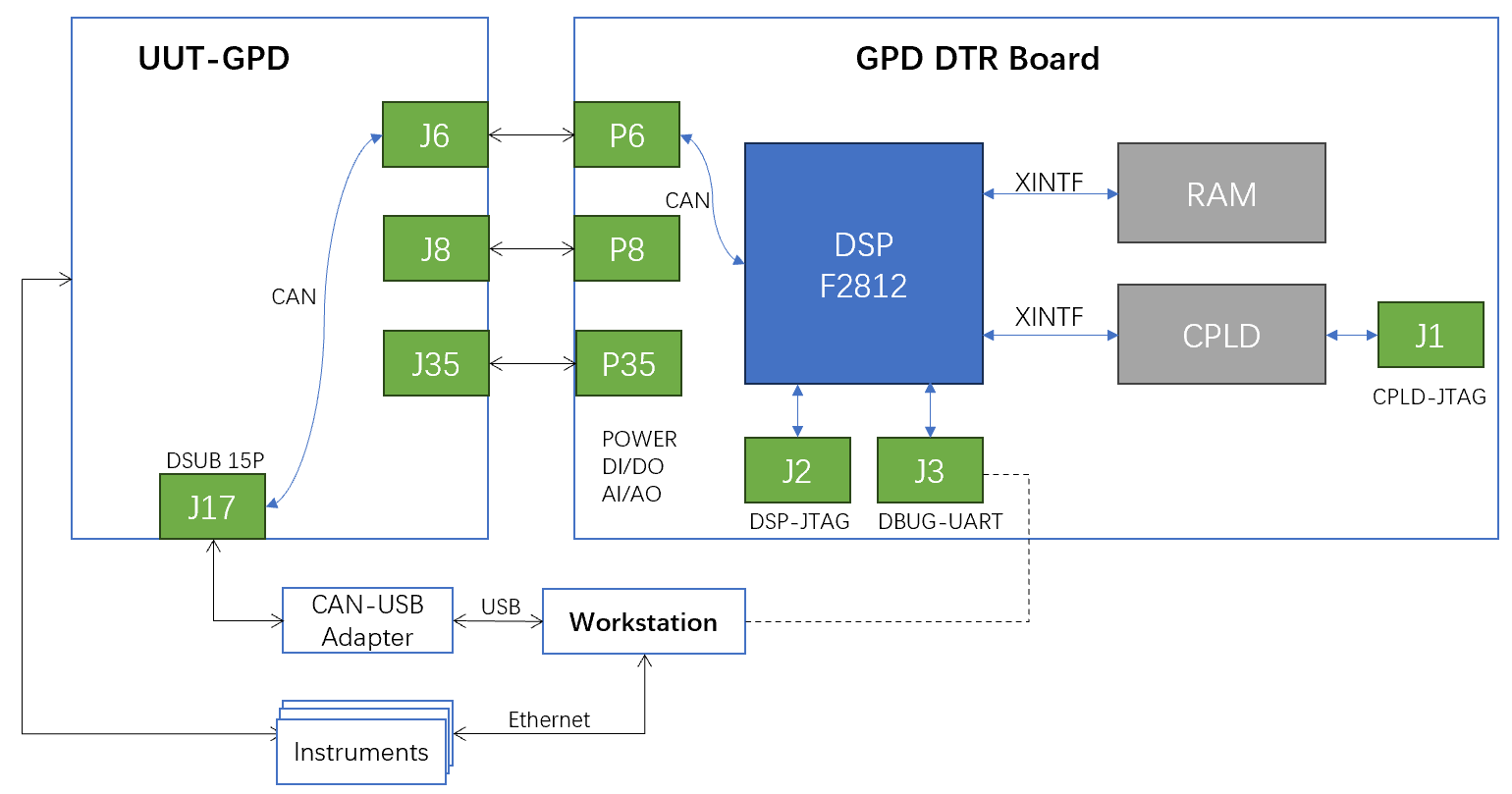
# **GPD测试固件开发指导**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **作者** | **日期** | **描述** |
| 0.1 | XYL | 2025/4/5 | 基于原ATP的初始版本 |
| 0.2 | XYL | 2025/4/7 | 修改CAN Message响应机制 |

1. **概要**

本文档用于指导GPD测试固件开发，即DSP F2812程序设计。

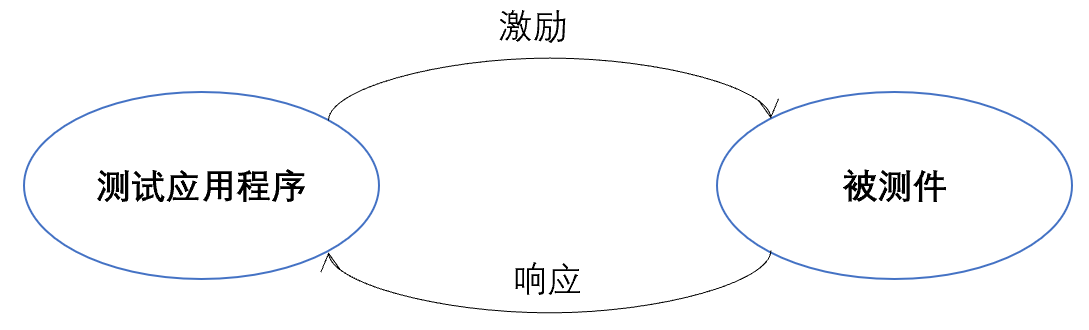
GPD测试系统主要组成包括：被测件GPD PCBA，GPD DTR Board，CAN-USB Adapter、Instruments、Workstation等等。如下图所示：



测试系统各子部件的大致功能描述如下：

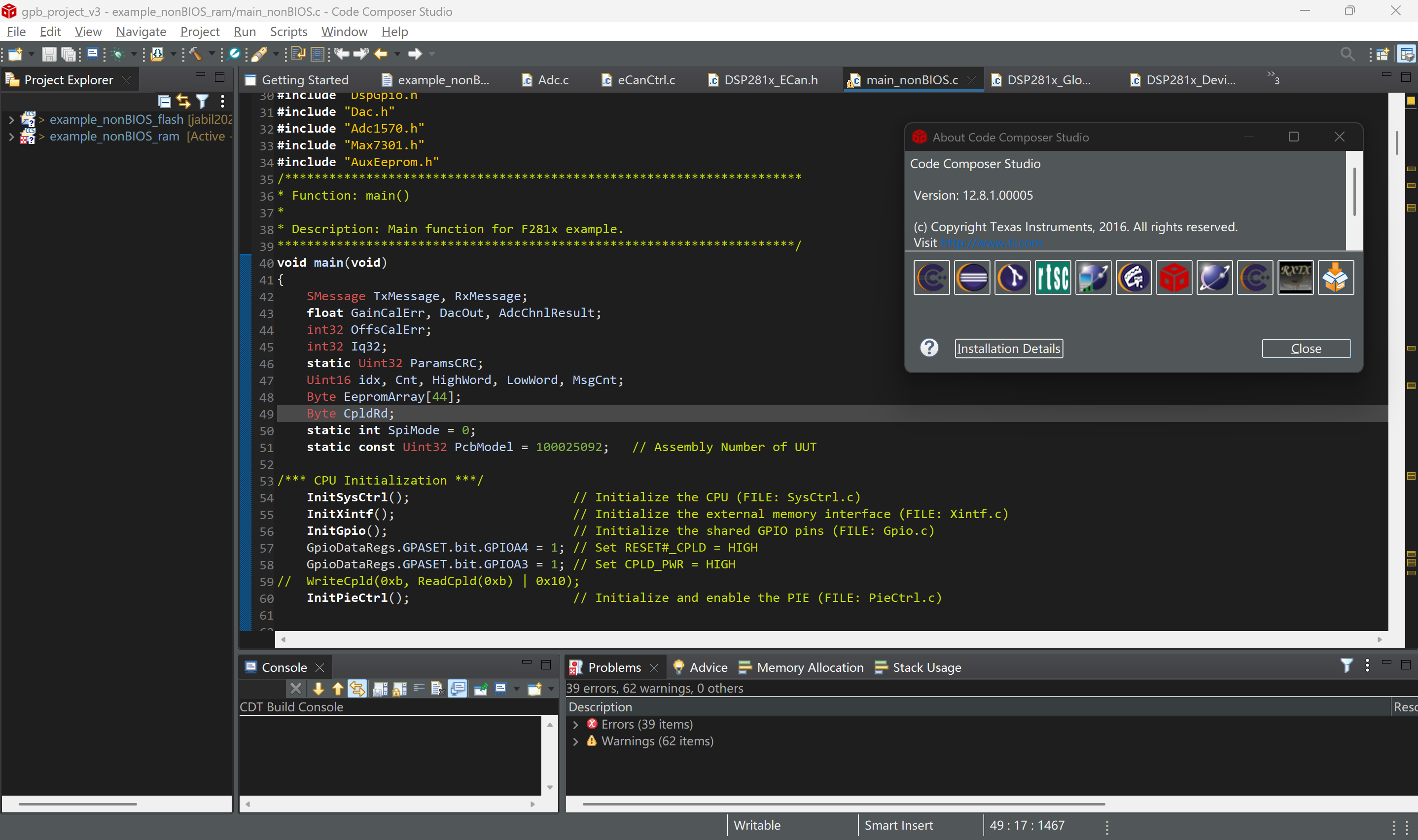
|  |  |  |
| --- | --- | --- |
| **编号** | **部件名字** | **功能描述** |
| 1 | GPD PCBA | 被测件，测试项包括目检、上电、器件功能等，具体见文档GPD ATP |
| 2 | GPD DTR Board | GPD子板，运行测试固件，协助测试应用程序完成各项测试任务 |
| 3 | CAN-USB Adapter | CAN到USB转换器 |
| 4 | Instruments | 测试测量仪表，包括电源、信号源、示波器、频谱仪、负载等等 |
| 5 | Workstation | 服务器电脑，运行测试应用程序，控制仪表和GPD子板，完成ATP定义的各项测试任务 |

每一项测试(目检除外)均由测试应用程序主动发起，控制测试仪表或者GPD子板输出激励到被测件，然后读取被测件的响应。如果响应符合预期(ATP定义的测试通过标准)，则表明这一项测试PASS，否则FAIL。依次遍历所有的测试项，完成测试任务并打印日志。



由此可见，GPD测试固件的主要任务是响应上层测试应用程序的任务请求，包括读写DIO（数字IO）、写DAC（模拟IO）、读ADC（模拟IO），片上存储芯片如EEPROM的读写测试等等。

在GPD测试系统中，上层测试应用程序运行在服务器电脑，通过CAN Bus和GPD子板的DSP控制器进行通信。上层应用程序基于JabilTest或者.NET开发，DSP F2812固件基于CCS12.8开发。CCS IDE如下图：

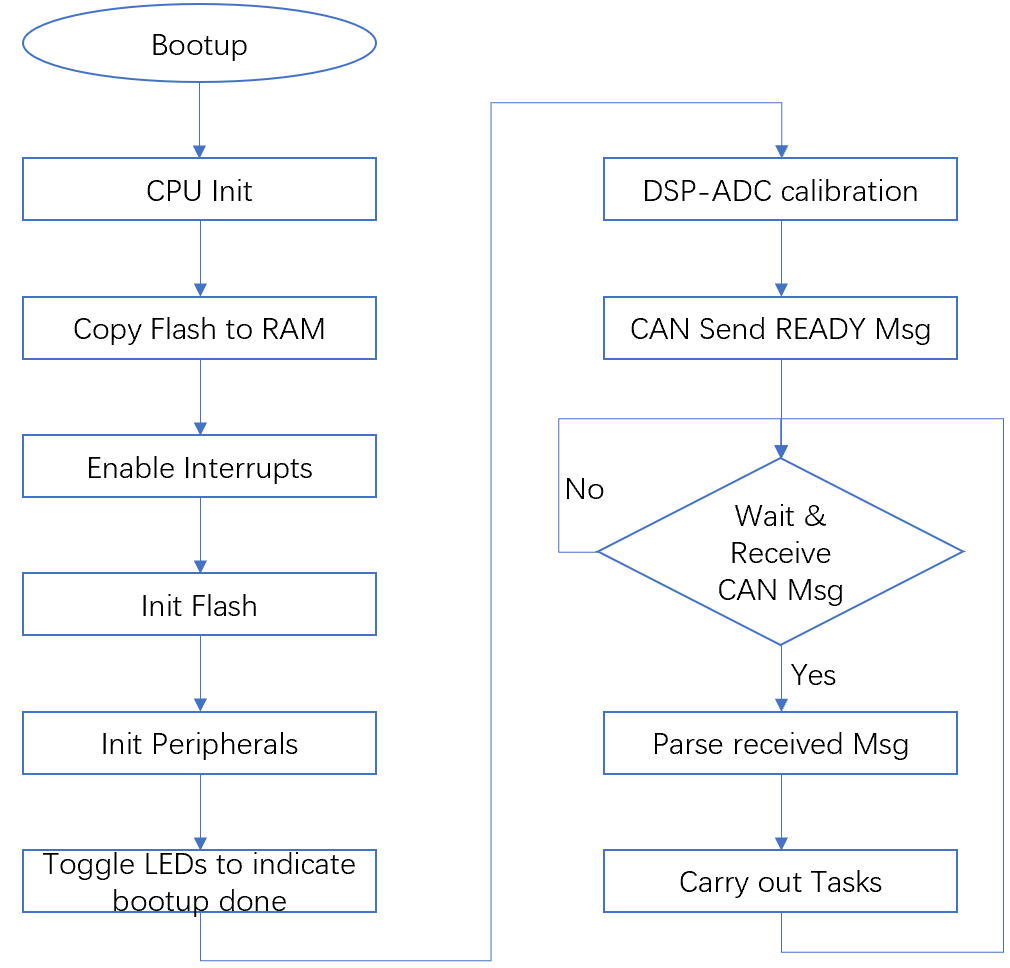


1. **主程序流程图**

DSP 固件采用noBIOS\_Flash工程形式，即无RTOS操作系统+Flash保存固件镜像。

GPD子板上电bootup后转入main函数，执行一系列的初始化操作，然后不停的检查CAN Rx Mail Boxes，解析上层应用程序的命令并执行对应任务，包括DIO/AIO操作，读写RAM/EEPROM，或者将检测结果通过CAN Tx返回给应用程序。

主程序流程图如下：

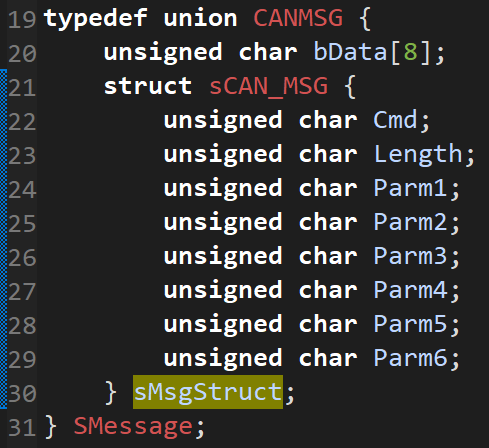


1. **子程序介绍**

**TBD**

1. **通信协议定义**

测试系统采用CAN 2.0B协议，每条Message包含8 bytes数据，第1字节是Cmd，第2字节是Length，第3-8字节命令参数或者测试结果。CANMSG定义如下：



**基于CANMSG的通信协议定义如下表：**

注意：

1. 方向“收”表示消息从上位机应用程序到DSP，方向“发”表示消息从DSP到上位机应用程序
2. (2) DSP需要回复每条上位机消息；如果没有指定消息内容，返回Cmd即可。
3. (3) Parm均为十六进制表示
4. (4) X 或者空表示该字节数据不予考虑
5. (5) static const Uint32 PcbModel = 100025092

| **Cmd** | **Len** | **Parm1** | **Parm2** | **Parm3** | **Parm4** | **Parm5** | **Parm6** | **方向** | **描述** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ‘P’ | 6 | AA | 55 | CC | 33 | DD | 22 | 发 | DSP准备就绪，此时应用程序可以下发命令 |
| ‘A’ | 1 | 0 |  |  |  |  |  | 收 | GetAdcConversion ADCIN0: 1.8V |
| ‘A’ | 6 | 0 | 00 | Iq32[3] | Iq32[2] | Iq32[1] | Iq32[0] | 发 | Return ADC Result (IQ32 format) |
| ‘A’ | 2 | 0 | FF |  |  |  |  | 发 | Number out of IQ32 range |
| ‘A’ | 1 | 1 |  |  |  |  |  | 收 | GetAdcConversion ADCIN1: MTR3；参考ADCIN0，Parm1 = 1 |
| ‘A’ | 1 | 2 |  |  |  |  |  | 收 | GetAdcConversion ADCIN2: MTR5；参考ADCIN0，Parm1 = 2 |
| ‘A’ | 1 | 3 |  |  |  |  |  | 收 | GetAdcConversion ADCIN3: MTR24；参考ADCIN0，Parm1 = 3 |
| ‘A’ | 1 | 4 |  |  |  |  |  | 收 | GetAdcConversion ADCIN4: REF0.9；参考ADCIN0，Parm1 = 4 |
| ‘A’ | 1 | 5 |  |  |  |  |  | 收 | GetAdcConversion ADCIN5: REF2.5；参考ADCIN0，Parm1 = 5 |
| ‘A’ | 1 | 6 |  |  |  |  |  | 收 | GetAdcConversion ADCIN6: 0.1908V；参考ADCIN0，Parm1 = 6 |
| ‘A’ | 1 | 7 |  |  |  |  |  | 收 | GetAdcConversion ADCIN7: AO\_MF\_TST0；参考ADCIN0，Parm1 = 7 |
| ‘A’ | 1 | 8 |  |  |  |  |  | 收 | GetAdcConversion ADCIN8: AO\_MF\_TST1；参考ADCIN0，Parm1 = 8 |
| ‘A’ | 1 | 9 |  |  |  |  |  | 收 | GetAdcConversion ADCIN9: AO\_MF\_TST2；参考ADCIN0，Parm1 = 9 |
| ‘A’ | 1 | A |  |  |  |  |  | 收 | GetAdcConversion ADCIN10: AO\_MF\_TST3；参考ADCIN0，Parm1 = 10 |
| ‘A’ | 1 | B |  |  |  |  |  | 收 | GetAdcConversion ADCIN11: AGND；参考ADCIN0，Parm1 = 11 |
| ‘A’ | 1 | C |  |  |  |  |  | 收 | GetAdcConversion ADCIN12: PEL\_MON；参考ADCIN0，Parm1 = 12 |
| ‘A’ | 1 | D |  |  |  |  |  | 收 | GetAdcConversion ADCIN13: MEL\_MON；参考ADCIN0，Parm1 = 13 |
| ‘A’ | 1 | E |  |  |  |  |  | 收 | GetAdcConversion ADCIN14: BEL\_MON；参考ADCIN0，Parm1 = 14 |
| ‘A’ | 1 | F |  |  |  |  |  | 收 | GetAdcConversion ADCIN15: TEMP\_MON；参考ADCIN0，Parm1 = 15 |
| ‘B’ | 6 | 0 | X | pcbHwVersion[3] | pcbHwVersion[2] | pcbHwVersion[1] | pcbHwVersion[0] | 收 | 写pcbHwVersion到EepromArray |
| ‘B’ | 6 | 1 | X | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] | 收 | 写pcbSerialNum到EepromArray |
| ‘B’ | 1 | 2 |  |  |  |  |  | 收 | 写offsCalErr和GainCalErr到EpromArray |
| ‘B’ | 1 | 3 |  |  |  |  |  | 收 | DSP计算EEPROM内容CRC |
| ‘B’ | 6 | 3 | 0 | CRC[3] | CRC[2] | CRC[1] | CRC[0] | 发 | 返回EEPROM内容CRC |
| ‘B’ | 1 | 4 |  |  |  |  |  | 收 | DSP写EEPROM并验证 |
| ‘B’ | 4 | 4 | errBytes | errBytes | errBytes |  |  | 发 | DSP写EEPROM错误字节数。  如果写成功，Parm2=Parm3=Parm4=0；否则，写失败 |
| ‘B’ | 3 | 6 | start | end |  |  |  | 收 | 读EEPROM  NOTE: 上位机命令参数必须满足 1) 0<=Parm2<Parm3<=max(EEPROM ADDR) 2) Cnt = Parm3-Parm2<=6  3) 如果Cnt>6，DSP不回复 |
| ‘B’ | Cnt | EEPROM[start] | EEPROM[start+1] | EEPROM[start+2] | EEPROM[start+3] | EEPROM[start+4] | EEPROM[start+5] | 发 | 返回读EEPROM字节，最多6个字节 |
| ‘B’ | 1 | 7 |  |  |  |  |  | 收 | 读CAN ID Validation Password和CAN ID |
| ‘B’ | 6 | 0 | 0 | EEPROM[0x1ff4] | EEPROM[0x1ff5] | EEPROM[0x1ff6] | EEPROM[0x1ff7] | 发 | 返回CAN ID Validation Password和CAN ID  EEPROM[0x1ff4] = 0xA5  EEPROM[0x1ff5] = 0x77  EEPROM[0x1ff6] = 0x88  EEPROM[0x1ff7] = 0x94 |
| ‘C’ | 1 | 0 |  |  |  |  |  | 收 | 读CPLD Version Code，需要立刻返回 |
| ‘C’ | 2 | CPLD\_VER1 | CPLD\_VER0 |  |  |  |  | 发 | 返回CPLD Version Code |
| ‘D’ | 6 | DacChnl | 0 | HighWord[1] | HighWord[0] | LowWord[1] | LowWord[0] | 收 | 写DAC(通道A/B/C/D)，注意数据格式为IQ32  HighWord = ((RxMessage.sMsgStruct.Parm3 << 8) | (RxMessage.sMsgStruct.Parm4)) & 0xffff;  LowWord = ((RxMessage.sMsgStruct.Parm5 << 8) | (RxMessage.sMsgStruct.Parm6)) & 0xffff;  DacOut = ((**float**)(HighWord & 0x7fff)) + ((**float**)(LowWord))/65536.0; |
| ‘E’ | 1 | 0 |  |  |  |  |  | 收 | Test EEPROM |
| ‘E’ | 1 | errCount |  |  |  |  |  | 发 | 返回Test EEPROM结果(>3分钟) errCount=0，测试通过；否则测试失败 |
| ‘F’ | 1 | 0 |  |  |  |  |  | 收 | Test Ext EEPROM |
| ‘F’ | 1 | errCount |  |  |  |  |  | 发 | 返回Test Ext EEPROM结果(>3分钟) errCount=0，测试通过；否则测试失败 |
| ‘F’ | 1 | 1 |  |  |  |  |  | 收 | Read Ext EEPROM status register |
| ‘F’ | 1 | Status Register |  |  |  |  |  | 发 | 返回Ext EEPROM status register |
| ‘F’ | 3 | 2 | HiAddr | LoAddr |  |  |  | 收 | Read Byte from [HiAddr LoAddr] |
| ‘F’ | 1 | EEPROM[Addr] |  |  |  |  |  | 发 | 返回读取的字节，Addr = HiAddr<<8+LoAddr |
| ‘F’ | 2 | 3 | StsData |  |  |  |  | 收 | Write StsData to Status Register |
| ‘F’ | 4 | 4 | HiAddr | LoAddr | Data |  |  | 收 | 写Ext EEPROM，Eeprom[Addr]=Data  Addr = HiAddr<<8+LoAddr |
| ‘F’ | 6 | 5 | 0 | pcbModel[0] | pcbModel[1] | pcbModel[2] | pcbModel[3] | 收 | Get PCB Model Number |
| ‘F’ | 6 | 6 | 0 | pcbHwVer[0] | pcbHwVer [1] | pcbHwVer [2] | pcbHwVer [3] | 收 | Get PCB Hardware Version |
| ‘F’ | 6 | 7 | 0 | pcbSerialNum[0] | pcbSerialNum[1] | pcbSerialNum[2] | pcbSerialNum[3] | 收 | Get PCB Serial Number |
| ‘F’ | 1 | 8 |  |  |  |  |  | 收 | Calc EEPROM CRC |
| ‘F’ | 6 | 8 | 0 | CRC[3] | CRC[2] | CRC[1] | CRC[0] | 发 | 返回EEPROM CRC |
| ‘F’ | 1 | 9 |  |  |  |  |  | 收 | Write and verify EEPROM |
| ‘F’ | 4 | 9 | errBytes | errBytes | errBytes |  |  | 发 | 返回Write Ext EEPROM错误字节数 errBytes=0，写EERPROM成功；否则失败 |
| ‘F’ | 3 | 10 | start | end |  |  |  | 收 | 读Ext EEPROM  NOTE: 上位机命令参数必须满足 1) 0<=Parm2<Parm3<=max(EEPROM ADDR) 2) Cnt = Parm3-Parm2<=6 |
| ‘F’ | Cnt | EEPROM[start] | EEPROM[start+1] | EEPROM[start+2] | EEPROM[start+3] | EEPROM[start+4] | EEPROM[start+5] | 发 | 返回读Ext EEPROM字节，最多6个字节 |
| ‘F’ | 2 | 11 | start |  |  |  |  | 收 | 读EEPROM，起始地址为start的4个字节  Note: reserve地址写日期，如04/05/2025 |
| ‘F’ | 4 | EEPROM[start] | EEPROM[start+1] | EEPROM[start+2] | EEPROM[start+3] |  |  | 发 | 返回读EEPROM数据 |
| ‘G’ | 1 | 0 |  |  |  |  |  | 收 | Toggle D7 and D8 ON and OFF |
| ‘I’ | 1 | 0 |  |  |  |  |  | 收 | Read Digital Inputs DI\_A[27..0], ID\_A[3..0], and SYNC\_I |
| ‘I’ | 6 | DI\_A[27:24] | DI\_A[23:16] | DI\_A[15:8] | DI\_A[7:0] | DI\_A[3:0] | Bit0:SYNC\_I Bit1:SERIAL\_IN | 发 | 返回Digital Inputs DI\_A[27..0], ID\_A[3..0], and SYNC\_I |
| ‘J’ | 1 | 0 |  |  |  |  |  | 收 | Configure MAX7301 0n STAND Power Distribution |
| ‘J’ | 3 | 1 | 0 | spiData |  |  |  | 收 | Write to external SPI Device，P24 |
| ‘J’ | 3 | 1 | 1 | spiData |  |  |  | 收 | Write to external SPI Device，P25 |
| ‘J’ | 3 | 1 | 2 | spiData |  |  |  | 收 | Write to external SPI Device，P26 |
| ‘J’ | 3 | 1 | 3 | spiData |  |  |  | 收 | Write to external SPI Device，P27 |
| ‘J’ | 3 | 1 | 4 | spiData |  |  |  | 收 | Write to external SPI Device，P28\_P31 |
| ‘J’ | 3 | 1 | 5 | spiData |  |  |  | 收 | Write to external SPI Device，P29 |
| ‘J’ | 1 | 2 |  |  |  |  |  | 收 | Read from external SPI Device |
| ‘J’ | 2 | Cnt[0] | Cnt[1] |  |  |  |  | 发 | 返回读Ext SPI Device结果，Cnt定义如下：  Cnt = **Max7301Input**(MAX7103\_READ\_PORT23);  Cnt <<= 1;  Cnt |= **Max7301Input**(MAX7103\_READ\_PORT21);  Cnt <<= 1;  Cnt |= **Max7301Input**(MAX7103\_READ\_PORT20);  Cnt <<= 8;  Cnt |= **Max7301Input**(MAX7103\_READ\_PORT12\_19);  Cnt &= 0x0fff;  TxMessage.sMsgStruct.Parm1 = Cnt & 0xff;  TxMessage.sMsgStruct.Parm2 = (Cnt >> 8) & 0xff;  TxMessage.sMsgStruct.Length = 2; |
| ‘N’ | 1 | 0 |  |  |  |  |  | 收 | Read ADC result on ch0(AI\_MF\_A0) |
| ‘N’ | 1 | 1 |  |  |  |  |  | 收 | Read ADC result on ch1(AI\_MF\_A1) |
| ‘N’ | 1 | 2 |  |  |  |  |  | 收 | Read ADC result on ch2(AI\_MF\_A2) |
| ‘N’ | 1 | 3 |  |  |  |  |  | 收 | Read ADC result on ch3(AI\_MF\_A3) |
| ‘N’ | 1 | 4 |  |  |  |  |  | 收 | Read ADC result on ch4(AI\_MF\_A4) |
| ‘N’ | 1 | 5 |  |  |  |  |  | 收 | Read ADC result on ch5(AI\_MF\_A5) |
| ‘N’ | 1 | 6 |  |  |  |  |  | 收 | Read ADC result on ch6(AI\_MF\_A6) |
| ‘N’ | 1 | 7 |  |  |  |  |  | 收 | Read ADC result on ch7(AI\_MF\_A7) |
| ‘N’ | 6 | Ch(0..7) | 0 | Iq32[3] | Iq32[2] | Iq32[1] | Iq32[0] | 发 | 返回ADC结果，格式为IQ32 |
| ‘O’ | 6 | DO\_A[31..24] | DO\_A[23..16] | DO\_A[15..8] | DO\_A[7..0] | Bit0:SERIAL\_OUT | Enable DO\_A  1: Set DO\_EN 0: Clear DO\_EN | 收 | Write to Digital Outputs DO[23..0] and DMO[31..24] |
| ‘P’ | 1 | X |  |  |  |  |  | 收 | 读BEL\_SNS, MEL\_SNS, and PEL\_SNS |
| ‘P’ | 1 | Cpld\_status |  |  |  |  |  | 发 | Read CPLD Loop Enable Sense inputs  Cpld\_status=ReadCpld(0x4) |
| ‘Q’ | 1 | X |  |  |  |  |  | 收 | 读CPLD Faults |
| ‘Q’ | 1 | Cpld\_faults |  |  |  |  |  | 发 | Read CPLD Loop Faults  Cpld\_status=ReadCpld(0x3) |
| ‘R’ | 1 | X |  |  |  |  |  | 收 | Test The External SRAM (U43) |
| ‘R’ | 1 | errCount |  |  |  |  |  | 发 | 返回Ext SRAM测试结果 errCount=0，测试通过；否则失败 |
| ‘S’ | 1 | B0-V5\_ON B1-AD\_TEST B2-DSPTEST |  |  |  |  |  |  | Set or Clear V5\_ON, AD\_TEST, and DSPTEST B0=0，Clear; B0=1，Set V5\_ON;以此类推 |
| ‘T’ | 1 | X |  |  |  |  |  |  | Change Watchdog timer period to 325mS: Force Watchdog timer to time out |
| ‘V’ | X | X |  |  |  |  |  |  | Get DSP Firmware Version |
| ‘V’ | 2 | 01 | 02 | 01 |  |  |  |  | 返回DSP FW Version |
|  |  |  |  |  |  |  |  |  |  |

1. **EEPROM Data Format**
   1. The address range test of the EEPROM will be performed by writing a unique data value to each EEPORM page. The EEPROM has 256-page; each page has 32-byte. In this test, the data value will be written started with 0 and increment by one for each new page. For instance, page 1 will be written with the value of 0, page 2 will be written with the value of 1, and last page 256 will be written with 256. The test will verify that all EEPROM addresses can be accessed by reading the data back from each address and calculating the sum of the reading data, and verify the sum of data is equal to 1,144,480.

This test requirement is done in function EpromTest(void), and the command “E 1 0” will trigger DSP to run it.

* 1. **EepromArray定义**

EEPROM Chip有两个，GPD PCBA的U1-25AA640A，DTR PCBA的U38-25LC640。 EepromArray的内容需要参考客户定义，分析源代码，其定义如下表，也可以根据客户要求修改。

Note: **Address = Row\*8 + offset**

**DTR PCBA-U38 EepromArray定义如下表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Row\offset** | **0x00** | **0x01** | **0x02** | **0x03** | **0x04** | **0x05** | **0x06** | **0x07** |
| **0x00** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] |
| **0x01** | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] |
| **…** |  |  |  |  |  |  |  |  |
| **0x200** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] |
| **0x201** | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] |
| **…** |  |  |  |  |  |  |  |  |
| **0x400** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] |
| **0x401** | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] |

此外，CAN ID Password和CAN ID保存在以下地址：

**EEPROM[0x1ff4] = 0xA5**

**EEPROM[0x1ff5] = 0x77**

**EEPROM[0x1ff6] = 0x88**

**EEPROM[0x1ff7] = 0x94**

**GPD PCBA-U1 EepromArray定义如下表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Row\offset** | **0x00** | **0x01** | **0x02** | **0x03** | **0x04** | **0x05** | **0x06** | **0x07** |
| **0x00** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | 0 | 0 | 0 | 0 |
| **0x01** | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] |
| **0x02** | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] | offsCalErr[3] | offsCalErr[2] | offsCalErr[1] | offsCalErr[0] |
| **0x03** | GainCalErr[3] | GainCalErr[2] | GainCalErr[1] | GainCalErr[0] | 0 | 0 | 0 | 0 |
| **…** |  |  |  |  |  |  |  |  |
| **0x200** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | 0 | 0 | 0 | 0 |
| **0x201** | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] |
| **0x202** | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] | offsCalErr[3] | offsCalErr[2] | offsCalErr[1] | offsCalErr[0] |
| **…** |  |  |  |  |  |  |  |  |
| **0x400** | CRC[3] | CRC[2] | CRC[1] | CRC[0] | 0 | 0 | 0 | 0 |
| **0x401** | pcbModel[3] | pcbModel[2] | pcbModel[1] | pcbModel[0] | pcbHwVer[3] | pcbHwVer[2] | pcbHwVer[1] | pcbHwVer[0] |
| **0x402** | pcbSerialNum[3] | pcbSerialNum[2] | pcbSerialNum[1] | pcbSerialNum[0] | offsCalErr[3] | offsCalErr[2] | offsCalErr[1] | offsCalErr[0] |