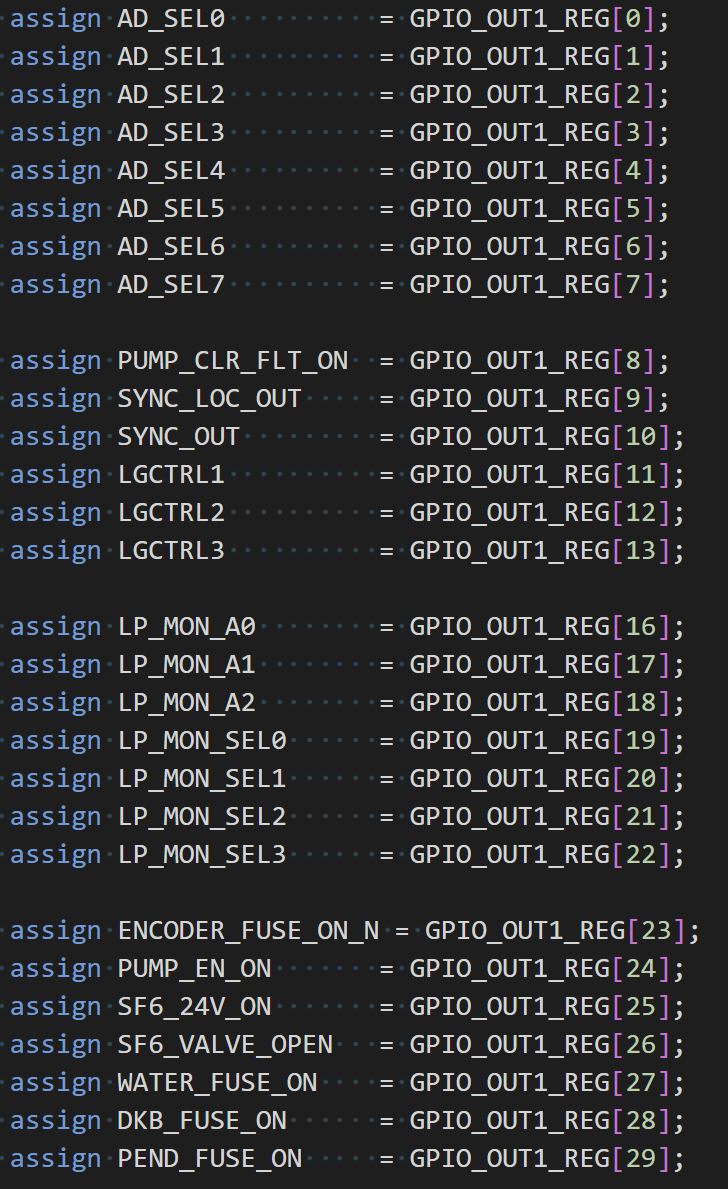
****

如下OPB\_Read(0x00030001)命令读取GPIO\_IN2，即

**5B 00 03 00 01 00 00 00 A4**

示例如下：

* + 1. **GPIO Out1**

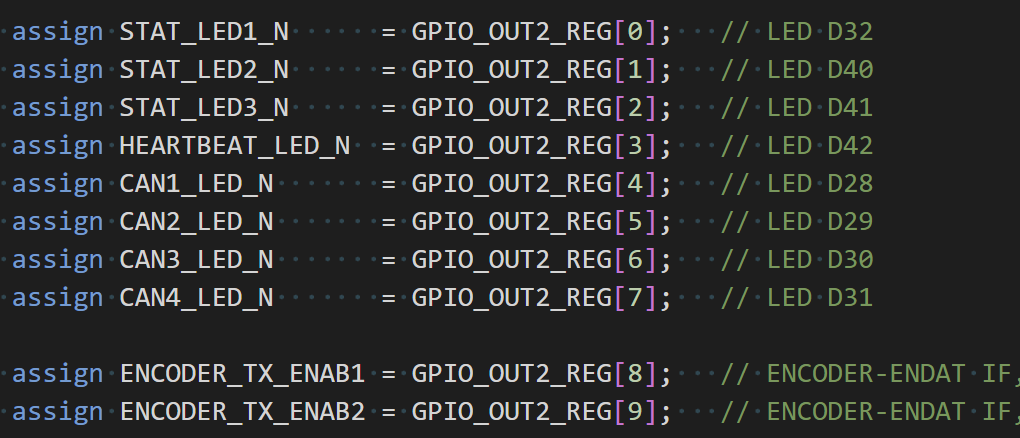
****

如下OPB\_Write(0x00030002, 0xFFFFFFFF)命令设置所有GPIO Outputs为1：

**5A 00 03 00 02 FF FF FF FF A5**

示例如下：

* + 1. **GPIO Out2**

****

如下OPB\_Write(0x00030003, 0x000003FF)命令设置所有GPIO Outputs为1：

**5A 00 03 00 03 00 00 03 FF A5**

示例如下：

### **FPGA IF子模块 (TODO)**

### **ADC模块**

该模块用于ADC ADS8864的测试, 覆盖ATP测试项**8.4 ADC Test。**

* + 1. **寄存器列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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4.4.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



### **DAC模块**

该模块用于DACx0504的测试，覆盖ATP测试项**8.6 DAC Tests。**

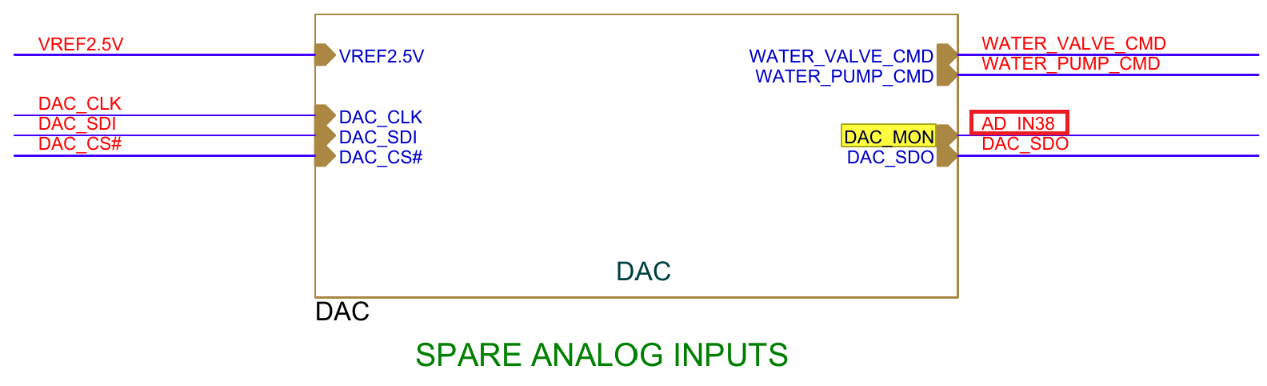
* + 1. **寄存器列表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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**

写DAC0/1/2/3数据寄存器，DAC的输出对应的电平Out = Data/65536\*5

* + 1. **举例：DAC-ADC环回测试，选通DAC\_MON(AD\_IN38)作为ADC的输入**



STEP-1. OPB Write DAC data for VOUT3

* Measure Voltage @TP69 (2.50V for DAC data = 0xFFFF )
* Measure Voltage @TP69 (1.25V for DAC data = 0x8000 )

STEP-2. OPB Write to select AD\_IN38(DAC\_MON) as ADC input

* Measure Voltage @TP62 (2.50V for DAC data = 0xFFFF )
* Measure Voltage @TP62 (1.25V 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

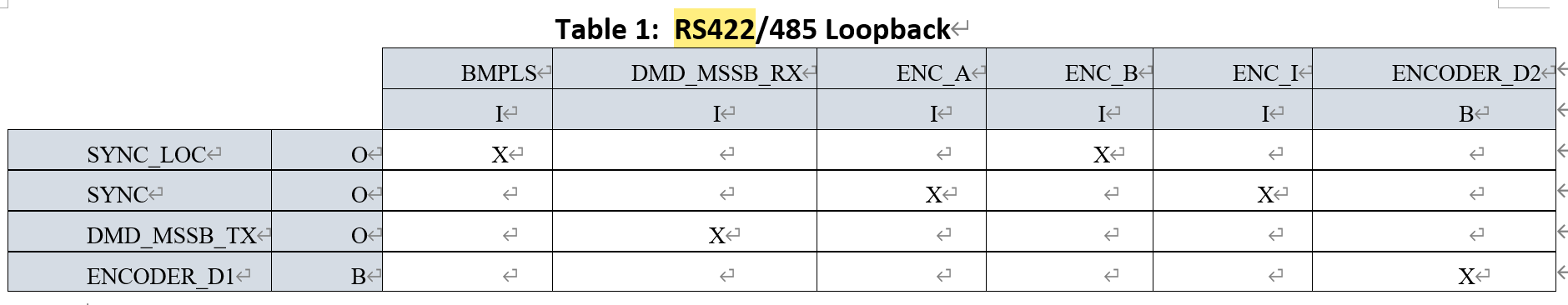
****

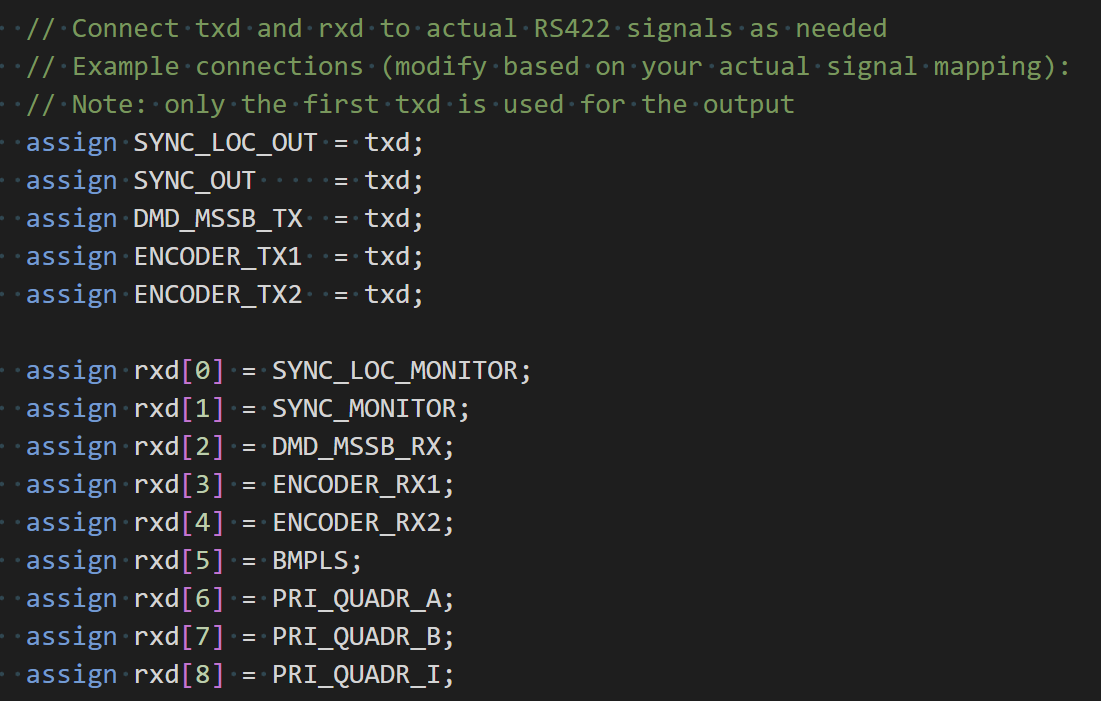
### **RS422模块**

* + 1. **测试方法**

**参考ATP 8.18 Network Interface: SP485，采用如下环回测试。**

首先，通过SYNC\_LOC\_OUT, SYNC\_OUT, DMD\_MSSB\_TX, ENCODER\_TX1, ENCODER\_TX2发送特定测试数据，测试字节和发送次数可配置。然后，读取接收状态寄存器和错误次数寄存器，确认测试是否通过，具体参照如下测试用例。



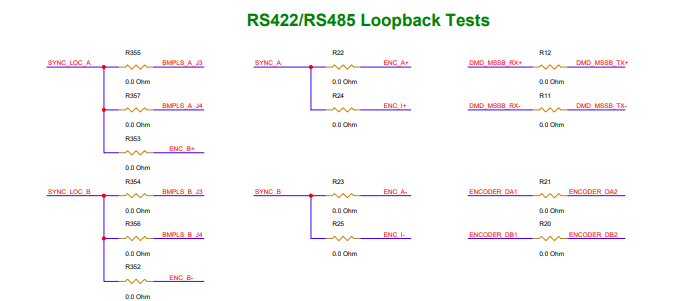
****

* + 1. **寄存器说明**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **地址（宏定义）** | **偏移地址** | **读写** | **位宽** | **说明1** | **说明2** |
| TEST\_PATTERN | 0x00 | R | 32位 | 测试字节 | NA，低8位有效 |
| TX\_TRANS\_BYTES | 0x01 | R | 32位 | 发送次数 | NA |
| TX\_CTRL | 0x02 | R/W | 32位 | bit-0: start test  bit-1: clear status  bit-2: stop test | 32’h0000\_0000 |
| RX\_STATUS | 0x03 | R | 32位 | Bit-0: 所有发送完成  Bit-1: rxd[0]收发次数一致  Bit-2: rxd[1]收发次数一致  Bit-3: rxd[2]收发次数一致  Bit-4: rxd[3]收发次数一致  Bit-5: rxd[4]收发次数一致  Bit-6: rxd[5]收发次数一致  Bit-7: rxd[6]收发次数一致  Bit-8: rxd[7]收发次数一致  Bit-9: rxd[8]收发次数一致 | 复位值：32’h0000\_0000  测试通过：  32’h0000\_03FF |
| RX1\_RECV\_BYTES | 0x04 | R |  | rxd[0]接收字节次数 | 测试通过等于TX\_TRANS\_BYTES |
| RX2\_RECV\_BYTES | 0x05 | R |  | rxd[1]接收字节次数 | 同上 |
| RX3\_RECV\_BYTES | 0x06 | R |  | rxd[2]接收字节次数 | 同上 |
| RX4\_RECV\_BYTES | 0x07 | R |  | rxd[3]接收字节次数 | 同上 |
| RX5\_RECV\_BYTES | 0x08 | R |  | rxd[4]接收字节次数 | 同上 |
| RX6\_RECV\_BYTES | 0x09 | R |  | rxd[5]接收字节次数 | 同上 |
| RX7\_RECV\_BYTES | 0x0A | R |  | rxd[6]接收字节次数 | 同上 |
| RX8\_RECV\_BYTES | 0x0B | R |  | rxd[7]接收字节次数 | 同上 |
| RX9\_RECV\_BYTES | 0x0C | R |  | rxd[8]接收字节次数 | 同上 |
| RX1\_ERR\_BYTES | 0x0D | R |  | rxd[0]接收错误字节数 | 测试通过为0 |
| RX2\_ERR\_BYTES | 0x0E | R |  | rxd[1]接收错误字节数 | 同上 |
| RX3\_ERR\_BYTES | 0x0F | R |  | rxd[2]接收错误字节数 | 同上 |
| RX4\_ERR\_BYTES | 0x10 | R |  | rxd[3]接收错误字节数 | 同上 |
| RX5\_ERR\_BYTES | 0x11 | R |  | rxd[4]接收错误字节数 | 同上 |
| RX6\_ERR\_BYTES | 0x12 | R |  | rxd[5]接收错误字节数 | 同上 |
| RX7\_ERR\_BYTES | 0x13 | R |  | rxd[6]接收错误字节数 | 同上 |
| RX8\_ERR\_BYTES | 0x14 | R |  | rxd[7]接收错误字节数 | 同上 |
| RX9\_ERR\_BYTES | 0x15 | R |  | rxd[8]接收错误字节数 | 同上 |

* + 1. **测试用例**

参考ATP8.18 Figure-1如下硬件连线：



**Step-1.** OPB Write test pattern register, pattern byte = 0x5A

>>5A 00 07 00 00 00 00 00 5A A5

**Step-2**. OPB Write Tx Trans bytes register, 0x2710 = 10000

>>5A 00 07 00 01 00 00 27 10 A5

**Step-3**. OPB Write Tx Ctrl register to trigger data transfer

>>5A 00 07 00 02 00 00 00 01 A5

**Step-4**. OPB Read Rx Status register to check transfer done

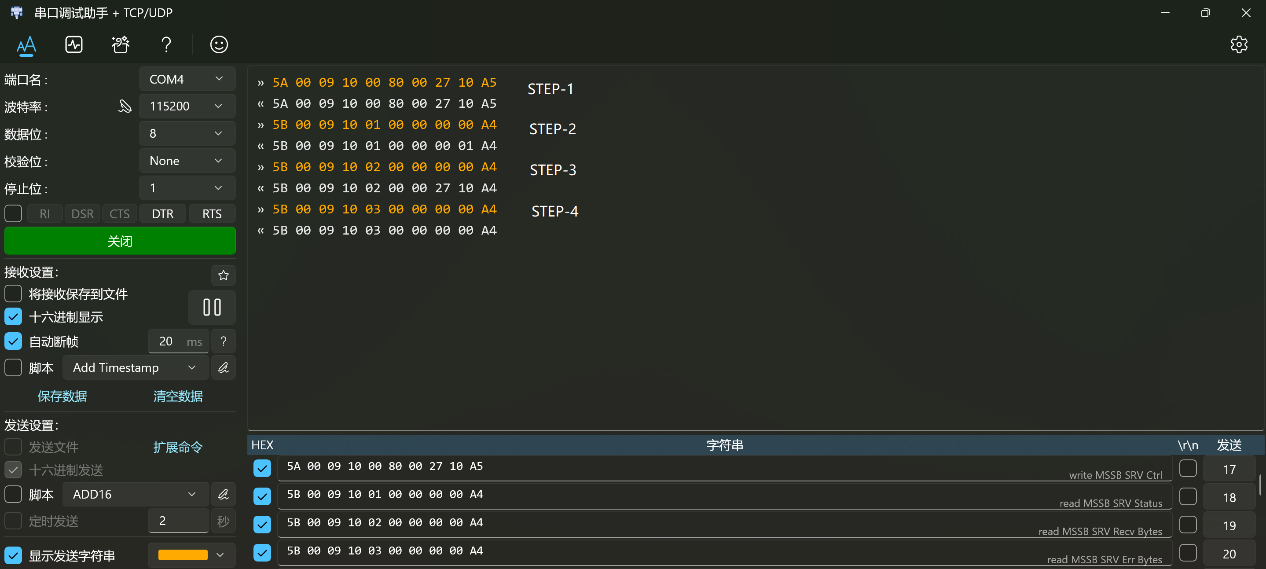
>>5B 00 07 00 03 00 00 00 00 A4

<<5B 00 07 00 03 00 00 03 FF A4

**Step-5**. OPB Read Rx error bytes register to check receive done without error

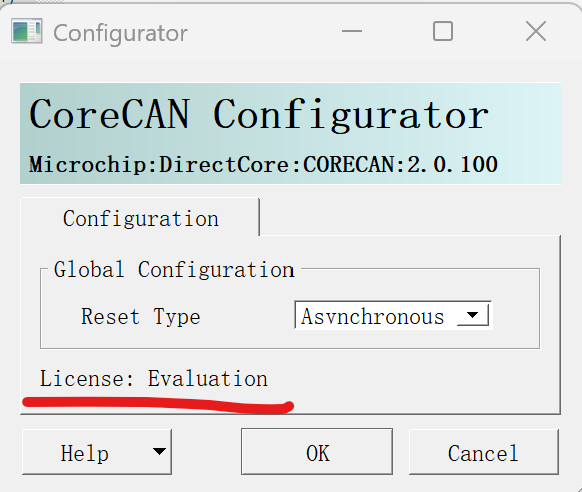
>>5B 00 07 00 0D 00 00 00 00 A4

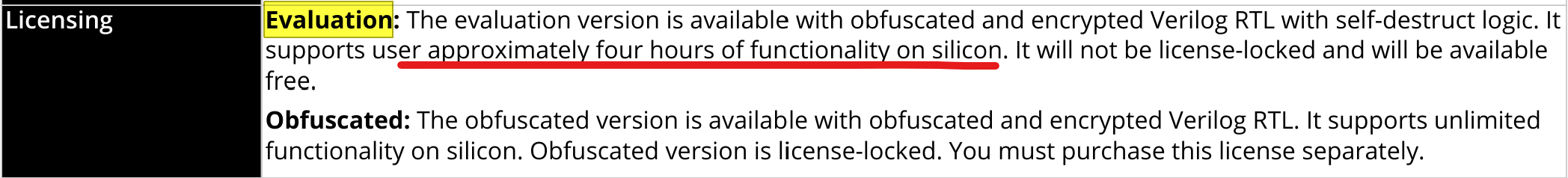
<<5B 00 07 00 0D 00 00 00 00 A4



### **CAN模块**

**注意：CAN测试部分使用了Microchip CoreCAN IP，但由于开发软件LIBERO采用的是Evaluation License，本模块上电后只能稳定工作4小时(40M pclk)。实际情况中，工作时长大约在2.5小时(100M pclk)。**

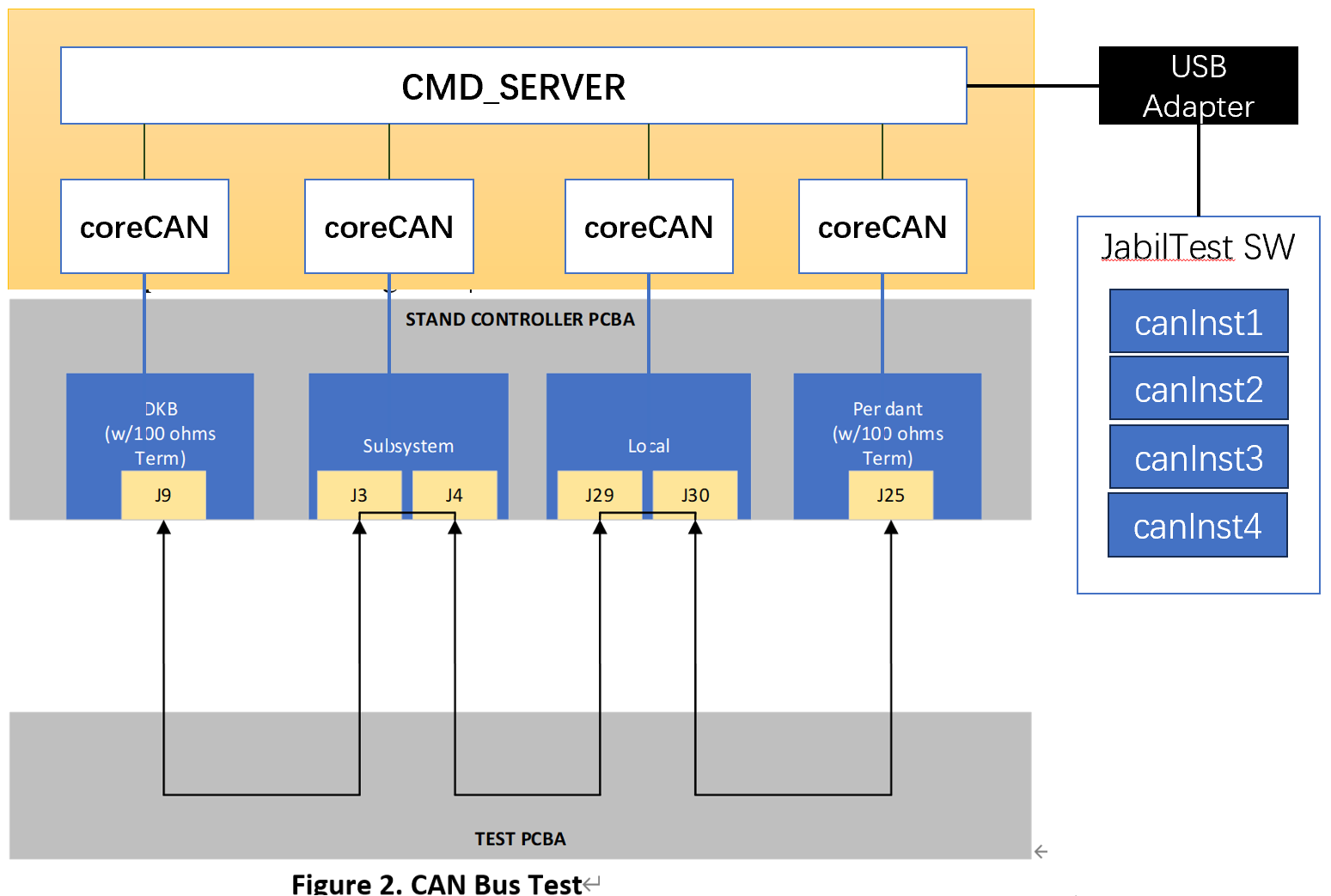




* + 1. **测试方法**

参考ATP 章节8.20 Network Interface: CAN Bus要求的测试方法，物理连线如下所示：1）J9和J3的CAN\_H和CAN\_L短接；2）J4和J29的CAN\_H和CAN\_L短接；3）J30和J25的CAN\_H和CAN\_L短接。

在APP FPGA固件设计中，为每个CAN PHY transceiver配置一个CAN Controller, 不妨设定为CAN1, CAN2, CAN3, CAN4。按照上述连线方式，这4个控制器挂载在同一CAN网络。在测试过程中，CAN1发送特定TX MSG ID到CAN2 RX MSG FIFO；类似地，CAN2发送到CAN3，CAN3发送到CAN4，CAN4发送到CAN1。然后依次检查每个CAN RX MSG FIFO的内容是否和对应的TX MSG一致。如果发送和接收消息均一致，那么CAN测试通过。



CAN Controller的参考配置如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Controller** | **TX\_ID** | **TX\_MSG** | **RX\_AMR** | **RX\_ACR** |
| CAN1 | 0x7E1 | 10 20 30 40 50 60 70 80 | 0x01E | 0x7E4 |
| CAN2 | 0x7E2 | 11 22 33 44 55 66 77 88 | 0x01D | 0x7E1 |
| CAN3 | 0x7E3 | A1 B2 C3 D4 E5 F6 17 28 | 0x01C | 0x7E2 |
| CAN4 | 0x7E4 | 12 34 56 78 9A BC DE F0 | 0x01B | 0x7E3 |

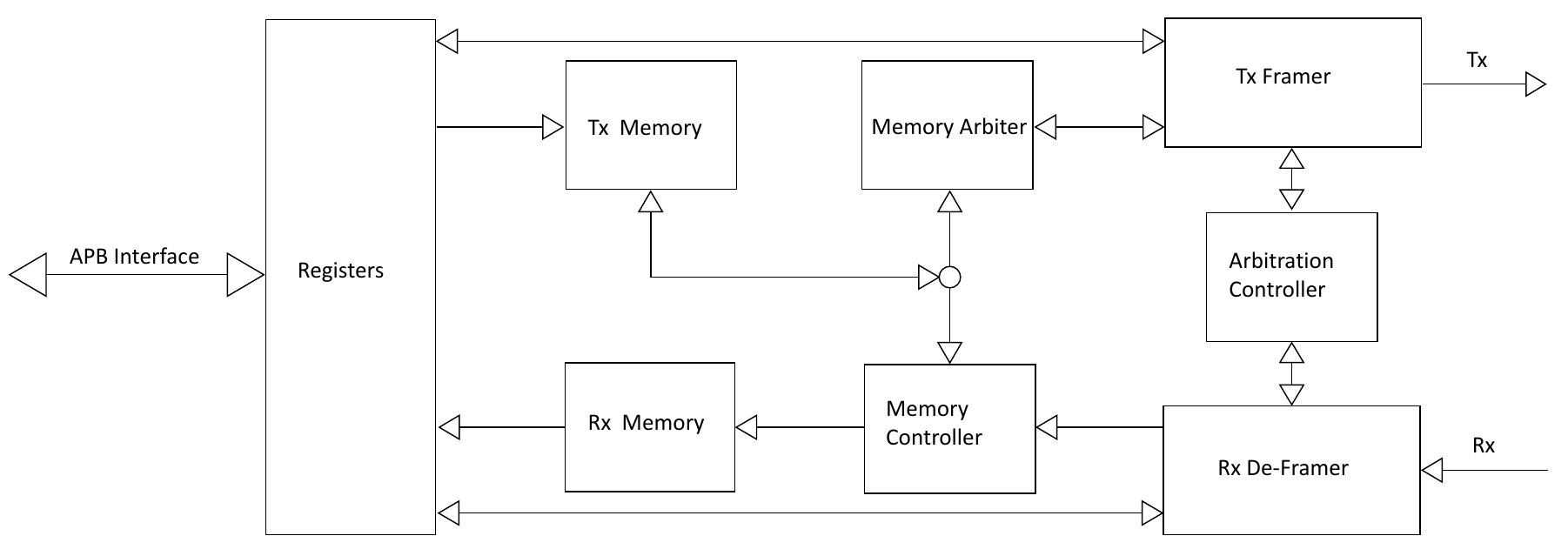
注意：AMR和ACR的滤波器配置保证控制器接收到特定TX\_ID的消息。

* + 1. **CORECAN IP**
       1. **CORECAN介绍**

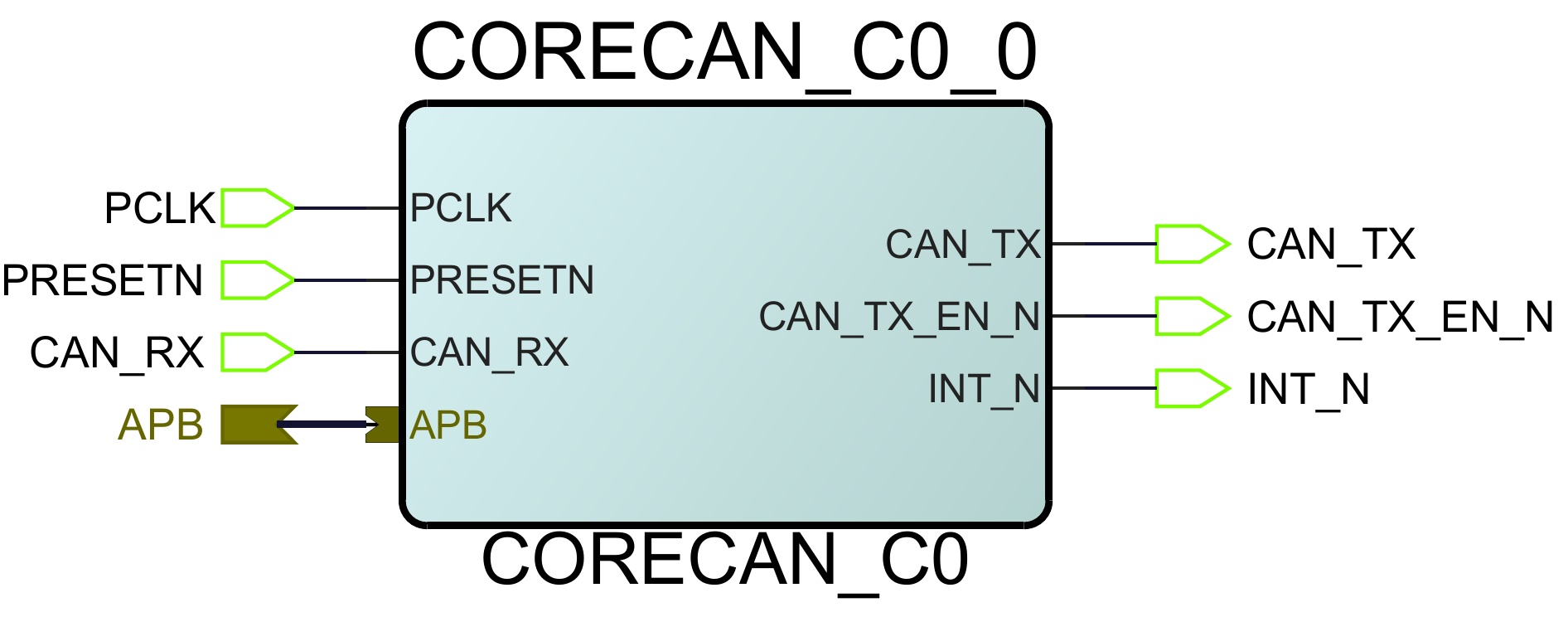
CORECAN是Microchip提供的符合CAN规范的控制器IP，它能够正确收发消息、处理各种错误和管理总线冲突等。其用户文档可以从官网下载：

<https://www.microchip.com/en-us/products/fpgas-and-plds/ip-core-tools/corecan_fd>

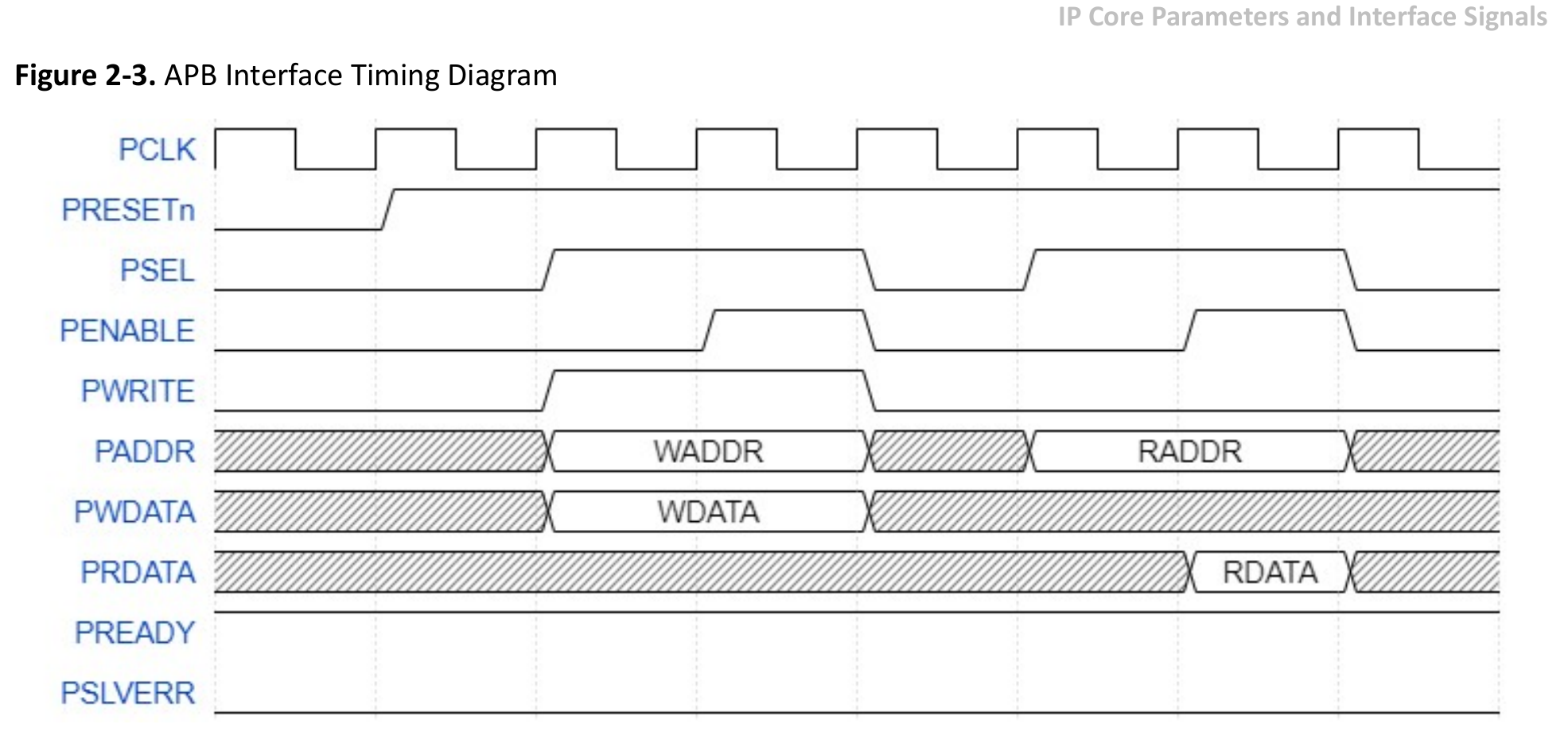
以下是它的组件示意图：



将CORECAN IP集成到FPGA设计中，我们更关心的是其顶层接口：



APB总线用于CORECAN内部寄存器的读写，其时序如下：



* + - 1. **CORECAN寄存器配置**

**CAN1 Controller: OPB Base Address = 32’h0008\_0000**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Offset | Register | W/R | Description | OPB\_ADDR | OPB\_DATA |
| 0x04 | INT ENA | W | Enable all Interrupts | 32’h00080004 | 32’hFFFFFFFF |
| 0x14 | CMD | W | B3: LPBK MODE  B2: LISTEN MODE  B1: RUN\_STOP MODE | 32’h00080014 | 32’h00000001 |
| 0x18 | CONFIG | W | CLK100M, Baud rate = 1Mbps | 32’h00080018 | 32’h00090460 |
| 0x24 | TX\_MSG0\_ID | W | 11’H7F1 | 32’h00080024 | 32’hFE200000 |
| 0x28 | TX\_MSG0\_WORD1 | W | TX\_MSG0\_WORD1 | 32’h00080028 | 32’h10203040 |
| 0x2C | TX\_MSG0\_WORD0 | W | TX\_MSG0\_WORD0 | 32’h0008002C | 32’h50607080 |
| 0x230 | RX\_MSG0\_AMR | W | AMR for TX4\_MSG0\_ID | 32’h00080230 | 32’h017FFFFF |
| 0x234 | RX\_MSG0\_ACR | W | ACR for TX4\_MSG0\_ID | 32’h00080234 | 32’hFE800000 |
| 0x238 | RX\_MSG0\_AMR\_DATA | W | 32’hFFFFFFFF | 32’h00080238 | 32’hFFFFFFFF |
| 0x23C | RX\_MSG0\_ACR\_DATA | W | 32’ h00000000 | 32’h0008023C | 32’ h00000000 |
| 0x220 | RX\_MSG0\_CTRL | W | 32’hFFFFFFFF | 32’h00080220 | 32’hFFFFFFFF |
| 0x20 | TX\_MSG0\_CTRL | W | Init TX Message Transmit | 32’h00080020 | 32’ h00880001 |
| 0x08 | RX\_BUF STATUS | R | RX MSG is received | 32’h00088008 | 32’ h00000001 |
| 0x10 | ERR\_STATUS | R | CAN ERR status | 32’h00088010 | 32’ h00000000 |
| 0x224 | RX\_MSG0 ID | R | CAN4 TX\_MSG0 ID | 32’h00088224 | 32’hFE9FFFF8 |
| 0x228 | RX\_MSG0 WORD1 | R | CAN4 TX\_MSG0 WORD1 | 32’h00088228 | 32’h12345678 |
| 0x22C | RX\_MSG0 WORD0 | R | CAN4 TX\_MSG0 WORD0 | 32’h0008822C | 32’h9ABCDEF0 |

**CAN2 Controller: OPB Base Address = 32’h0009\_0000**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Offset | Register | W/R | Description | OPB\_ADDR | OPB\_DATA |
| 0x04 | INT ENA | W | Enable all Interrupts | 32’h00090004 | 32’hFFFFFFFF |
| 0x14 | CMD | W | B3: LPBK MODE  B2: LISTEN MODE  B1: RUN\_STOP MODE | 32’h00090014 | 32’h00000001 |
| 0x18 | CONFIG | W | CLK100M, Baud rate = 1Mbps | 32’h00090018 | 32’h00090460 |
| 0x24 | TX\_MSG0\_ID | W | 11’H7F2 | 32’h00090024 | 32’hFE400000 |
| 0x28 | TX\_MSG0\_WORD1 | W | TX\_MSG0\_WORD1 | 32’h00090028 | 32’h11223344 |
| 0x2C | TX\_MSG0\_WORD0 | W | TX\_MSG0\_WORD0 | 32’h0009002C | 32’h55667788 |
| 0x230 | RX\_MSG0\_AMR | W | AMR for TX1\_MSG0\_ID | 32’h00090230 | 32’h01DFFFFF |
| 0x234 | RX\_MSG0\_ACR | W | ACR for TX1\_MSG0\_ID | 32’h00090234 | 32’hFE200000 |
| 0x238 | RX\_MSG0\_AMR\_DATA | W | 32’hFFFFFFFF | 32’h00090238 | 32’hFFFFFFFF |
| 0x23C | RX\_MSG0\_ACR\_DATA | W | 32’ h00000000 | 32’h0009023C | 32’ h00000000 |
| 0x220 | RX\_MSG0\_CTRL | W | 32’hFFFFFFFF | 32’h00090220 | 32’hFFFFFFFF |
| 0x20 | TX\_MSG0\_CTRL | W | Init TX Message Transmit | 32’h00090020 | 32’ h00880001 |
| 0x08 | RX\_BUF STATUS | R | RX MSG is received | 32’h00098008 | 32’ h00000001 |
| 0x10 | ERR\_STATUS | R | CAN ERR status | 32’h00098010 | 32’ h00000000 |
| 0x224 | RX\_MSG0 ID | R | CAN1 TX\_MSG0 ID | 32’h00098224 | 32’hFE3FFFF8 |
| 0x228 | RX\_MSG0 WORD1 | R | CAN1 TX\_MSG0 WORD1 | 32’h00098228 | 32’h10203040 |
| 0x22C | RX\_MSG0 WORD0 | R | CAN1 TX\_MSG0 WORD0 | 32’h0009822C | 32’h50607080 |

**CAN3 Controller: OPB Base Address = 32’h000A\_0000**

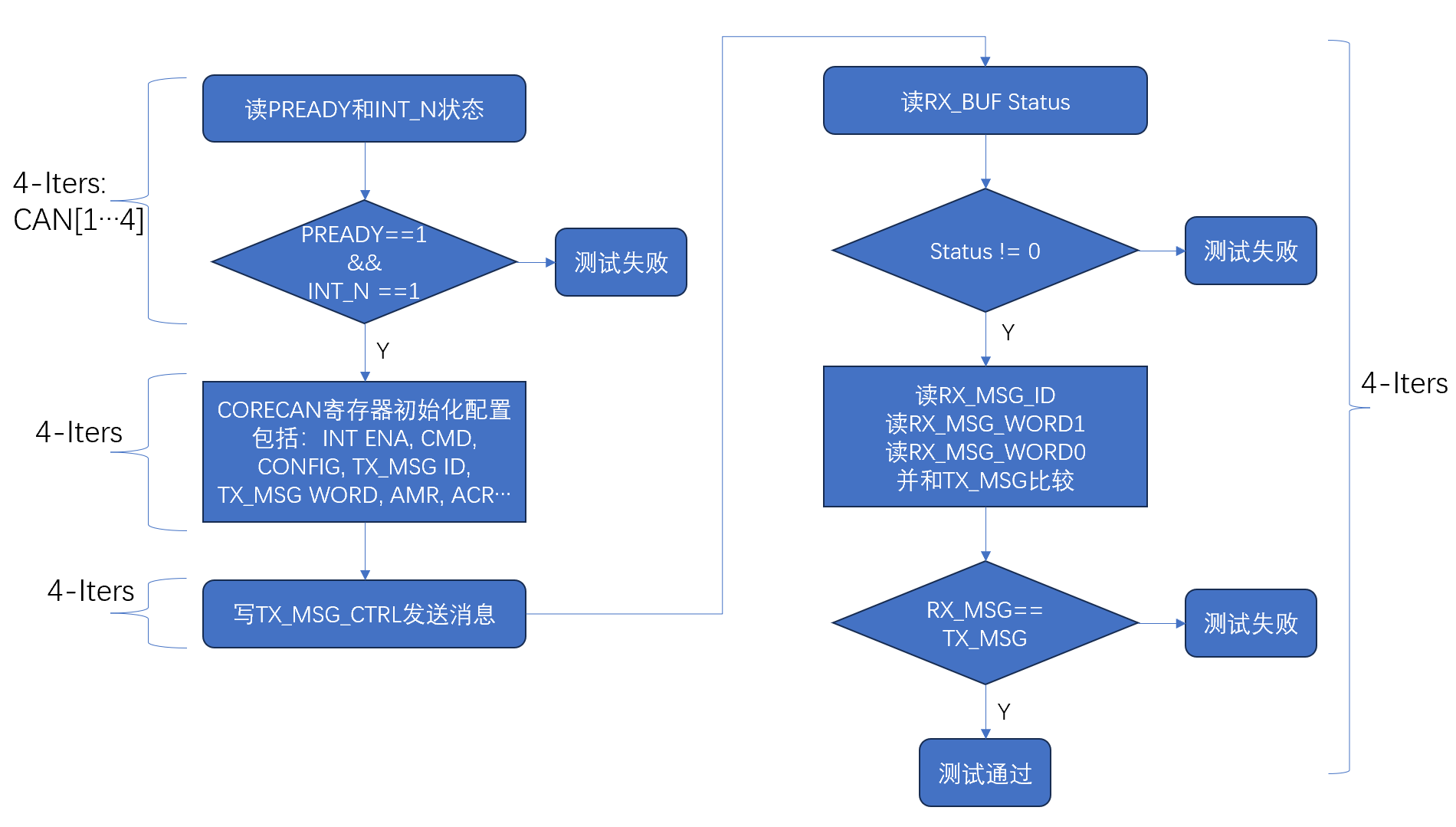
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Offset | Register | W/R | Description | OPB\_ADDR | OPB\_DATA |
| 0x04 | INT ENA | W | Enable all Interrupts | 32’h000B0004 | 32’hFFFFFFFF |
| 0x14 | CMD | W | B3: LPBK MODE  B2: LISTEN MODE  B1: RUN\_STOP MODE | 32’h000B0014 | 32’h00000001 |
| 0x18 | CONFIG | W | CLK100M, Baud rate = 1Mbps | 32’h000B0018 | 32’h00090460 |
| 0x24 | TX\_MSG0\_ID | W | 11’H7F3 | 32’h000B0024 | 32’hFE600000 |
| 0x28 | TX\_MSG0\_WORD1 | W | TX\_MSG0\_WORD1 | 32’h000B0028 | 32’hA1B2C3D4 |
| 0x2C | TX\_MSG0\_WORD0 | W | TX\_MSG0\_WORD0 | 32’h000B002C | 32’hE5F61728 |
| 0x230 | RX\_MSG0\_AMR | W | AMR for TX2\_MSG0\_ID | 32’h000B0230 | 32’h01BFFFFF |
| 0x234 | RX\_MSG0\_ACR | W | ACR for TX2\_MSG0\_ID | 32’h000B0234 | 32’hFE400000 |
| 0x238 | RX\_MSG0\_AMR\_DATA | W | 32’hFFFFFFFF | 32’h000B0238 | 32’hFFFFFFFF |
| 0x23C | RX\_MSG0\_ACR\_DATA | W | 32’ h00000000 | 32’h000B023C | 32’ h00000000 |
| 0x220 | RX\_MSG0\_CTRL | W | 32’hFFFFFFFF | 32’h000B0220 | 32’hFFFFFFFF |
| 0x20 | TX\_MSG0\_CTRL | W | Init TX Message Transmit | 32’h000B0020 | 32’ h00880001 |
| 0x08 | RX\_BUF STATUS | R | RX MSG is received | 32’h000B8008 | 32’ h00000001 |
| 0x10 | ERR\_STATUS | R | CAN ERR status | 32’h000B8010 | 32’ h00000000 |
| 0x224 | RX\_MSG0 ID | R | CAN2 TX\_MSG0 ID | 32’h000B8224 | 32’hFE5FFFF8 |
| 0x228 | RX\_MSG0 WORD1 | R | CAN2 TX\_MSG0 WORD1 | 32’h000B8228 | 32’ h11223344 |
| 0x22C | RX\_MSG0 WORD0 | R | CAN2 TX\_MSG0 WORD0 | 32’h000B822C | 32’ h55667788 |

**CAN4 Controller: OPB Base Address = 32’h000B\_0000**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Offset | Register | W/R | Description | OPB\_ADDR | OPB\_DATA |
| 0x04 | INT ENA | W | Enable all Interrupts | 32’h000A0004 | 32’hFFFFFFFF |
| 0x14 | CMD | W | B3: LPBK MODE  B2: LISTEN MODE  B1: RUN\_STOP MODE | 32’h000A0014 | 32’h00000001 |
| 0x18 | CONFIG | W | CLK100M, Baud rate = 1Mbps | 32’h000A0018 | 32’h00090460 |
| 0x24 | TX\_MSG0\_ID | W | 11’H7F4 | 32’h000A0024 | 32’hFE800000 |
| 0x28 | TX\_MSG0\_WORD1 | W | TX\_MSG0\_WORD1 | 32’h000A0028 | 32’h12345678 |
| 0x2C | TX\_MSG0\_WORD0 | W | TX\_MSG0\_WORD0 | 32’h000A002C | 32’h9ABCDEF0 |
| 0x230 | RX\_MSG0\_AMR | W | AMR for TX3\_MSG0\_ID | 32’h000A0230 | 32’h019FFFFF |
| 0x234 | RX\_MSG0\_ACR | W | ACR for TX3\_MSG0\_ID | 32’h000A0234 | 32’hFE600000 |
| 0x238 | RX\_MSG0\_AMR\_DATA | W | 32’hFFFFFFFF | 32’h000A0238 | 32’hFFFFFFFF |
| 0x23C | RX\_MSG0\_ACR\_DATA | W | 32’ h00000000 | 32’h000A023C | 32’ h00000000 |
| 0x220 | RX\_MSG0\_CTRL | W | 32’hFFFFFFFF | 32’h000A0220 | 32’hFFFFFFFF |
| 0x20 | TX\_MSG0\_CTRL | W | Init TX Message Transmit | 32’h000A0020 | 32’ h00880001 |
| 0x08 | RX\_BUF STATUS | R | RX MSG is received | 32’h000A8008 | 32’ h00000001 |
| 0x10 | ERR\_STATUS | R | CAN ERR status | 32’h000A8010 | 32’ h00000000 |
| 0x224 | RX\_MSG0 ID | R | CAN3 TX\_MSG0 ID | 32’h000A8224 | 32’hFE7FFFF8 |
| 0x228 | RX\_MSG0 WORD1 | R | CAN3 TX\_MSG0 WORD1 | 32’h000A8228 | 32’hA1B2C3D4 |
| 0x22C | RX\_MSG0 WORD0 | R | CAN3 TX\_MSG0 WORD0 | 32’h000A822C | 32’ hE5F61728 |

* + 1. **参考测试流程**

程序流程图如下：

****

JabilTest测试脚本见附件。

* + 1. **测试结果**

测试结果见日志文件。