应用指南AN-104
InnoSwitch4-Pro产品系列

编程手册

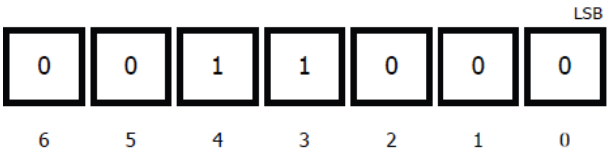
简介

本手册介绍软件实现，包括用于控制InnoSwitch4-Pro操作的驱动程序库。本文档的重要内容是计算各种配置的编程值，如电压、电流、输出线压降补偿、恒功率、防止任何意外行为的I²C命令序列、器件响应和代码示例。

本手册将使用以下约定：
[A] - 从ack
[a] - 主ack
[na] - 主NACK
[W] - 写命令(1’b0)
[r] - 读命令(1’b1)
[PI_COMMAND] - PI Command寄存器地址分配
[TELEMETRY_REGISTER_ADDRESS] - 遥测寄存器地址分配

InnoSwitch4-Pro的I²C通信

InnoSwitch4-Pro的7个比特位从控地址为0x18 (7’b001 1000)

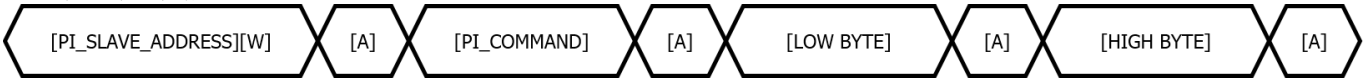


InnoSwitch4-Pro的写操作

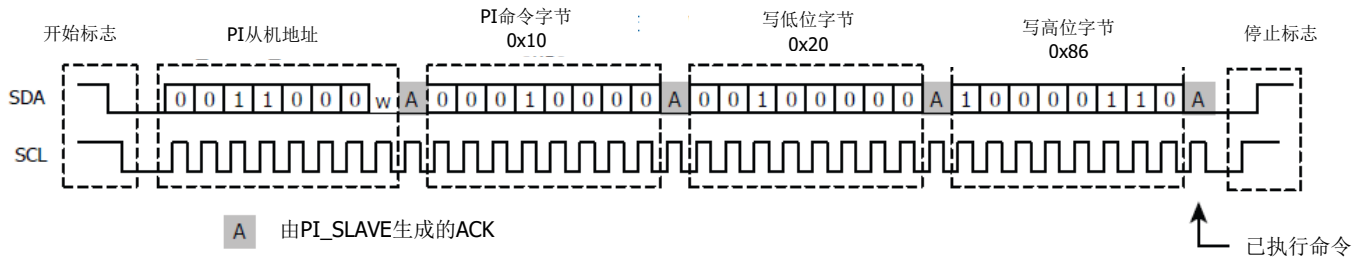
I²C写操作格式如下：

对于1字节数据，写入

对于1字或2字节数据，写入

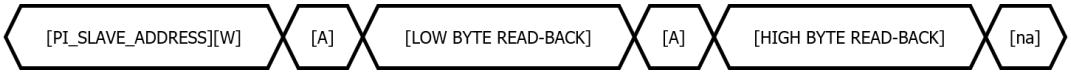


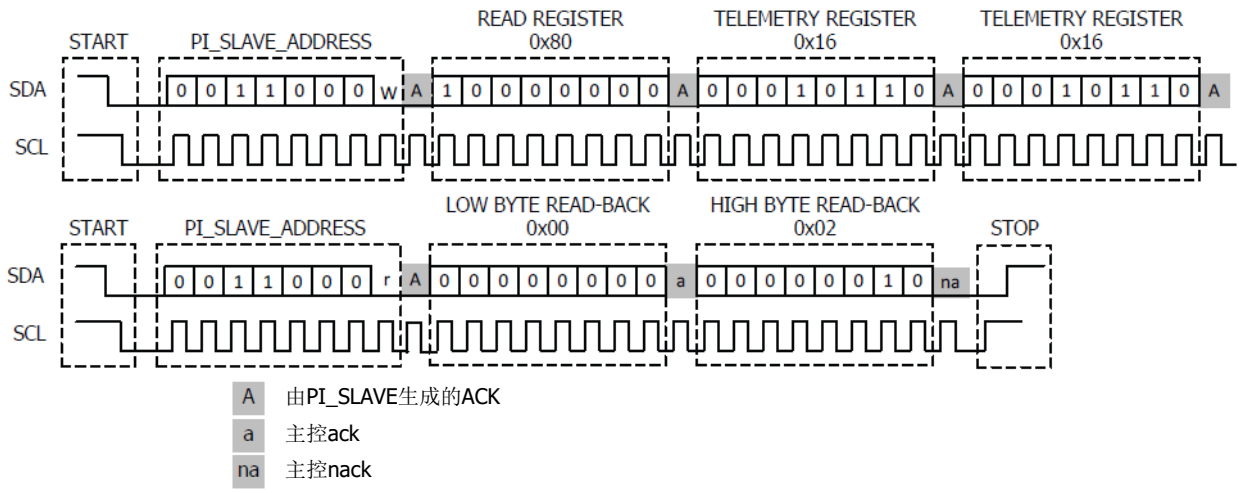
下面的命令说明了向CV寄存器写入8V时的I²C数据包。原本为0x18 (7'b001 1000)的[PI_SLAVE_ADDRESS]向左移动1个比特位，以占用通过I²C通信发送的第一个字节的前7个比特位。该字节的LSB用于命令R/W操作。向该比特位写入‘**0**’表示向从控中的任何[PI_COMMAND]写入数据，向该比特位写入‘**1**’表示从任何[TELEMETRY_REGISTER_ADDRESS]读取数据。因此，对于写操作，[PI_SLAVE_ADDRESS]与R/W比特位上的‘0’相结合，表示写操作，并通过I²C通信发送0x30作为第一个字节，如下图所示。



InnoSwitch4-Pro的读操作

I²C读操作格式如下:



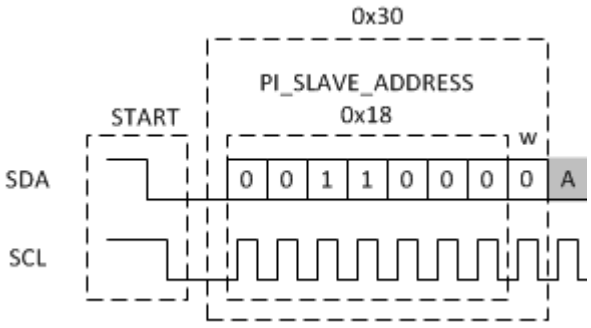


A 由PI_SLAVE生成的ACK

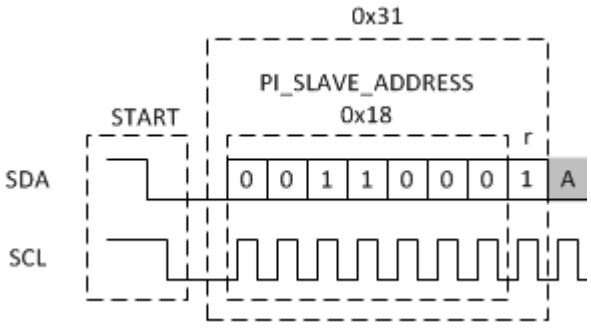
a 主控ack

na 主控nack

首先将需要从中读取数据的遥测寄存器的地址写入读寄存器，读寄存器就像指向该遥测寄存器的指针。读寄存器的地址为0x80。由于读取时的第一个操作是写操作，因此I²C事务的第一个字节如下所示：



然后在下一个I²C操作中，向从控发送读取请求，作为对该请求的响应，从控发送低位字节，然后发送存储在相应遥测寄存器中的高位字节，该遥测寄存器的地址已写入读寄存器。I²C事务的此读取操作期间的第一个字节如下：



恒压设置计算

输出电压设置值根据其具体分辨率（例如10mV/LSB）计算。建议在代码中设置该参数的上限和下限，以确保器件工作于正确的范围内。

输出电压计算:

寄存器	调整范围	精度
CV	3V至24V	10mV/LSB

公式：

$$LSB\,表示形式 = \frac{设置点(V)}{分辨率}$$

示例：

x伏特转换为十进制表示

$$LSB\,表示形式 = \frac{x}{\frac{10mV}{LSB}} = \frac{x}{\frac{10}{1000}V}LSB$$

$$LSB\,表示形式 = x * 100$$

这可以转换为等效的十六进制值，并且必须添加奇校验

代码示例

饱和宏设置输出电压的下限和上限。任何超过2400的**fTemp**值都将被限制为2400。同样，任何小于300的值都将被限制为300。

```
#define INNO4PRO_CV_SET_PT_MULT (float)(100)
#define
sig_minmax(sig,min,max)((sig<min)?sig=min:(sig >max)?sig=
max:0)

float Inno4Pro_Compute_CV( float fSetPt)
{
          float fTemp = 0;
          fTemp = (float)(fSetPt *
                          INNO4PRO_CV_SET_PT_MULT);
          sig_minmax (fTemp,300,2400); //Set Limits
          return fTemp;
`
```

恒流设置计算

恒流(CC)调节设置点计算为IS与GND引脚之间的检测电阻所设置的满量程CC阈值的百分比。可以在20%到100%的范围内进行设定。

ISV(TH)参数值（典型值约为32mV）被视为满量程CC调节电压阈值。对于CC设置点，该值被视为100%，其在十进制表示中被设定为192。如果使用10mΩ检测电阻，则3.2A将被视为满量程CC设置点。如果需要对小于3.2A的CC设置点值进行编程，则必须将其计算为满量程CC设置点的百分比，然后需要将百分比值转换为等效的十进制表示，如下所示。

公式：

$$百分比 = CC\,设置点(A) * 检测电阻(m\Omega) * \frac{100}{Isv_{(Th)}\,(mV)}$$

$$十进制等值 = 百分比 * \frac{192}{100}$$

示例：
Rsense = 10mΩ
CC设置点 = 1.6A

$$百分比 = 1.6A * 10\,m\Omega * \frac{100}{32\,mV}$$

$$百分比 = 50\,\%$$

$$十进制等值 = 50 * \frac{192}{100} = 96$$

这可以转换为等效的十六进制值，并且必须添加奇校验。

代码示例

CC设置点的计算

$$设置点 = 电流 * Rsense * \frac{192}{32}$$

```
#define INNO4PRO_RSENSE (float)(5)
#define INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE (float)(32)
#define INNO4PRO_ADC_FULL_RANGE (float)(192)

 #define INNO4PRO_CC_SET_PT_MULT (float)
                    ((INNO4PRO_ADC_FULL_RANGE *
                      INNO4PRO_RSENSE) /
                      INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE)

float Inno4Pro_Compute_CC( float fSetPt)
{
          float fTemp = 0;
          fTemp = (float)(fSetPt *
                    INNO4PRO_CC_SET_PT_MULT);
          sig_minmax (fTemp,25,192); //Set Limits
          return fTemp;
}
```

恒功率拐点电压设置计算

与写入恒压设置类似，由于恒功率拐点电压设置(VKP)的精度为100mV，因此VKP设置点需要乘以10才能转换为要写入的等效十进制值。可以在5.3V到24V的范围内进行设定。

公式：

$$LSB表示形式 = \frac{设置点(V)}{分辨率}$$

示例：
x伏特转换为十进制表示

$$LSB表示形式 = \frac{x}{\frac{100mV}{LSB}} = \frac{x}{\frac{100}{1000}V}LSB$$

$$LSB表示形式 = x * 10$$

这可以转换为等效的十六进制值，并且必须添加奇校验

代码示例

建议在程序中使用V_{KP}设置点的最小限值53和最大限值240。

```
#define INNO4PRO_VKP_SET_PT_MULT (float)(10)

float Inno4Pro_Compute_VKP( float fSetPt)
{
          float fTemp = 0;
          fTemp = (float)(fSetPt *
                    INNO4PRO_VKP_SET_PT_MULT);
          sig_minmax (fTemp,53,240); //Set Limits
          return fTemp;
}
```

输出线压降补偿设置计算

输出线压降补偿(CDC)可在0至600mV范围内以50mV的增量进行设定。600mV对应于12LSB，0mV相当于0LSB。

公式：

$$LSB表示形式 = \frac{设置点(V)}{分辨率}$$

示例：
x伏特转换为十进制表示

$$LSB表示形式 = \frac{x}{\frac{50mV}{LSB}} = \frac{x}{\frac{50}{1000}V}LSB$$

$$LSB表示形式 = x * 20$$

这可以转换为十六进制值以进行编程。

代码示例

CDC设置点的值限制为0和12

```
#define INNO4PRO_CDC_SET_PT_MULT (float)(50)

float Inno4Pro_Compute_CDC( float fSetPt)
{
          float fTemp = 0;
          fTemp = (float)(fSetPt *
                    INNO4PRO_CDC_SET_PT_MULT);
          sig_minmax (fTemp,0,12); //Set Limits
          return fTemp;
}
```

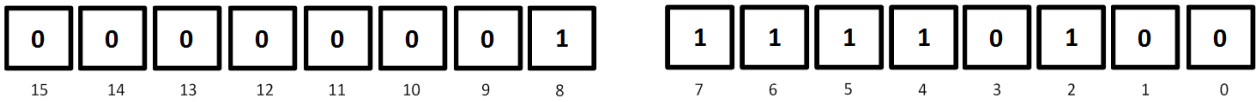
奇偶校验位实现

一些寄存器具有使用奇校验检查的简单错误检测机制。在奇校验位错误检查中，值（包括奇偶校验位）的二进制格式中1的总数必须是奇数。选择示例：

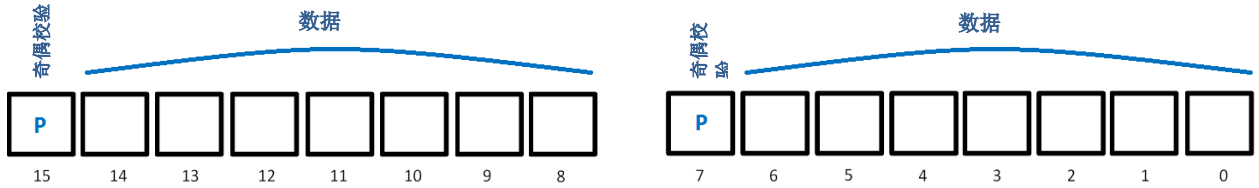
寄存器	输出电压	LSB表示形式	无奇偶校验的十六进制	有奇校验的十六进制
CV	5V	500	0x01F4	0x83F4

性寄存器在每个低位字节和高位字节上都包含奇偶校验位。所有寄存器的详细信息见数据手册。

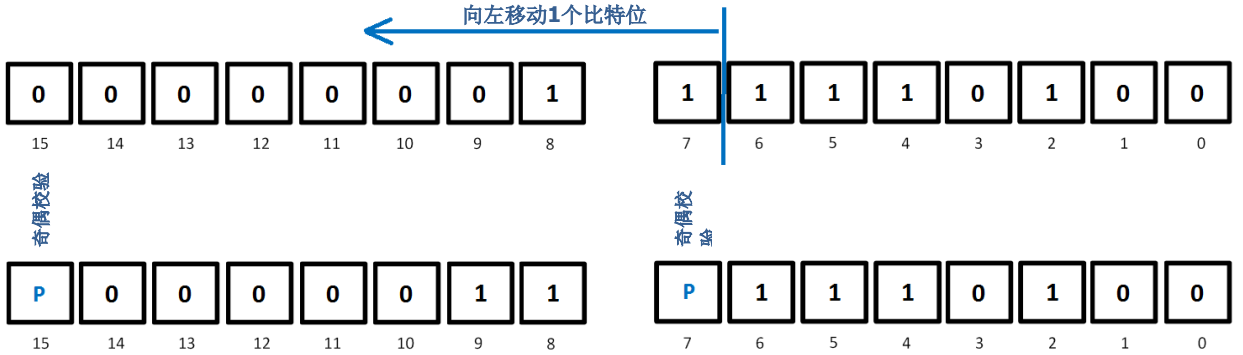
简单的二进制转换：



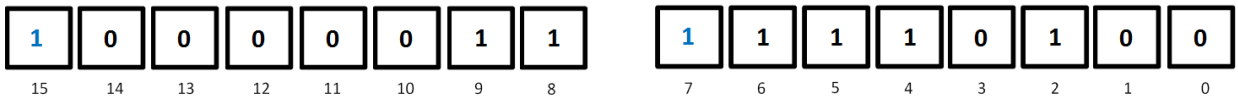
某些寄存器要求使用比特位[15]和比特位[7]来实现奇校验位；这些比特位不得包含转换后的二进制数据。



将数据转换为二进制时，从低位字节第7个比特位开始的二进制数据必须向左移动一位，以预留奇偶校验位的位置。



如果数据有奇数个1，则奇偶校验位为**0**，否则设为**1**。



奇偶校验代码示例

```
 bool Inno4Pro_OddParity(uint8_t u8OddParity)
{
          u8OddParity ^= (u8OddParity >> 4);
          u8OddParity ^= (u8OddParity >> 2);
          u8OddParity ^= (u8OddParity >> 1);
          return u8OddParity & 1;
}

void Inno4Pro_Encode_Buffer_Parity(uint16_t u16Temp, uint8_t *u8WriteBuffer)
 {
          uint16_t u16TempMsb = 0;
          uint8_t u8ConvertedMsb = 0;
          uint8_t u8ConvertedLsb = 0;

          // Clears Bit 0-6 and Shift the remaining to left by 1
          // The 7th Bit is used for Parity Purposes
          // Example for 5V : 01F4 Hex (500 in decimal) , Returns 0x300
          u16TempMsb = (u16Temp & 0xFF80) << 1;

          // Begin MSB Extraction
          // From 0x300 , Returns 0x03
          u8ConvertedMsb = (u16TempMsb & 0xFF00) >> 8;

          // Check Odd Parity and Fill the MSB buffer
          if(Inno4Pro_OddParity(u8ConvertedMsb))
          {
                    // No of Zero is Odd
                    u8WriteBuffer[1] = u8ConvertedMsb;
          }
          else
          {
                    // No of Zero is Even
                    u8WriteBuffer[1] = set_bit(u8ConvertedMsb,7);  //Sets bit[7]
          }

          // Clears 7th Bit, This is used for parity purposes
          // Example for 5V : 01F4 Hex (500 in decimal) , Returns 0x74
          u8ConvertedLsb = (u16Temp & 0x7F);

          // Check Odd Parity and Fill the LSB buffer
          if(Inno4Pro_OddParity(u8ConvertedLsb))
          {
                    // No of Zero is Odd
                    u8WriteBuffer[0] = u8ConvertedLsb;
          }
          else
          {
                    // No of Zero is Even
                    u8WriteBuffer[0] = set_bit(u8ConvertedLsb,7);  //Sets bit[7]
          }
}
}
```

命令序列

为了更新恒压(CV)和恒流(CC)，需要遵循一定的命令序列，以避免无意中触发欠压(UV)和过压(OV)故障。

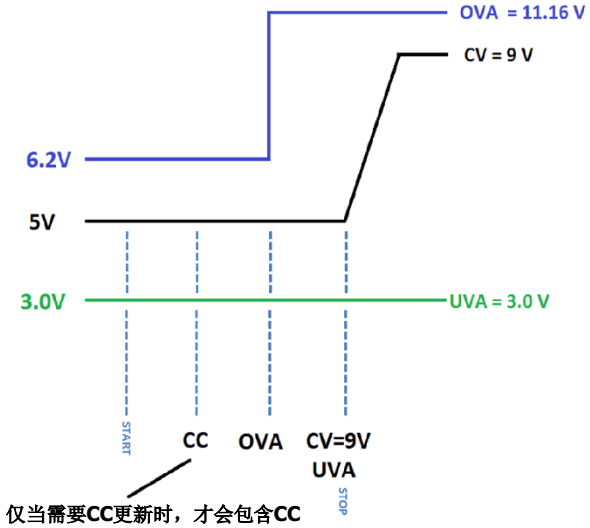
电压递增过程

如果在写入新的CV值之前没有将过压设置点(OVA)增加到更高的值，就会触发OV保护。在对CV寄存器设定之前，需要对OVA进行设定。

对于最初以低输出电压和高输出电流工作，然后转换为高输出电压和低输出电流条件的电源，如果在升高电压之前不降低CC设置点，则电源可能会因在高输出电压下吸收高负载而导致功率受限。在增加输出电压之前，也需要对CC寄存器进行编程。

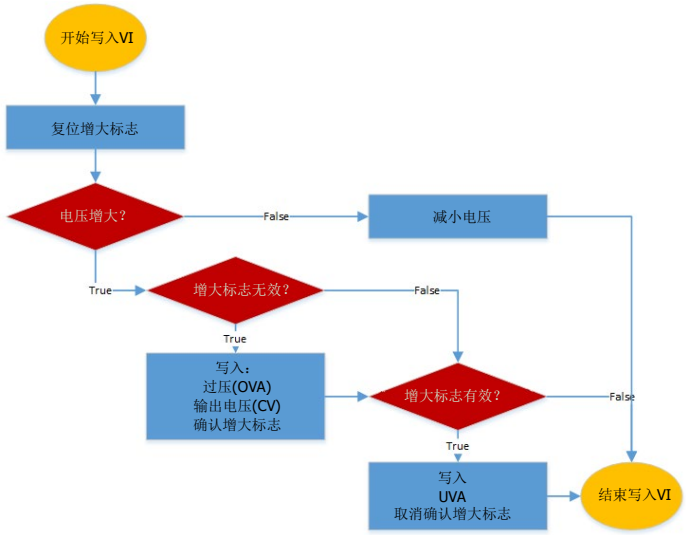
另一种可能发生的情况是，当CC设置低于当前CC设置点时，会导致电源工作于CC区域。这种突然的变化可能会将输出电压降低到UVA设置点以下，并触发UV保护。

下图显示了将电压递增到更高值的命令序列，初始化关闭10ms更新限制（快速VI命令）。无论FAST VI命令寄存器的状态如何，通过I^2C总线传输的连续命令之间仍需要至少150µs的延迟。



在上图中，命令序列为CC、OVA、CV和UVA。CC设置点仅在收到请求时才会更新。上面的示例显示OVA跟踪CV值。OVA设置值始终比CV值高24%。

电压递增流程图



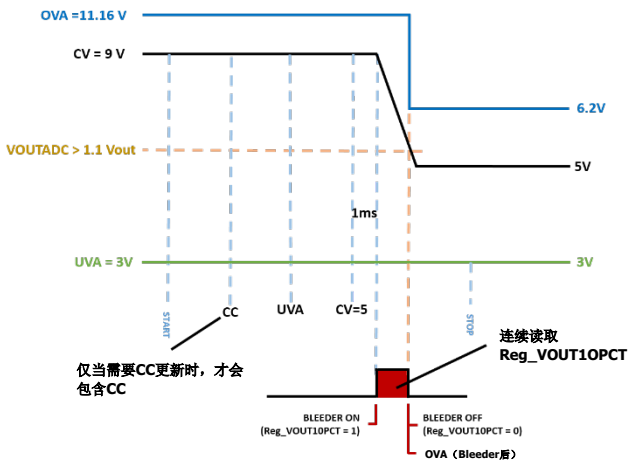
电压递增代码示例

```
//Voltage Increase Routine
if(bVoltIncrease)
{
   //Initial Command Sequence
   //WR_WORD indicates that there are 2 bytes to be sent
   if(!bControlFlag_Increase)
   {
     I2C_Write16( INNO4PRO_ADDRESS, INNO4PRO_OVA,
                  u8_Buffer_OVA,WR_WORD)
     I2C_Write16( INNO4PRO_ADDRESS, INNO4PRO_CV,
                  u8_Buffer_CV,WR_WORD)
     bControlFlag_Increase = true;
   }

   if(bControlFlag_Increase)
   {
         //Check If Vout already reached 90% of the
         desired Set Point
         if(Inno4Pro_Read_Volts() >
            (Inno4Pro_Get_Register_CV()*0.9))
         {
                  I2C_Write16(INNO4PRO_ADDRESS,
                              INNO4PRO_UVA,
                              u8_Buffer_UVA,
                              WR_WORD)
                  //New Set Point Was Reached
                  bVoutIncOk = true;

                  bControlFlag_Increase = false;
          }
    }
//Return Increment Voltage Status
return bVoutIncOk;

}
```

电压递减过程

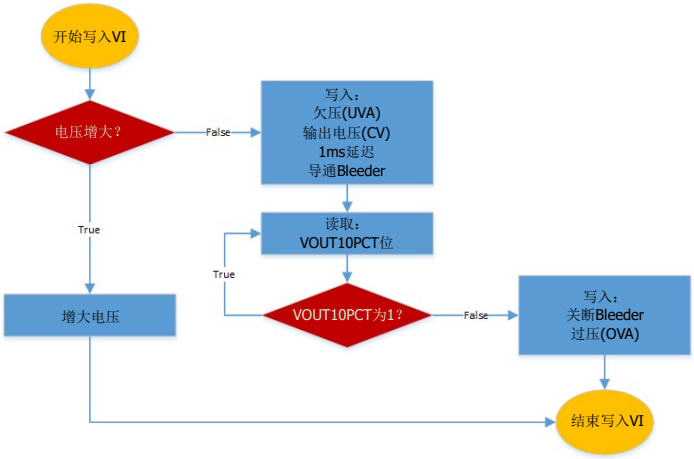
当输出电压设置为较低值时，需要使能BLEEDER，以加快输出电压的转换，尤其是在空载条件下。如果BLEEDER在较大的电压转换期间（例如20V至5V）未导通，则在没有开关活动的较长转换时间内可能会触发InnoSwitch4-Pro IC的自动重启动(AR)。一旦电压达到所需的设置点，就需要关断BLEEDER，否则可能导致IC出现不良的功率耗散和发热。

通常，在额定功率相同的情况下，与较高的输出电压相比，在较低的输出电压下可以提供较大的负载电流。因此，在转换到较低的输出电压时，建议先增加CC设置点限值，然后再降低输出电压。这可以防止电源进入恒流(CC)工作模式，尤其是在转换之前电源一直在恒功率区域工作的情况下。在CC工作模式下，如果输出电压低于UVA，由于转换前未增加CC限值而导致电源意外工作，也会触发UV保护。



在上图中，首先对CC设置点进行编程。在更新CV寄存器之前，需要将UVA设置为低于新CV设置点的值。BLEEDER应在新的CV值写入1ms后开启。当BLEEDER导通时，应持续监测输出电压。在输出电压达到CV设置点的110%后，READ10寄存器的VOUT10PCT位将被清除或设置为0。该位通知何时必须关断BLEEDER，以防止控制器出现不必要的功率耗散。OVA设置点只能在BLEEDER关断后进行更新，以防止触发OVP。

电压递减流程图



电压递减代码示例

```
//Voltage Decrease Routine
else
{
   //Initial Command Sequence
   //WR_WORD indicates that there are 2 bytes to be sent
   I2C_Write16(INNO4PRO_ADDRESS, INNO4PRO_UVA,
               u8_Buffer_UVA,WR_WORD);
   I2C_Write16(INNO4PRO_ADDRESS, INNO4PRO_CV,
                u8_Buffer_CV,WR_WORD);
   __delay_ms(1);
   Inno4Pro_Bleeder_Enable(true);

   do
   {
         bVout10pct_Flag =
         Inno4Pro_Read_Status_Vout10pct();
   }while (bVout10pct_Flag == true);

   Inno4Pro_Bleeder_Enable(true);

   I2C_Write16(INNO4PRO_ADDRESS, INNO4PRO_OVA,
                u8_Buffer_OVA, WR_WORD)

   bVoutDecOk = true;

   return bVoutDecOk;
}
```

计时器的使用

快速VI命令计时器

由于FAST VI寄存器的原因，连续的CV和CC I²C写命令的发送速度不能超过10ms。必要时，可以禁止用于设置输出电压/电流的连续命令的10ms更新限值。默认情况下，CV/CC更新速度处于使能状态。固件实现必须有一个**计时器时钟**来处理必要的时序和延迟。这通常是一个通过中断运行的1ms计时器。

使能CV/CC更新限值时的电压递增

下面的示例在CV和CC I²C写命令之间有12ms的延迟

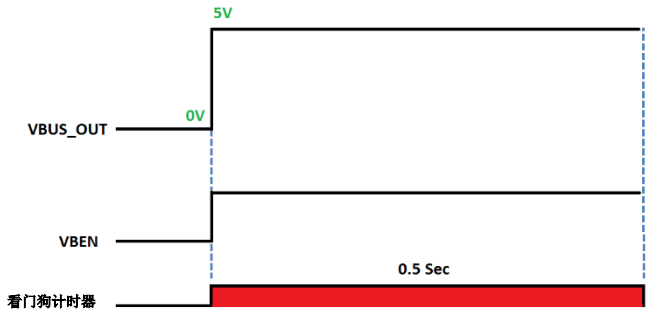
```
#define INNO4PRO_VI_STATE_DELAY (uint16_t)(6)

//Voltage Increase Routine
if(bVoltIncrease)
{
   if(clock_HasTimeElapsedMs(u16_Config_Timer_Hi,INNO4PRO_VI_STATE_DELAY)) //Delay Time
   {
      switch(u16_Config_State_Hi)
      {
          case 0; break; //Delay
                         //I2C Address ,PI COMMAND LSB & MSB  TYPE
          case 1; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_CC,u8_Buffer_CC ,WR_WORD);     break;
          case 2; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_OVA ,u8_Buffer_OVA ,WR_WORD);  break;
          case 3; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_CV,u8_Buffer_CV ,WR_WORD);     break;
          case 4; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_UVA ,u8_Buffer_UVA ,WR_WORD);  break;

          default: break;
      }

      u16_Config_State_Hi++;
      u16_Config_State_Hi = clock_GetTimeStampMs();

      //Maximum Time to complete voltage transitions
      if(u16_Config_State_Hi > INNO4PRO_OUTPUT_HI_TIME) // Delay
      {
         //Reset variables
         u16_Config_State_Hi = 0;
         u16_Config_Timer_Hi = 0;
         return true;
      }
      else
      {
         return false;
      }
   }

}
```

使能CV/CC更新限值时的电压递减

```
//Voltage Decrease Routine
else
{
   if(clock_HasTimeElapsedMs(u16_Config_Timer_Lo,INNO4PRO_VI_STATE_DELAY)) //Delay Time
   {
      switch(u16_Config_State_Hi)
      {
          case 0; break; //Delay
                         //I2C Address ,PI COMMAND LSB & MSB  TYPE
          case 1; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_CC,u8_Buffer_CC ,WR_WORD); break;
          case 2; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_UVA ,u8_Buffer_OVA ,WR_WORD); break;
          case 3; I2CWrite16(INNO4PRO_ADDRESS  ,INNO4PRO_CV,u8_Buffer_CV ,WR_WORD);

                  //1ms Delay
                  __delay_ms(1);

                  //BLEEDER Turn on Control
                  //Immediately Executed after CV
                  //Enable BLEEDER
                  u8_Buffer_BLEEDER[0] = 0x01;
                  u8_Buffer_BLEEDER[1] = 0x00;

                  //Write BLEEDER ON
                  I2CWrite16(INNO4PRO_ADDRESS,INNO4PRO_BLEEDER ,u8_Buffer_BLEEDER ,WR_WORD);
                  break;

          default: break;
      }

      u16_Config_State_Lo++;
      u16_Config_State_Lo = clock_GetTimeStampMs();

      //BLEEDER Turn Off Control
      if(bBleederTurnOffCntrl)
      {
         if(!Inno4Pro_VOUT10PCT_Enabled())
         {
            //Disable BLEEDER
            u8_Buffer_BLEEDER[0] = 0x00;
            u8_Buffer_BLEEDER[1] = 0x00;

            //Write BLEEDER Off
            I2CWrite16(INNO4PRO_ADDRESS,INNO4PRO_BLEEDER ,u8_Buffer_BLEEDER ,WR_WORD);


            //OVA must be executed after BLEEDER turn Off to avoid OVP trigger
            I2CWrite16(INNO4PRO_ADDRESS,INNO4PRO_OVA ,u8_Buffer_OVA ,WR_WORD);

            bVOUT10PCT_disabled = true;
         }
      }

      //Maximum Time to complete voltage transition
      //Must be longer than BLEEDER turn OFF
      if(bVOUT10PCT_disabled)
      {
         //Reset variables
         u16_Config_State_Hi = 0;
         u16_Config_Timer_Hi = 0;
         return true;
      }
      else
      {
         return false;
      }
   }
```

看门狗计时器

默认情况下，看门狗计时器已使能并设置为0.5秒。在这段时间内，至少需要进行一个I²C事务，以防止InnoSwitch4-Pro返回默认复位。



看门狗代码示例

```
void main(void)
{
  //Initialize the device
  SYSTEM_Initialize();
  INTERRUPT_GlobalInterruptEnable();
  INTERRUPT_PeripheralInterruptEnable();

  //Write Initial Commands to Inno4-Pro
  Inno4Pro_Initialization();

  //Call the Functions on the Main Loop
  while (1)
  {
    //Inno4-Pro Control Functions
    Inno4Pro_Write_VI(5 ,5.3);       //5V and 5.3A
    Inno4Pro_Vbus_Switch_Control(3); //VBEN Enable
    Inno4Pro_ReadVolts();            //Measure Voltage
    Inno4Pro_ReadAmps();             //Measure Current
  }
}
```

遥测/读回

I²C主控可使用遥测寄存器读取用户编程的寄存器、监测和测量输出参数、更新器件保护功能以及检测故障情况。

在使用遥测功能之前，必须根据所用微控制器的数据手册设置I²C读/写驱动程序。

系统就绪信号

InnoSwitch4-Pro必须在任何I²C事务开始之前表明它已准备好接收I²C命令。这可以通过读取READ10寄存器上的Reg_control_s位的状态来监控。对该位的确认表示InnoSwitch4-Pro已做好通信和接受命令的准备。

系统就绪代码示例

```
bool Inno4Pro_Read_Status_SystemReady(void)
{
  //READ10, System Ready Signal
  return Inno4Pro_Read_Bit(INNO4PRO_READ10,
                           READ10_Reg_CONTROL_S);
}
```

VOUT10%信号

每当输出电压从高电压设置点转换到低电压设置点时，InnoSwitch4-Pro都会在每次转换开始时确认VOUT10PCT。该器件监测输出电压ADC，然后在输出电压稳定到所设稳压阈值的10%以下时清除Reg_VOUT10PCT寄存器。

VOUT10% 代码示例

```
bool Inno4Pro_Read_Status_Vout10pct(void)
{
  //READ10, VOUTADC > 1.10 * Vout
  return Inno4Pro_Read_Bit(INNO4PRO_READ10,
                           READ10_Reg_VOUT10PCT);
}
```

I²C读回代码示例

Inno4Pro_Telemetry函数是用于读取所需寄存器地址的API。它使用的I²C_Read16函数是专为InnoSwitch4-Pro开发的I²C驱动程序。当使用该函数读取遥测寄存器的值时，将返回MSB值，然后返回LSB值。例如，5V读数的电压DAC值为0x01F4。

```
uint16_t Inno4Pro_Telemtry(uint8_t ReadBack_Address)
{
  uint16_t u16TempRead = 0;
  //I2C_Read16 reads 16 bits of data
  u16TempRead = Inno4Pro_Read16(INNO4PRO_READ10,
                                ReadBack_Address);

  return u16TempRead;
}
```

读取1位遥测

Inno4Pro_Read_Bit函数是用于读取遥测寄存器所需比特位的API。该实现还使用I²C_Read16函数。该函数返回值为1或0。

```
bool Inno4Pro_Read_Bit(uint8_t ReadBack_Address,
                       uint8_t Bit)
{
  uint16_t u16TempRead = 0;
  //I2C_Read16 reads 16 bits of data
  u16TempRead = Inno4Pro_Read16(INNO4PRO_READ10,
                                ReadBack_Address);

  if(test_bit(u16TempRead,Bit))
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

读取1字节遥测

Inno4Pro_Read_Byte函数是用于读取遥测寄存器所需字节的API。用户可以选择MSB或LSB。

```
uint8_t Inno4Pro_Read_Byte(uint8_t ReadBack_Address,
                           bool bHighByte)
{
  uint16_t u16TempRead = 0;
  //I2C_Read16 reads 16 bits of data
  u16TempRead = Inno4Pro_Read16(INNO4PRO_READ10,
                                ReadBack_Address);

  if(bHighByte)
  {
    return (u16TempRead & 0xFF00) >> 8;
  }
  else
  {
    return (u16TempRead & 0x00FF);
  }
}
```

读取2位遥测

Inno4Pro_Read_2Bits函数是用于读取遥测寄存器的2个比特位的API。

```
uint8_t Inno4Pro_Read_2Bits (uint8_t ReadBack_Address,
                             uint8_t u8ShiftCnt)
{
 uint16_t u16TempRead = 0;
 //I2C_Read16 reads 16 bits of data
 u16TempRead = Inno4Pro_Read16(INNO4PRO_READ10,
                               ReadBack_Address);

 return (u16TempRead >> u8ShiftCnt) & 0x0003;
}
```

读取设置点和阈值

Inno4Pro_Read_SetPoint函数用于读取与电压相关的读回。

```
float Inno4Pro_Read_SetPoint (uint8_t ReadBack_Address,
                              float fMultiplier)
{
 uint16_t u16TempReadValue = 0;
 uint16_t u16ConvertedValue = 0;

 u16TempReadValue = Inno4Pro_Telemetry(ReadBack_Address);

 u16ConvertedValue = ((u16TempReadValue & 0x7F00 >> 1) +
                      (u16TempReadValue & 0x007F);

 return (float) (u16ConvertedValue / fMultiplier;
}
```

读取电压代码示例

测得的输出电压读数使用Inno4Pro_Telemetry函数来获取READ9的值。启动时输出电压默认值设置为5V。执行读取命令时，我们预计MSB和LSB的READ9值分别为0x01和0xF4。该值转换为数值500，根据InnoSwitch4-Pro数据手册，相当于5V读数。

```
float Inno4Pro_Read_Volts(void)
{
 uint16_t u16TempReadValue = 0;
 uint16_t u16ConvertedValue = 0;

 //Read instantaneous output voltage
 u16TempReadValue = Inno4Pro_Telemetry(INNO4PRO_READ9);

 //Clear bit [15:12], use bit [11:0]
 u16ConvertedValue = (u16TempReadValue & 0x0FFF);

 //Calculate Reading - 0x01F4 -> 0b500 -> 5V
 return (float) (u16ConvertedValue /
                 INNO4PRO_CV_SET_PT_MULT);
}

float Inno4Pro_Read_VoltsAverage(void)
{
 uint16_t u16TempReadValue = 0;
 uint16_t u16ConvertedValue = 0;

 //Read average output voltage
 u16TempReadValue = Inno4Pro_Telemetry(INNO4PRO_READ13);

 //Clear bit [15:12], use bit [11:0]
 u16ConvertedValue = (u16TempReadValue & 0x0FFF);

 //Calculate Reading - 0x01F4 -> 0b500 -> 5V
 return (float) (u16ConvertedValue /
                 INNO4PRO_CV_SET_PT_MULT);
}
```

读取电流示例

Inno4Pro_Read_Amps和Inno4Pro_Read_AmpsAverage返回的值均基于固件中设置的Rsense值。确保固件中的Rsense值接近实际Rsense值。

```
float Inno4Pro_Read_Amps(void)
{
 uint16_t u16TempReadValue = 0;
 uint16_t u16ConvertedValue = 0;

 //Read instantaneous output current
 u16TempReadValue = Inno4Pro_Telemetry(INNO4PRO_READ8);

 //Clear bits [15:9] and the parity bit [7]
 u16ConvertedValue = ((u16TempReadValue & 0x0100 >> 1) +
                      (u16TempReadValue & 0x007F);

 //Sensed Current Value = (N  *  32) / (Rsense * 192)
 //Example Calculated Reading
 //Example Rsense is 5 ohms
 //Sense Current Value = (87 * 32) / (Rsense *192) = 2.9A
 return (float) ((u16ConvertedValue *
                  INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE) /
                 (INNO4PRO_RSENSE *
                  INNO4PRO_ADC_FULL_RANGE));
}

float Inno4Pro_Read_AmpsAverage(void)
{
 uint16_t u16TempReadValue = 0;
 uint16_t u16ConvertedValue = 0;

 //Read instantaneous output current
 u16TempReadValue = Inno4Pro_Telemetry (INNO4PRO_READ12);

 //Clear bits [15:9] and the parity bit [7]
 u16ConvertedValue = ((u16TempReadValue & 0x0100 >> 1) +
                      (u16TempReadValue & 0x007F);

 //Sensed Current Value = (N  *  32) / (Rsense * 192)
 //Example Calculated Reading
 //Example Rsense is 5 ohms
 //Sense Current Value = (87 * 32) / (Rsense *192) = 2.9A

 return (float) ((u16ConvertedValue *
                  INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE) /
                 (INNO4PRO_RSENSE *
                  INNO4PRO_ADC_FULL_RANGE));
}
```

代码库

为了简化控制InnoSwitch4-Pro的技术细节，我们提供了一个简单的代码库作为参考。该代码库包含控制器件所需的所有寄存器。这些寄存器分为命令寄存器和遥测寄存器。命令寄存器被发送到器件以进行性能控制，而遥测寄存器用于读回值。提供的计算宏有助于设置点计算。此外，还定义了寄存器默认值，以简化器件初始化时对所需寄存器的写入。

PIC16F18325 MCU实现

实现

标头文件包含

代码库标头文件包含所有函数声明和宏定义。如图所示，这必须包含在主页中。

```
#include "Drv_Rtc.h"
#include "Drv_i2c.h"
#include "Inno4Pro.h"
#include "Inno4Pro_Config.h"
```

InnoSwitch4-Pro 初始化

在连续执行主代码之前，会监控系统就绪信号的状态，以确保InnoSwitch4-Pro已准备好接收I²C命令。之后，可根据需要向器件发送初始化命令，以重新配置默认设置。该初始化例程禁止看门狗计时器和快速VI限制。UVL计时器也被初始化为64ms。

```
Inno4Pro_Initialization();

void main(void)
{
    //Main Loop Codes
}
```

控制函数设置

更新输出电压和恒流设置

这些函数执行以下操作：

- 遵循特定的I²C命令序列，以避免意外触发UV或OV故障
- 当电压从高电平降至低电平设置时，控制VOUT引脚强泄放
- 自动更新过压(OVA)和欠压(UVA)设置：
  1. OVA为CV设置点的124%
  2. UVA固定为3V

```
Inno4Pro_Write_VI(Volts, Amps)
```

在无需泄放控制的情况下更新输出电压

```
Inno4Pro_Write_Volts(Volts)
```

设置恒流设置

```
Inno4Pro_Write_Amps(Amps)
```

设置过压设置

```
Inno4Pro_Write_Overer_Volts(Value)
```

设置欠压设置

```
Inno4Pro_Write_Under_Volts(Volts)
```

设置输出线压降补偿值

```
Inno4Pro_Write_Cable_Drop_Comp(Value)
```

设置恒定输出功率阈值

```
Inno4Pro_Write_Volt_Peak(Value)
```

用于导通或关断母线电压开关

```
Inno4Pro_Vbus_Switch_Control(Value)
```

用于导通或关断VOUT引脚强泄放

BLEEDER不能长时间处于使能状态，以防止控制器中的功耗过大

```
Inno4Pro_Bleeder_Enable(Value)
```

遥测函数设置

在主环路上使用遥测函数读取InnoSwitch4-Pro的寄存器。

用于读取所需的寄存器地址

```
Inno4Pro_Telemetry(Register_Address)
```

用于读取遥测寄存器的特定比特位

```
Inno4Pro_Read_Bit(Register_Address, Bit)
```

告知InnoSwitch4-Pro何时准备好进行通信和接受命令

```
Inno4Pro_Read_Status_SystemReady()
```

返回测得的输出电压

```
Inno4Pro_Read_Volts()
```

返回测得的输出电流

```
Inno4Pro_Read_Amps()
```

返回VOUT10PCT状态信息

VOUT10PCT状态用于禁止VOUT引脚强泄放

```
Inno4Pro_Read_Status_Vout10pct()
```

返回VOUT2PCT状态信息

VOUT2PCT状态也用于禁止VOUT引脚强泄放

```
Inno4Pro_Read_Status_Vout2pct()
```

基本代码示例

此代码示例用于演示InnoSwitch4-Pro代码库的基本用法。

- 初始命令使用InnoSwitch4-Pro初始化例程发送。
- 主例程将输出电压设置为5V，恒流设置为6A。
- 输出线压降补偿设定为300mV。
- 恒功率的拐点电压设置为7V。
- VBUS开关导通。

```
//MPLAB Code Configurator Header File
#include “mcc_generated_files/mcc.h”

//Step 1: Add Header Files
#include “Code/Drv_i2c.h”
#include “Code/Drv_Rtc.h”
#include “Code/Inno4Pro_Config.h”
#include “Code/Inno4Pro.h”

void main(void)
{
 //Initialize the device
 SYSTEM_Initialize();
 INTERRUPT_GlobalInterruptEnable();
 INTERRUPT_PeripheralInterruptEnable();

 //Step 2: Write Initialize Commands to InnoSwitch4-Pro
 Inno4Pro_Initialization();

 //Step 3: Call functions on the Main Loop
 while(1)
 {
  //Main Loop Variable Initialization
  //Initialize Voltage at 5V
  float fVolts = 5;
  //Initialize Constant Current at 6A
  float fAmps = 6;
  //Initialize Cable Drop Compensation to 300mV
  float fCableDropComp = 300;
  //Initialize Knee Voltage at 7V
  float fVoltPeak = 7;
  //Initialize Vbus Enable to ON
  float fVbusEn = 3;


  //Library Call in the Mainloop
  //Set Voltage and Current
  Inno4Pro_Write_VI              ( fVolts  ,fAmps );
  //Set Cable Drop Compensation
  Inno4Pro_Write_Cable_Drop_Comp ( fCableDropComp );
  //Set Constant Output Power Knee Voltage
  Inno4Pro_Write_Volt_Peak       ( fVoltPeak     );
  //Set Vbus Enable
  Inno4Pro_Vbus_Switch_Control   ( fVbusEn       );
 }
```

I²C驱动程序

必须根据所使用的微控制器正确配置I²C驱动程序。必须根据InnoSwitch4-Pro数据手册中的I²C数据包格式进行配置，以处理读写事务。每个I²C事务在命令之间至少有150 µs的延迟

I²C写代码示例

```
int I2C_Write16(uint16_t slaveAddress,
                uint8_t dataAddress,
                uint8_t *dataBuffer,
                uint8_t buflen)
{
 //150us delay on every I2C transaction
 __delay_us(150);

 uint8_t writeBuffer[3];
 I2C1_MESSAGE_STATUS status = I2C1_MESSAGE_PENDING;

 //Set address as the bytes to be written first
 writeBuffer[0] = dataAddress;

 //Limit buffer length
 if(buflen > 3)
 {
  buflen = 3;
 }

 //Copy data bytes to write buffer
 writeBuffer[1] = dataBuffer[0];
 writeBuffer[2] = dataBuffer[1];

 //Set up for ACK polling
 timeOut = 0;

 while(status != I2C1_MESSAGE_FAIL)
 {
  //Initiate a write to device
  I2C1_MasterWrite(writeBuffer,buflen,
                   slaveAddress,&status);

  //Wait for the message status to change
  while(status == I2C1_MESSAGE_PENDING);

  //If transfer is complete, break the loop
  if(status == I2C1_MESSAGE_COMPLETE);
  {
   break;
  }

  //If transfer fails, break the loop
  if(status == I2C1_MESSAGE_FAIL)
  {
   break;
  }

  //Check for max retry and skip this byte
  if(timeOut == MAX_RETRY)
  {
   break;
  }
  else
  {
   timeOut++;
  }
 }
 //If the transfer failed, stop at this point
 if(status == I2C1_MESSAGE_FAIL)
 return 1;
}
```

I²C读代码示例

```
int I2C_Read16(uint16_t slaveAddress,
               uint8_t dataAddress)
{
 int check = 0;
 I2C1_MESSAGE_STATUS status = I2C1_MESSAGE_PENDING;

 uint8_t buflen = 0x02;      //Read 2 Bytes
 uint8_t writeDataBuffer[2]; //Write Buffer Array
 uint8_t readDataBuffer[2];  //Read Buffer Array
 uint16_t u16Lsb;            //Storage for LSB
 uint16_t u16Msb;            //Storage for MSB

 //Copy data bytes to write buffer
 writeBuffer[0] = dataAddress;
 writeBuffer[1] = dataAddress;

 //I2C_Write16 has a built in 150us delay
 //Write register address to read
 check = I2C_Write16(slaveAddress, 0x80, writeDataBuffer,
                     0x03);

 //check if address write is successful
 if(check == 1)
  return;

 //Set up for ACK polling
 timeOut = 0;

 //Delay in between write and read commands
 __delay_us(150);

 while(status != I2C1_MESSAGE_FAIL)
 {
  //Initiate a write to device
  I2C1_MasterWrite(writeBuffer,buflen,
                   slaveAddress,&status);

  //Wait for the message status to change
  while(status == I2C1_MESSAGE_PENDING);

  //If transfer is complete, break the loop
  if(status == I2C1_MESSAGE_COMPLETE);
  {
   break;
  }

  //If transfer fails, break the loop
  if(status == I2C1_MESSAGE_FAIL)
  {
   break;
  }

  //Check for max retry and skip this byte
  if(timeOut == MAX_RETRY)
  {
   break;
  }
  else
  {
   timeOut++;
  }
 }
 //If the transfer failed, stop at this point
 ret = readDataBuffer[0];
 ret <<= 8;
 ret |= readDataBuffer[1];
 return ret;
}
```

Arduino实现

实现

标头文件包含

代码库标头文件包含所有函数声明和宏定义。如图所示，这必须包含在主页中。

```
#include "Drv_Rtc.h"
#include "Drv_i2c.h"
#include "Inno4Pro.h"
```

```
#include "Inno4Pro_Config.h"
```

类实例创建

构造一个类实例来调用Inno4Pro_Application内部的函数。构造Inno4Pro_Rtc的类实例是可选的。

```
Inno4Pro_Application Inno4ProApp;
Inno4Pro_Rtc        Inno4ProClk;
```

InnoSwitch4-Pro初始化

在连续执行主代码之前，会监控系统就绪信号的状态，以确保InnoSwitch4-Pro已准备好接收I²C命令。之后，可根据需要向器件发送初始化命令，以重新配置默认设置。该初始化例程禁止看门狗计时器和快速VI限制。UVL计时器也被初始化为64ms。

I²C通信的400kHz时钟频率在初始化时设置。

```
void setup()
{
 Inno4ProApp.Inno3Pro_Initialization();
}
```

控制函数设置

更新输出电压和恒流设置

这些函数执行以下操作：

- 遵循特定的I²C命令序列，以避免意外触发UV或OV故障
- 当电压从高电平降至低电平设置时，控制VOUT引脚强泄放
- 自动更新过压(OVA)和欠压(UVA)设置：
  3. OVA为CV设置点的124%
  4. UVA固定为3V

```
Inno4ProApp.Inno4Pro_Write_VI(Volts, Amps)
```

在无需泄放控制的情况下更新输出电压

```
Inno4ProApp.Inno4Pro_Write_Volts(Volts)
```

设置恒流设置

```
Inno4ProApp.Inno4Pro_Write_Amps(Amps)
```

设置过压设置

```
Inno4ProApp.Inno4Pro_Write_Overer_Volts(Value)
```

设置欠压设置

```
Inno4ProApp.Inno4Pro_Write_Under_Volts(Volts)
```

设置输出线压降补偿值

```
Inno4ProApp.Inno4Pro_Write_Cable_Drop_Comp(Value)
```

设置恒定输出功率阈值

```
Inno4ProApp.Inno4Pro_Write_Volt_Peak(Value)
```

用于导通或关断母线电压开关

```
Inno4ProApp.Inno4Pro_Vbus_Switch_Control(Value)
```

用于导通或关断VOUT引脚强泄放

BLEEDER不能长时间处于使能状态，以防止控制器中的功耗过大

```
Inno4ProApp.Inno4Pro_Bleeder_Enable(Value)
```

遥测函数设置

在主环路上使用遥测函数读取InnoSwitch4-Pro的寄存器。

用于读取所需的寄存器地址

```
Inno4ProApp.Inno4Pro_Telemetry(Register_Address)
```

用于读取遥测寄存器的特定比特位

```
Inno4ProApp.Inno4Pro_Read_Bit(Register_Address,
                              Bit)
```

告知InnoSwitch4-Pro何时准备好进行通信和接受命令

```
Inno4ProApp.Inno4Pro_Read_Status_SystemReady()
```

返回测得的输出电压

```
Inno4ProApp.Inno4Pro_Read_Volts()
```

返回测得的输出电流

```
Inno4ProApp.Inno4Pro_Read_Amps()
```

返回VOUT10PCT状态信息

VOUT10PCT状态用于禁止VOUT引脚强泄放

```
Inno4ProApp.Inno4Pro_Read_Status_Vout10pct()
```

返回VOUT2PCT状态信息

VOUT2PCT状态也用于禁止VOUT引脚强泄放

```
Inno4ProApp.Inno4Pro_Read_Status_Vout2pct()
```

基本代码示例

```
//Step 1: Add Header Files
#include “Code/Drv_i2c.h”
#include “Code/Drv_Rtc.h”
#include “Code/Inno4Pro_Config.h”
#include “Code/Inno4Pro.h”

//Step 2: Create the class instance
Inno4Pro_Application Inno4Pro;

//Step 3: Write initial commands to InnnoSwitch4-Pro
void setup(void)
{
 Inno4ProApp.Inno4Pro_Initialization();
}

//Step 4: Call the functions on the main loop
void loop()
{
 //5V 5.6A, Voltage and Constant Current
 Inno4ProApp.Inno4Pro_Write_VI(5,5.6);

 //300mV,   Cable Drop Compensation
 Inno4ProApp.Inno4Pro_Write_Cable_Drop_Comp(300);

 //7V,       Constant Output Power Knee Voltage
 Inno4ProApp.Inno4Pro_Write_Volt_Peak(7);

 //ON,       Vbus Enable
 Inno4ProApp.Inno4Pro_Vbus_Switch_Control(3);
}
```

I²C驱动程序

必须根据Arduino Wire Library正确配置I²C驱动程序。

https://www.arduino.cc/en/Reference/Wire

必须根据InnoSwitch4-Pro数据手册中的I²C数据包格式进行配置，以处理读写事务。每个I²C事务在命令之间必须至少有150 µs的延迟。

I²C写代码示例

```
void Inno4Pro_I2C::I2C_Write16(uint8_t slaveAddress,
                               uint8_t dataAddress,
                               uint8_t *dataBudder,
                               uint8_t buflen)
{
 //150us delay on every I2C transaction
 delayMicroseconds(150);
 Wire.beginTransmission((uint8_t)slaveAddress);

#if ARDUINO >= 100
 Wire.write((uint8_t)dataAddress);  //Send address
 Wire.write((uint8_t)dataBuffer[0]);
 if(buflen == 3)
 {
  Wire.write((uint8_t)dataBuffer[1]);
 }
#else
 Wire.send((uint8_t)dataAddress);   //Send address
 Wire.send((uint8_t)dataBuffer[0]);
 if(buflen == 3)
 {
  Wire.send((uint8_t)dataBuffer[1]);
 }
#endif

 Wire.endTransmission();
}
```

I²C读代码示例

```
void Inno4Pro_I2C::I2C_Read16(uint8_t slaveAddress,
                              uint8_t dataAddress,
                              uint8_t *dataBudder,
                              uint8_t buflen)
{
 //150us delay on every I2C transaction
 delayMicroseconds(150);
 uint8_t u8Lsb;          //Storage for LSB
 uint8_t u8Msb;          //Storage for MSB

 //Start transmission to device
 Wire.beginTransmission(slaveAddress);

#if (ARDUINO >= 100)
 Wire.write(0x80);
 Wire.write(dataAddress);
 Wire.write(dataAddress);
#else
 Wire.send(0x80);
 Wire.send(dataAddress);
 Wire.send(dataAddress);
#endif
 Wire.endTransmission();

 //150us delay on every I2C Transaction
 delayMicroseconds(150);

 //Start transmission to device
 Wire.beginTransmission(slaveAddress);
 //Send data n-bytes read
 Wire.requestFrom(slaveAddress,(uint8_t)0x02);

#if (ARDUINO >= 100)
 //Example 5V, Returns F4
 u8Lsb = Wire.read();//Receive DATA

 //Example 5V, Returns 01
 u8Msb = Wire.read();//Receive DATA
#else
 //Example 5V, Returns F4
 u8Lsb = Wire.receive();//Receive DATA
```

```
  //Example 5V, Returns 01
  u8Msb = Wire.receive();//Receive DATA
#endif

  Wire.endTransmission();//End transmission
  //Returns 0x01F4
  return ((u8Msb<<8) | (u8Lsb));
}
```

文档

Configurations（配置）

这是包含InnoSwitch4-Pro的所有库宏和配置的标头文件。

Macros（宏）

固件修订宏

定义InnnoSwitch3-Pro代码库的修订号，以跟踪每个版本的更改。

注：

版本格式：v00.00.00

```
#define INNO4PRO_FW_VERSION_MAJOR  '0' ,' 1'
#define INNO4PRO_FW_VERSION_MINOR  '0' .' 2'
#define INNO4PRO FW VERSION TEST   '0' .' 0'
```

饱和宏

用于设置特定参数的限值

```
#define sig_minmax(sig,min,max)((sig < min)? sig = min:
                                (sig > max)? sig = max:0)
#define sig_max(sig,   max)   ((sig > max) ? sig = max : 0)
#define sig_min(sig,   min)   ((sig < min) ? sig = min : 0)
```

位操作宏

用于操作某个字节中的位

```
#define set_bit(address,bit)    (address |=   (1<<bit))
#define clear_bit(address,bit)  (address &= ~ (1<<bit))
#define toggle_bit(address,bit)(address ^=   (1<<bit))
#define test_bit(address,bit)   (address &    (1<<bit))
```

InnoSwitch4-Pro I2C宏

定义InnoSwitch4-Pro的从控地址和I2C写入大小

注：0x0011000 (0x18) – 7位地址方案

```
#define INNO4PRO_ADDRESS    0x18
#define WR_WORD             0x03
#define WR_BYTE             0x02
#define RD_MSB              1
#define RD_LSB              0
```

InnoSwitch4-Pro响应和计时器宏

用于寄存器的响应和计时器设置

注：使用这些宏更新PI Command寄存器设置

```
#define INNO4PRO_VBUS_ENABLE                        3
#define INNO4PRO_VBUS_DISABLE                       0
#define INNO4PRO_VBUS_DISABLE_NORESET               1

#define INNO4PRO_BLEEDER_ENABLE                     1
#define INNO4PRO_BLEEDER_DISABLE                    0
#define INNO4PRO_BLEEDER_ENABLE_AUTODISABLE         3

#define INNO4PRO_LOAD_DISCHARGE_ENABLE              3
#define INNO4PRO_LOAD_DISCHARGE_DISABLE             12
#define INNO4PRO_LOAD_DISCHARGE_ENABLE_NORESET      2

#define INNO4PRO_TURN_OFF_PSU_ENABLE                true
#define INNO4PRO_TURN_OFF_PSU_DISABLE               false

#define INNO4PRO_FASTVI_UPDATE_LIMIT_ENABLE         false
#define INNO4PRO_FASTVI_UPDATE_LIMIT_DISABLE        true

#define INNO4PRO_OVL_FAULT_RESPONSE_NORESPONSE      0
#define INNO4PRO_OVL_FAULT_RESPONSE_LATCHOFF        1
#define INNO4PRO_OVL_FAULT_RESPONSE_AUTORESTART     2
#define INNO4PRO_OVL_FAULT_RESPONSE_DISABLEOUTPUT   3

#define INNO4PRO_UVL_FAULT_RESPONSE_NORESPONSE      0
#define INNO4PRO_UVL_FAULT_RESPONSE_LATCHOFF        1
#define INNO4PRO_UVL_FAULT_RESPONSE_AUTORESTART     2
#define INNO4PRO_UVL_FAULT_RESPONSE_DISABLEOUTPUT   3

#define INNO4PRO_CCSC_FAULT_RESPONSE_NORESPONSE     0
#define INNO4PRO_CCSC_FAULT_RESPONSE_LATCHOFF       1
#define INNO4PRO_CCSC_FAULT_RESPONSE_AUTORESTART    2
#define INNO4PRO_CCSC_FAULT_RESPONSE_DISABLEOUTPUT  3

#define INNO4PRO_ISSC_FAULT_RESPONSE_NORESPONSE     0
#define INNO4PRO_ISSC_FAULT_RESPONSE_LATCHOFF       1
#define INNO4PRO_ISSC_FAULT_RESPONSE_AUTORESTART    2
#define INNO4PRO_ISSC_FAULT_RESPONSE_DISABLEOUTPUT  3

#define INNO4PRO_ISSC_FREQ_THRESHOLD_60KHZ          0
#define INNO4PRO_ISSC_FREQ_THRESHOLD_30KHZ          1
#define INNO4PRO_ISSC_FREQ_THRESHOLD_90KHZ          2
#define INNO4PRO_ISSC_FREQ_THRESHOLD_120KHZ         3

#define INNO4PRO_ISSC_CC_THRESHOLD_16               1
#define INNO4PRO_ISSC_CC_THRESHOLD_32               2
#define INNO4PRO_ISSC_CC_THRESHOLD_48               3
#define INNO4PRO_ISSC_CC_THRESHOLD_64               4
#define INNO4PRO_ISSC_CC_THRESHOLD_80               5
#define INNO4PRO_ISSC_CC_THRESHOLD_96               6
#define INNO4PRO_ISSC_CC_THRESHOLD_112              7

#define INNO4PRO_UVL_FAULT_TIMER_8MS                0
#define INNO4PRO_UVL_FAULT_TIMER_16MS               1
#define INNO4PRO_UVL_FAULT_TIMER_32MS               2
#define INNO4PRO_UVL_FAULT_TIMER_64MS               3

#define INNO4PRO_WATCHDOG_TIMER_NOWATCHDOG          0
#define INNO4PRO_WATCHDOG_TIMER_500MS               1
#define INNO4PRO_WATCHDOG_TIMER_1000MS              2
#define INNO4PRO_WATCHDOG_TIMER_2000MS              3

#define INNO4PRO_CVOL_FAULT_RESPONSE_NORESPONSE     0
#define INNO4PRO_CVOL_FAULT_RESPONSE_LATCHOFF       1
#define INNO4PRO_CVOL_FAULT_RESPONSE_AUTORESTART    2
#define INNO4PRO_CVOL_FAULT_RESPONSE_DISABLEOUTPUT  3

#define INNO4PRO_CVOL_FAULT_TIMER_8MS               0
#define INNO4PRO_CVOL_FAULT_TIMER_16MS              1
#define INNO4PRO_CVOL_FAULT_TIMER_32MS              2
#define INNO4PRO_CVOL_FAULT_TIMER_64MS              3

#define INNO4PRO_INTERRUPT_CONTROL_S_MASK           0x40
#define INNO4PRO_INTERRUPT_BPS_CURR_LO_FAULT_MASK   0x20
#define INNO4PRO_INTERRUPT_CVO_PKLOAD_TIMER_MASK    0x10
#define INNO4PRO_INTERRUPT_ISSC_MASK                0x08
#define INNO4PRO_INTERRUPT_CCSC_MASK                0x04
#define INNO4PRO_INTERRUPT_VOUT_UV_MASK             0x02
#define INNO4PRO_INTERRUPT_VOUT_OV_MASK             0x01

#define INNO4PRO_OTP_FAULT_HYST_40DEG               0
#define INNO4PRO_OTP_FAULT_HYST_60DEG               1

#define INNO4PRO_CVLOAD_DEFAULT                     0x20
#define INNO4PRO_CVLOAD_RECOMMENDED                 0x80

#define INNO4PRO_LOOPSPEED1_DEFAULT                 0x281E
#define INNO4PRO_LOOPSPEED1_RECOMMENDED             0x140A

#define INNO4PRO_LOOPSPEED2_DEFAULT                 0x08C8
#define INNO4PRO_LOOPSPEED2_RECOMMENDED             0x1F84
```

PI_COMMAND寄存器地址分配

定义要控制的命令寄存器

```
#define INNO4PRO_VBEN                               0x04
#define INNO4PRO_BLEEDER                            0x86
#define INNO4PRO_VDIS                               0x08
#define INNO4PRO_TURN_OFF_PSU                       0x8A
#define INNO4PRO_FAST_VI_CMD                        0x8C
#define INNO4PRO_CVO                                0x0E
#define INNO4PRO_CV                                 0x10
#define INNO4PRO_OVA                                0x92
#define INNO4PRO_UVA                                0x94
#define INNO4PRO_CDC                                0x16
#define INNO4PRO_VBEN                               0x04
#define INNO4PRO BLEEDER                            0x86
```

```
#define INNO4PRO_VDIS                          0x08
#define INNO4PRO_TURN_OFF_PSU                  0x8A
#define INNO4PRO_FAST_VI_CMD                   0x8C
#define INNO4PRO_CVO                           0x0E
#define INNO4PRO_CV                            0x10
#define INNO4PRO_OVA                           0x92
#define INNO4PRO_UVA                           0x94
#define INNO4PRO_CDC                           0x16
#define INNO4PRO_CC                            0x98
#define INNO4PRO_VKP                           0x1A
#define INNO4PRO_CCSC                          0x20
#define INNO4PRO_ISSC                          0xA2
#define INNO4PRO_WATCHDOG_TIMER                0x26
#define INNO4PRO_INTERRUPT                     0x2C
#define INNO4PRO_OTP                           0xAE
#define INNO4PRO_CV_LOAD                       0xB0
#define INNO4PRO_LOOP_SPEED_1                  0x32
#define INNO4PRO_LOOP_SPEED_2                  0x34
#define INNO4PRO_VBUSSC                        0xB6
#define INNO4PRO_DCM                           0xBA
#define INNO4PRO_SRZVS                         0x3E
```

遥测（读回）寄存器地址分配

定义要读取的遥测报回寄存器

```
#define INNO4PRO_READ0                         0x00
#define INNO4PRO_READ1                         0x02
#define INNO4PRO_READ2                         0x04
#define INNO4PRO_READ3                         0x06
#define INNO4PRO_READ4                         0x08
#define INNO4PRO_READ5                         0x0A
#define INNO4PRO_READ6                         0x0C
#define INNO4PRO_READ7                         0x0E
#define INNO4PRO_READ8                         0x10
#define INNO4PRO_READ9                         0x12
#define INNO4PRO_READ10                        0x14
#define INNO4PRO_READ11                        0x16
#define INNO4PRO_READ12                        0x18
#define INNO4PRO_READ13                        0x1A
#define INNO4PRO_READ14                        0x1C
#define INNO4PRO_READ16                        0x20
#define INNO4PRO_READ17                        0x22
#define INNO4PRO_READ_LOOP_SPEED_1             0x26
#define INNO4PRO_READ_LOOP_SPEED_2             0x28
```

InnoSwitch4-Pro读寄存器位移计数宏

定义读取寄存器值时使用的寄存器移位位置

```
#define INNO4PRO_READ_OV_FAULT_BITSHIFT          14
#define INNO4PRO_READ_UV_FAULT_BITSHIFT          12
#define INNO4PRO_READ_CCSC_OUTPUT_SHORT_BITSHIFT 10
#define INNO4PRO_READ_ISSC_SHORT_BITSHIFT        8
#define INNO4PRO_READ_UVL_TIMER_BITSHIFT         6
#define INNO4PRO_READ_WATCHDOG_TIMER_BITSHIFT    4
#define INNO4PRO_READ_CV_MODE_BITSHIFT           2
#define INNO4PRO_READ_CV_MODE_TIMER_BITSHIFT     0
```

InnoSwitch4-Pro读寄存器位移计数宏

定义读取寄存器值时使用的寄存器移位位置

```
#define INNO4PRO_READ_OV_FAULT_BITSHIFT          14
#define INNO4PRO_READ_UV_FAULT_BITSHIFT          12
#define INNO4PRO_READ_CCSC_OUTPUT_SHORT_BITSHIFT 10
#define INNO4PRO_READ_ISSC_SHORT_BITSHIFT        8
#define INNO4PRO_READ_UVL_TIMER_BITSHIFT         6
#define INNO4PRO_READ_WATCHDOG_TIMER_BITSHIFT    4
#define INNO4PRO_READ_CV_MODE_BITSHIFT           2
#define INNO4PRO_READ_CV_MODE_TIMER_BITSHIFT     0
```

遥测READ7位分配

定义READ7寄存器上的位分配

```
#define READ7_Reg_VBEN                           14
#define READ7_Reg_BLEEDER                        13
#define READ7_Reg_PSUOFF                         12
#define READ7_Reg_FSTVIC                         11
#define READ7_Reg_CVO                            10
#define READ7_Reg_OTP                            9
```

遥测READ10位分配

定义READ10寄存器上的位分配

```
#define READ10_Reg_INTERRUPT_EN                  15
#define READ10_Reg_CONTROL_S                     14
#define READ10_Reg_VDIS                          13
#define READ10_Reg_HIGH_FSW                      12
#define READ10_Reg_OTP                           9
#define READ10_Reg_VOUT2PCT                      5
#define READ10_Reg_VOUT10PCT                     4
#define READ10_Reg_ISSC                          3
#define READ10_Reg_CCSC                          2
#define READ10_Reg_VOUT_UV                       1
#define READ10_Reg_VOUT_OV                       0
```

遥测READ16位分配

定义READ16寄存器上的位分配

```
#define READ16_Reg_ar_CV                         15
#define READ16_Reg_ar_ISSC                       12
#define READ16_Reg_ar_CCSC                       11
#define READ16_Reg_ar_VOUT_OV                    10
#define READ16_Reg_ar_VOUT_UV                    9
#define READ16_Reg_LO                            7
#define READ16_Reg_Lo_CVO                        6
#define READ16_Reg_PSUOFF                        5
#define READ16_Reg_Lo_ISSC                       4
#define READ16_Reg_Lo_VOUT_OV                    2
#define READ16_Reg_Lo_VOUT_UV                    1
#define READ16_Reg_BPS_OV                        0
```

遥测READ17位分配

定义READ17寄存器上的位分配

```
#define READ17_Reg_CONTROL_S_MASK                15
#define READ17_Reg_LO_Fault_MASK                 14
#define READ17_Reg_CVO_MASK                      13
#define READ17_Reg_ISSC_MASK                     12
#define READ17_Reg_CCSC_MASK                     11
#define READ17_Reg_VOUT_UV_MASK                  10
#define READ17_Reg_VOUT_OV_MASK                  9
#define READ17_Reg_OMF                           8
#define READ17_Reg_VBUSSC                        7
#define READ17_Reg_CONTROL_S_STATUS              6
#define READ17_Reg_LO_FAULT_STATUS               5
#define READ17_Reg_CCAR_STATUS                   4
#define READ17_Reg_ISSC_STATUS                   3
#define READ17_Reg_CCSC_STATUS                   2
#define READ17_Reg_VOUT_UV_STATUS                1
#define READ17_Reg_VOUT_OV_STATUS                0
```

InnoSwitch4-Pro计算宏

定义计算单个寄存器所需的常数

```
#define INNO4PRO_RSENSE                          (float)(9)
#define INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE       (float)(32)
#define INNO4PRO_ADC_FULL_RANGE                  (float)(192)
#define INNO4PRO_CC_SET_PT_FACTOR                (float)(6)
#define INNO4PRO_CC_SET_PT_MULT                  (float)
                                  (INNO4PRO_RSENSE *
                                   INNO4PRO_CC_SET_PT_FACTOR)
#define INNO4PRO_CV_SET_PT_MULT                  (float)(100)
#define INNO4PRO_OV_PERCENTAGE_MULT              (float)(1.15)
#define INNO4PRO_UV_PERCENTAGE_MULT              (float)(0.85)
#define INNO4PRO_OV_SET_PT_MULT                  (float)(10)
#define INNO4PRO_UV_SET_PT_MULT                  (float)(10)
#define INNO4PRO_OV_READ_MULT                    (float)(100)
#define INNO4PRO_UV_READ_MULT                    (float)(100)
#define INNO4PRO_CDC_SET_PT_DIV                  (float)(0.02)
#define INNO4PRO_CDC_SET_PT_MULT                 (float)(50)
#define INNO4PRO_VKP_SET_PT_MULT                 (float)(10)
```

InnoSwitch4-Pro最大值和最小值配置设置

定义写入InnoSwitch4-Pro寄存器的最小值和最大值

```
#define INNO4PRO_VBEN_SET_PT_MAX                (3)
#define INNO4PRO_VBEN_SET_PT_MIN                (0)

#define INNO4PRO_RSENSE_MAX_LIMIT               (20)
#define INNO4PRO_RSENSE_MIN_LIMIT               (1)

#define INNO4PRO_CV_MAX_LIMIT                   (24)
#define INNO4PRO_CV_MIN_LIMIT                   (3)

#define INNO4PRO_OV_MAX_LIMIT                   (25)
#define INNO4PRO_OV_MIN_LIMIT                   (3.3)

#define INNO4PRO_UV_MAX_LIMIT                   (24)
#define INNO4PRO_UV_MIN_LIMIT                   (2.7)
#define INNO4PRO_CDC_MAX_LIMIT                  (600)
#define INNO4PRO_CDC_MIN_LIMIT                  (0)

#define INNO4PRO_CC_MAX_LIMIT                   (192)
#define INNO4PRO_CC_MIN_LIMIT                   (25)

#define INNO4PRO_VKP_MAX_LIMIT                  (24)
#define INNO4PRO_VKP_MIN_LIMIT                  (5.3)
```

宏定义

INNO4PRO_ADDRESS **0x18**
InnoSwitch4-Pro I²C 7位从地址

WR_WORD **0x03**
向I²C写函数发出信号以发送2个字节的数据（不包括slaveAddress和registerAddress）

WR_BYTE **0x02**
向I²C写函数发出信号以发送1个字节的数据（不包括slaveAddress和registerAddress）

RD_MSB **1**
向I²C读函数发出信号，以读取接收数据的MSB

RD_LSB **0**
向I²C读函数发出信号，以读取接收数据的LSB

set_bit(address,bit) **(address |= (1<<bit))**
将地址的位设置为1

clear_bit(address,bit) **(address &= ~ (1<<bit))**
将地址的位设置为0

toggle_bit(address,bit) **(address ^= (1<<bit))**
将地址的位从1切换为0或从0切换为1

test_bit(address,bit) **(address & (1<<bit))**
从地址中获取一个位的值

sig_minmax(sig,min,max) **((sig < min)? sig = min: (sig > max)? sig = max:0)**
将sig的值限制在最小值和最大值之间或等于最小值和最大值

sig_max(sig, max) **((sig > max) ? sig = max : 0)**
将sig的值限制为小于或等于最大值

sig_min(sig, min) **((sig < min) ? sig = min : 0)**
将sig的值限制为大于或等于最小值

INNO4PRO_READ_OV_FAULT_BITSHIFT **14**
READ6的过压故障移位计数

INNO4PRO_READ_UV_FAULT_BITSHIFT **12**
READ6的欠压故障移位计数

INNO4PRO_READ_CCSC_OUTPUT_SHORT_BITSHIFT **10**
READ6的CCSC故障移位计数

INNO4PRO_READ_ISSC_SHORT_BITSHIFT **8**
READ6的ISSC故障移位计数

INNO4PRO_READ_UVL_TIMER_BITSHIFT **6**
READ6的UVL计时器移位计数

INNO4PRO_READ_WATCHDOG_TIMER_BITSHIFT **4**
READ6的看门狗计时器移位计数

INNO4PRO_READ_CV_MODE_BITSHIFT **2**
READ6的CV模式移位计数

INNO4PRO_READ_CV_MODE_TIMER_BITSHIFT **0**
READ6的CV模式计时器移位计数

INNO4PRO_VBUS_ENABLE **3**
串联母线开关使能

INNO4PRO_VBUS_DISABLE_NORESET **1**
串联母线开关禁止但不复位

INNO4PRO_VBUS_DISABLE **0**
串联母线开关禁止且复位

INNO4PRO_BLEEDER_ENABLE_AUTODISABLE **3**
具有自动禁止特性的有源泄放使能

INNO4PRO_BLEEDER_ENABLE **1**
有源泄放使能

INNO4PRO_BLEEDER_DISABLE **0**
有源泄放禁止

INNO4PRO_LOAD_DISCHARGE_DISABLE **12**
负载放电禁止

INNO4PRO_LOAD_DISCHARGE_ENABLE **3**
负载放电使能

INNO4PRO_LOAD_DISCHARGE_ENABLE_NORESET **2**
负载放电使能但不复位

INNO4PRO_TURN_OFF_PSU_ENABLE **true**
锁存关断器件使能

INNO4PRO_TURN_OFF_PSU_DISABLE **false**
锁存关断器件禁止

INNO4PRO_FASTVI_UPDATE_LIMIT_DISABLE **true**
快速VI命令禁止

INNO4PRO_FASTVI_UPDATE_LIMIT_ENABLE **false**
快速VI命令使能

INNO4PRO_OVL_FAULT_RESPONSE_NORESPONSE **0**
过压故障响应设置为无响应

INNO4PRO_OVL_FAULT_RESPONSE_LATCHOFF **1**
过压故障响应设置为锁存关断

INNO4PRO_OVL_FAULT_RESPONSE_AUTORESTART **2**
过压故障响应设置为自动重启动

INNO4PRO_OVL_FAULT_RESPONSE_DISABLEOUTPUT **3**
过压故障响应设置为关输出

INNO4PRO_UVL_FAULT_RESPONSE_NORESPONSE **0**
欠压故障响应设置为无响应

INNO4PRO_UVL_FAULT_RESPONSE_LATCHOFF **1**
欠压故障响应设置为锁存关断

INNO4PRO_UVL_FAULT_RESPONSE_AUTORESTART **2**
欠压故障响应设置为自动重启动

INNO4PRO_UVL_FAULT_RESPONSE_DISABLEOUTPUT **3**
欠压故障响应设置为关输出

INNO4PRO_CCSC_FAULT_RESPONSE_NORESPONSE **0**
CCSC故障响应设置为无响应

INNO4PRO_CCSC_FAULT_RESPONSE_LATCHOFF **1**
CCSC故障响应设置为锁存关断

INNO4PRO_CCSC_FAULT_RESPONSE_AUTORESTART **2**
CCSC故障响应设置为自动重启动

INNO4PRO_CCSC_FAULT_RESPONSE_DISABLEOUTPUT **3**
CCSC故障响应设置为关输出

INNO4PRO_ISSC_FAULT_RESPONSE_NORESPONSE **0**
ISSC故障响应设置为无响应

INNO4PRO_ISSC_FAULT_RESPONSE_LATCHOFF **1**
ISSC故障响应设置为锁存关断

INNO4PRO_ISSC_FAULT_RESPONSE_AUTORESTART **2**
ISSC故障响应设置为自动重启动

INNO4PRO_ISSC_FAULT_RESPONSE_DISABLEOUTPUT **3**
ISSC故障响应设置为关输出

INNO4PRO_ISSC_FREQ_THRESHOLD_60KHZ **0**
ISSC频率阈值设置为60kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_30KHZ **1**
ISSC频率阈值设置为30kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_90KHZ **2**
ISSC频率阈值设置为90kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_120KHZ **3**
ISSC频率阈值设置为120kHz

INNO4PRO_ISSC_CC_THRESHOLD_16 **1**
ISSC CC阈值设置为16 LSB

INNO4PRO_ISSC_CC_THRESHOLD_32 **2**
ISSC CC阈值设置为32 LSB

INNO4PRO_ISSC_CC_THRESHOLD_48 **3**
ISSC CC阈值设置为48 LSB

INNO4PRO_ISSC_CC_THRESHOLD_64 **4**
ISSC CC阈值设置为64 LSB

INNO4PRO_ISSC_CC_THRESHOLD_80 **5**
ISSC CC阈值设置为80 LSB

INNO4PRO_ISSC_CC_THRESHOLD_96 **6**
ISSC CC阈值设置为96 LSB

INNO4PRO_ISSC_CC_THRESHOLD_112 **7**
ISSC CC阈值设置为112 LSB

INNO4PRO_UVL_FAULT_TIMER_8MS **0**
欠压故障计时器设置为8ms

INNO4PRO_UVL_FAULT_TIMER_16MS **1**
欠压故障计时器设置为16ms

INNO4PRO_UVL_FAULT_TIMER_32MS **2**
欠压故障计时器设置为32ms

INNO4PRO_UVL_FAULT_TIMER_64MS **3**
欠压故障计时器设置为64ms

INNO4PRO_WATCHDOG_TIMER_NOWATCHDOG **0**
看门狗计时器禁止

INNO4PRO_WATCHDOG_TIMER_500MS **1**
看门狗计时器设置为0.5s

INNO4PRO_WATCHDOG_TIMER_1000MS **2**
看门狗计时器设置为1.0s

INNO4PRO_WATCHDOG_TIMER_2000MS **3**
看门狗计时器设置为2.0s

INNO4PRO_CVOL_FAULT_RESPONSE_NORESPONSE **0**
仅CV模式故障响应设置为无响应

INNO4PRO_CVOL_FAULT_RESPONSE_LATCHOFF **1**
仅CV模式故障响应设置为锁存关断

INNO4PRO_CVOL_FAULT_RESPONSE_AUTORESTART **2**
仅CV模式故障响应设置为自动重启动

INNO4PRO_CVOL_FAULT_RESPONSE_DISABLEOUTPUT **3**
仅CV模式故障响应设置为关输出

INNO4PRO_CVOL_FAULT_TIMER_8MS **0**
仅CV模式故障计时器设置为8ms

INNO4PRO_CVOL_FAULT_TIMER_16MS **1**
仅CV模式故障计时器设置为16ms

INNO4PRO_CVOL_FAULT_TIMER_32MS **2**
仅CV模式故障计时器设置为32ms

INNO4PRO_CVOL_FAULT_TIMER_64MS **3**
仅CV模式故障计时器设置为64ms

INNO4PRO_INTERRUPT_CONTROL_S_MASK **0x40**
控制初级故障的中断掩码

INNO4PRO_INTERRUPT_BPS_CURR_LO_FAULT_MASK **0x20**
BPS电流锁存关断故障的中断掩码

INNO4PRO_INTERRUPT_CVO_PKLOAD_TIMER_MASK **0x10**
CVO模式峰值负载计时器故障的中断掩码

INNO4PRO_INTERRUPT_ISSC_MASK **0x08**
IS引脚短路故障的中断掩码

INNO4PRO_INTERRUPT_CCSC_MASK **0x04**
输出短路故障的中断掩码

INNO4PRO_INTERRUPT_VOUT_UV_MASK **0x02**
欠压故障的中断掩码

INNO4PRO_INTERRUPT_VOUT_OV_MASK **0x01**
过压故障的中断掩码

INNO4PRO_OTP_FAULT_HYST_40DEG **0**
过温故障滞回设置为40摄氏度

INNO4PRO_OTP_FAULT_HYST_60DEG **1**
过温故障滞回设置为60摄氏度

INNO4PRO_CVLOAD_DEFAULT **0x20**
恒压负载默认设置

INNO4PRO_CVLOAD_RECOMMENDED **0x80**
恒压负载推荐设置

INNO4PRO_LOOPSPEED1_DEFAULT **0x281E**
环路速度1默认设置

INNO4PRO_LOOPSPEED1_RECOMMENDED **0x140A**
环路速度1推荐设置

INNO4PRO_LOOPSPEED2_DEFAULT **0x08C8**
环路速度2默认设置

INNO4PRO_LOOPSPEED2_RECOMMENDED **0x1F84**
环路速度2推荐设置

PI Command寄存器地址分配

INNO4PRO_VBEN **0x04**
串联母线开关控制寄存器

INNO4PRO_BLEEDER **0x86**
主动泄放功能寄存器

INNO4PRO_VDIS **0x08**
负载(VBUS)放电寄存器

INNO4PRO_TURN_OFF_PSU **0x8A**
锁存关断器件寄存器

INNO4PRO_FAST_VI_CMD **0x8C**
输出CV/CC的更新速度寄存器

INNO4PRO_CVO **0x0E**
仅恒压、计时器和响应寄存器

INNO4PRO_CV **0x10**
输出电压寄存器

INNO4PRO_OVA **0x92**
过压阈值和响应寄存器

INNO4PRO_UVA **0x94**
欠压阈值、响应和计时器寄存器

INNO4PRO_CDC **0x16**
输出线压降补偿寄存器

INNO4PRO_CC **0x98**
恒流调整寄存器

INNO4PRO_VKP **0x1A**
恒定输出功率拐点电压寄存器

INNO4PRO_CCSC **0x20**
输出短路故障检测寄存器

INNO4PRO_ISSC **0xA2**
IS引脚短路故障响应、电流和频率寄存器

INNO4PRO_WATCHDOG_TIMER **0x26**
通信速率监测寄存器

INNO4PRO_INTERRUPT **0x2C**
中断掩码管理器

INNO4PRO_OTP **0xAE**
次级过温故障滞回寄存器

INNO4PRO_CV_LOAD **0xB0**
恒压负载寄存器

INNO4PRO_LOOP_SPEED_1 **0x32**
环路速度1寄存器

INNO4PRO_LOOP_SPEED_2 **0x34**
环路速度2寄存器

INNO4PRO_VBUSSC **0xB6**
串联母线开关短路故障

INNO4PRO_DCM **0xBA**
仅断续模式(DCM)寄存器

INNO4PRO_SRZVS **0x3E**
SRFET零电压开关寄存器

遥测寄存器地址分配

INNO4PRO_READ0 **0x00**
版本ID遥测寄存器

INNO4PRO_READ1 **0x02**
输出电压设置点遥测寄存器

INNO4PRO_READ2 **0x04**
输出电流设置点遥测寄存器

INNO4PRO_READ3 **0x06**
过压阈值和响应遥测寄存器

INNO4PRO_READ4 **0x08**
欠压阈值、响应和计时器遥测寄存器

INNO4PRO_READ5 **0x0A**
恒流和恒功率遥测寄存器

INNO4PRO_READ6 **0x0C**
OVL、UVL、CCSC、ISSC、UVLTIMER、WDTIMER、CVMODE和CVTimer故障和设置遥测寄存器

INNO4PRO_READ7 **0x0E**
VBUS开关使能、最小负载、电源关断、快速VI、CV模式、OTP滞回和CDC遥测寄存器

INNO4PRO_READ8 **0x10**
实测输出电流遥测寄存器

INNO4PRO_READ9 **0x12**
实测输出电压遥测寄存器

INNO4PRO_READ10 **0x14**
INTERRUPT、CONTROL_S、VDIS、HIGH_FSW、OTP、VOUT2PCT、VOUT10PCT、ISSC、CCSC、VOUT_UV和VOUT_OV遥测寄存器

INNO4PRO_READ11 **0x16**
OMF标志

INNO4PRO_READ12 **0x18**
平均输出电流遥测寄存器

INNO4PRO_READ13 **0x1A**
平均输出电压遥测寄存器

INNO4PRO_READ14 **0x1C**
电压DAC遥测寄存器

INNO4PRO_READ16 **0x20**
AR_CVO、AR_ISSC、CCSC、VOUT_OV、VOUT_UV、LO、LO_CVO、PSU_OFF、LO_ISSC、LO_CCSC、LO_VOUT_OV、LO_VOUT_UV和BPS_OV遥测寄存器

INNO4PRO_READ17 **0x22**
中断遥测寄存器

INNO4PRO_READ_LOOP_SPEED_1 **0x26**
环路速度1遥测寄存器

INNO4PRO_READ_LOOP_SPEED_2 **0x28**
环路速度2遥测寄存器

遥测READ7位分配

READ7_Reg_VBEN **14**
VBUS开关位

READ7_Reg_BLEEDER **13**
最小负载位

READ7_Reg_PSUOFF **12**
关断电源位

READ7_Reg_FSTVIC **11**
快速VI命令位

READ7_Reg_CVO **10**
仅恒压模式位

READ7_Reg_OTP **9**
过温保护滞回位

遥测READ10位分配

READ10_Reg_INTERRUPT_EN **15**
中断使能位

READ10_Reg_CONTROL_S **14**
系统就绪信号位

READ10_Reg_VDIS **13**
输出放电位

READ10_Reg_HIGH_FSW **12**
高开关频率位

READ10_Reg_OTP **9**
过温保护位

READ10_Reg_VOUT2PCT **5**
2% BLEEDER使能位

READ10_Reg_VOUT10PCT **4**
10% BLEEDER使能位

READ10_Reg_ISSC **3**
检测到IS引脚短路位

READ10_Reg_CCSC **2**
检测到输出短路位

READ10_Reg_VOUT_UV **1**
输出电压UV故障位

READ10_Reg_VOUT_OV **0**
输出电压OV故障位

遥测READ16位分配

READ16_Reg_ar_CV **15**
CVO模式自动重启动位

READ16_Reg_ar_ISSC **12**
ISSC自动重启动位

READ16_Reg_ar_CCSC **11**
CCSC自动重启动位

READ16_Reg_ar_VOUT_OV **10**
输出电压OV自动重启动

READ16_Reg_ar_VOUT_UV **9**
输出电压UV自动重启动

READ16_Reg_LO **7**
发生锁存关断位

READ16_Reg_Lo_CVO **6**
CVO模式自动重启动位

READ16_Reg_PSUOFF **5**
收到电源关断命令位

READ16_Reg_Lo_ISSC **4**
ISSC锁存关断位

READ16_Reg_Lo_VOUT_OV **2**
输出电压OV锁存关断位

READ16_Reg_Lo_VOUT_UV **1**
输出电压UV锁存关断位

READ16_Reg_BPS_OV **0**
BPS引脚锁存关断位

遥测READ17位分配

READ17_Reg_CONTROL_S_MASK **15**
Control_S中断掩码

READ17_Reg_LO_Fault_MASK **14**
锁存关断中断掩码

READ17_Reg_CVO_MASK **13**
CVO中断掩码

READ17_Reg_ISSC_MASK **12**
ISSC中断掩码

READ17_Reg_CCSC_MASK **11**
CCSC中断掩码

READ17_Reg_VOUT_UV_MASK **10**
输出电压UV中断掩码

READ17_Reg_VOUT_OV_MASK **9**
输出电压OV中断掩码

READ17_Reg_OMF **8**
OMF标志中断掩码

READ17_Reg_VBUSSC **7**
VBUS引脚短路掩码

READ17_Reg_CONTROL_S_STATUS **6**
Control_S中断状态

READ17_Reg_LO_FAULT_STATUS **5**
锁存关断中断状态

READ17_Reg_CVO_STATUS **4**
CVO中断状态

READ17_Reg_ISSC_STATUS **3**
ISSC中断状态

READ17_Reg_CCSC_STATUS **2**
CCSC中断状态

READ17_Reg_VOUT_UV_STATUS **1**
输出电压UV中断状态

READ17_Reg_VOUT_OV_STATUS **0**
输出电压OV中断状态

InnoSwitch4-Pro计算宏

INNO4PRO_RSENSE **(float)(9)**
电流检测电阻(mΩ)

INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE **(float)(32)**
电流检测电阻全范围电压(mV)

INNO4PRO_ADC_FULL_RANGE **(float)(192)**
模数转换器全范围

INNO4PRO_CC_SET_PT_FACTOR **(float)(6)**
CC设置点因数：(192 / 32)

INNO4PRO_CC_SET_PT_MULT **(float)**
(INNO4PRO_RSENSE * INNO4PRO_CC_SET_POINT_FACTOR)
CC设置点计算：（ADC电压 * Rsense * 192 / 32）

INNO4PRO_CV_SET_PT_MULT **(float)(100)**
输出电压步长为10mV/LSB的乘数

INNO4PRO_OV_PERCENTAGE_MULT **(float)(1.15)**
OV乘数设置为CV的115%

INNO4PRO_UV_PERCENTAGE_MULT **(float)(0.85)**
UV乘数设置为CV的85%

INNO4PRO_OV_SET_PT_MULT **(float)(10)**
过压写入步长为100mV/LSB的乘数

INNO4PRO_UV_SET_PT_MULT **(float)(10)**
欠压写入步长为100mV/LSB的乘数

INNO4PRO_OV_READ_MULT **(float)(100)**
过压读取步长为100mV/LSB的乘数

INNO4PRO_UV_READ_MULT **(float)(100)**
欠压读取步长为100mV/LSB的乘数

INNO4PRO_CDC_SET_PT_DIV **(float)(0.02)**
输出线压降补偿步长为50mV/LSB的除数

INNO4PRO_CDC_SET_PT_MULT **(float)(50)**
输出线压降补偿步长为50mV/LSB的乘数

INNO4PRO_VKP_SET_PT_MULT **(float)(10)**
恒定输出功率拐点电压步长为100mV/LSB的乘数

InnoSwitch4-Pro最大和最小限制

INNO4PRO_CV_SET_PT_MAX **(2400)**
10mV/LSB时的最大输出电压设置点

INNO4PRO_CV_SET_PT_MIN **(300)**
10mV/LSB时的最小输出电压设置点

INNO4PRO_OV_SET_PT_MAX **(250)**
100mV/LSB时的最大过压设置点

INNO4PRO_OV_SET_PT_MIN **(33)**
100mV/LSB时的最小过压设置点

INNO4PRO_UV_SET_PT_MAX **(240)**
100mV/LSB时的最大欠压设置点

INNO4PRO_UV_SET_PT_MIN **(27)**
100mV/LSB时的最小欠压设置点

INNO4PRO_CDC_SET_PT_MAX **(12)**
最大输出线压降补偿设置点

INNO4PRO_CDC_SET_PT_MIN **(0)**
最小输出线压降补偿设置点

INNO4PRO_CC_SET_PT_MAX **(192)**
最大恒流设置点

INNO4PRO_CC_SET_PT_MIN **(25)**
最小恒流设置点

INNO4PRO_VKP_SET_PT_MAX **(240)**
100mV/LSB时的最大恒定输出功率拐点电压设置点

INNO4PRO_VKP_SET_PT_MIN **(53)**
100mV/LSB时的最小恒定输出功率拐点电压设置点

INNO4PRO_VBEN_SET_PT_MAX **(3)**
最大VBEN设置点

INNO4PRO_VBEN_SET_PT_MIN **(0)**
最小VBEN设置点

INNO4PRO_RSENSE_MAX_LIMIT **(20)**
最大Rsense值(mΩ)

INNO4PRO_RSENSE_MIN_LIMIT **(1)**
最小Rsense值(mΩ)

INNO4PRO_CV_MAX_LIMIT **(24)**
最大输出电压(V)

INNO4PRO_CV_MIN_LIMIT **(3)**
最小输出电压(V)

INNO4PRO_OV_MAX_LIMIT **(25)**
最大过压设置点(V)

INNO4PRO_OV_MIN_LIMIT **(3.3)**
最小过压设置点(V)

INNO4PRO_UV_MAX_LIMIT **(24)**
最大欠压设置点(V)

INNO4PRO_UV_MIN_LIMIT **(2.7)**
最小欠压设置点(V)

INNO4PRO_CDC_MAX_LIMIT **(600)**
最大输出线压降补偿设置点(mV)

INNO4PRO_CDC_MIN_LIMIT **(0)**
最小输出线压降补偿设置点(mV)

INNO4PRO_CC_MAX_LIMIT **(192)**
最大恒流限值

INNO4PRO_CC_MIN_LIMIT **(25)**
最小恒流限值

INNO4PRO_VKP_MAX_LIMIT **(24)**
最大恒定输出功率拐点电压(V)

INNO4PRO_VKP_MIN_LIMIT **(5.3)**
最小恒定输出功率拐点电压(V)

I2C Drivers（I2C驱动程序）

该源文件包含I^2C的驱动程序API

I^2C函数

int I2C_Write16 (uint16_t slaveAddress, uint8_t dataAddress, uint8_t *dataBuffer, uint8_t buflen)
使用提供的参数处理一个I^2C主控写事务

uint16_t I2C_Read16 (uint16_t slaveAddress, uint8_t dataAddress)
使用提供的参数处理一个I^2C主控读事务

uint8_t I2C_Read8 (uint16_t slaveAddress, uint8_t dataAddress)
使用提供的参数处理一个I^2C主控读事务

函数文档

int I2C_Write16 (uint16_t slaveAddress, uint8_t dataAddress, uint8_t *dataBuffer, uint8_t buflen)
使用提供的参数处理一个I²C主控写事务。这会将1到2个字节的数据写入从器件

参数：
- slaveAddress：要访问的I²C器件的地址。使用7位地址方案。
- dataAddress：要访问的寄存器地址
- dataBuffer：指向要发送的数据块的指针
- buflen：要发送的数据块的长度

返回值
- 无

uint16_t I2C_Read16 (uint16_t slaveAddress, uint8_t dataAddress)
使用提供的参数处理一个I²C主控读事务。这会从从器件读取2个字节的数据

参数：
- slaveAddress：要访问的I²C器件的地址。使用7位地址方案。
- dataAddress：要访问的寄存器地址

返回值
- 来自从器件的合并LSB和MSB

uint8_t I2C_Read8 (uint16_t slaveAddress, uint8_t dataAddress)
使用提供的参数处理一个I²C主控读事务。这会从从器件读取2个字节的数据

参数：
- slaveAddress：要访问的I²C器件的地址。使用7位地址方案。
- dataAddress：要访问的寄存器地址

返回值
- 来自从器件的1个字节

Clock Driver（时钟驱动程序）
这是包含时钟信号驱动程序API的源文件。

函数

void clock_TimeUpdate (void)
用于计算

uint16_t clock_GetElapsedTimeUs (uint16_t u16TimeStamp)
用于计算以微秒为单位的经过时间

uint16_t clock_GetElapsedTimeMs (uint16_t u16TimeStamp)
用于计算以毫秒为单位的经过时间

uint16_t clock_GetElapsedTimeSec (uint16_t u16TimeStamp)
用于计算以秒为单位的经过时间

uint16_t clock_GetTimeStampUs (void)
以微秒为单位获取当前时间戳

uint16_t clock_GetTimeStampMs (void)
以毫秒为单位获取当前时间戳

uint16_t clock_GetTimeStampSec (void)
以秒为单位获取当前时间戳

bool clock_HasTimeElapsedUs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成微秒级延迟

bool clock_HasTimeElapsedMs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成毫秒级延迟

bool clock_HasTimeElapsedSec (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成秒级延迟

函数文档

void clock_TimeUpdate (void)
这是一个简单的时钟接口，必须在中断上运行以生成时钟信号（毫秒、秒），用于根据系统时钟创建延迟

注：
- 该函数必须在200 µs中断上运行

参数：
- 无

返回值
- 无

uint16_t clock_GetElapsedTimeUs (uint16_t u16TimeStamp)
用于计算以微秒为单位的经过时间

参数：
- u16TimeStamp：微秒时间戳变量

返回值
- 返回自u16TimeStamp以来经过的时间

uint16_t clock_GetElapsedTimeMs (uint16_t u16TimeStamp)
用于计算以毫秒为单位的经过时间

参数：
- u16Timestamp：毫秒时间戳变量

返回值
- 返回自u16Timestamp以来经过的时间

uint16_t clock_GetElapsedTimeSec (uint16_t u16TimeStamp)
用于计算以秒为单位的经过时间

参数：
- u16Timestamp：秒时间戳变量

返回值
- 返回自u16Timestamp以来经过的时间

uint16_t clock_GetTimeStampUs (void)
以微秒为单位获取当前时间戳

参数：
- 无

返回值
- 当前时间戳（以µs为单位）

uint16_t clock_GetTimeStampMs (void)
以毫秒为单位获取当前时间戳

参数：
- 无

返回值
- 当前时间戳（以ms为单位）

uint16_t clock_GetTimeStampSec (void)
以秒为单位获取当前时间戳

参数：
- 无

返回值
- 当前时间戳（以s为单位）

bool clock_HasTimeElapsedUs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成微秒级延迟

参数：
- u16TimeStamp：微秒时间戳变量
- u16TimeDurationCheck：与u16TimeStamp进行比较的持续时间

返回值
- 延迟时间结束后为1

bool clock_HasTimeElapsedMs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成毫秒级延迟

参数：
- u16TimeStamp：毫秒时间戳变量
- u16TimeDurationCheck：与u16TimeStamp进行比较的持续时间

返回值
- 延迟时间结束后为1

bool clock_HasTimeElapsedSec (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
用于生成秒级延迟

参数：
- u16TimeStamp：秒时间戳变量
- u16TimeDurationCheck：与u16TimeStamp进行比较的持续时间

返回值
- 延迟时间结束后为1

InnoSwitch4-Pro

函数

Setter函数
用于设置寄存器变量值的函数

void InnoProBase_Set_Register_CV (float fVout)

void InnoProBase_Set_Register_OVA (float fOva)

void InnoProBase_Set_Register_UVA (float fUva)

void InnoProBase_Set_Register_CC (float fCc)

void InnoProBase_Set_Register_CDC (float fCdc)

void InnoProBase_Set_Register_VKP (float fVkp)

void InnoProBase_Set_Register_VBEN (float fVben)

void InnoProBase_Set_Register_UVL (float fUvl)

void InnoProBase_Set_Rsense_Value (float fRsense)

Getter函数
用于获取寄存器变量内容的函数

float InnoProBase_Get_Register_CV (void)

float InnoProBase_Get_Register_OVA (void)

float InnoProBase_Get_Register_UVA (void)

float InnoProBase_Get_Register_CC (void)

float InnoProBase_Get_Register_CDC (void)

float InnoProBase_Get_Register_VKP (void)

float InnoProBase_Get_Register_VBEN (void)

float InnoProBase_Get_Register_UVL (void)

float InnoProBase_Get_Rsense_Value (void)

计算函数
特定寄存器的阈值计算和调整范围

float Inno4Pro_Compute_CV (float fSetPt)
InnoSwitch4-Pro输出电压(CV)计算

float Inno4Pro_Compute_OV (float fSetPt)
InnoSwitch4-Pro过压阈值(OVA)计算

float Inno4Pro_Compute_UV (float fSetPt)
InnoSwitch4-Pro欠压阈值(UVA)计算

float Inno4Pro_Compute_CDC (float fSetPt)
InnoSwitch4-Pro输出线压降补偿(CDC)计算

float Inno4Pro_Compute_CC (float fSetPt)
InnoSwitch4-Pro恒流调整(CC)计算

float Inno4Pro_Compute_VKP (float fSetPt)
InnoSwitch4-Pro输出功率拐点电压(VKP)计算

float Inno4Pro_Compute_VBEN (float fSetPt)
InnoSwitch4-Pro串联母线开关控制(VBEN)的计算

缓冲区相关函数
缓冲区和奇偶校验处理

void InnoProBase_Encode_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
处理输入数据到十六进制LSB和MSB（无奇偶校验位）的转换。

bool InnoProBase _OddParity (uint8_t u8OddParity)
处理奇校验位检测。

void InnoProBase _Format_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
处理输入数据到十六进制LSB和MSB（有奇偶校验位）的转换。

bool InnoProBase _AddOddParity (uint16_t u16Temp)
为LSB添加奇偶校验位

void Inno4Pro_Process_Volt_Buffers (void)
处理在InnoSwitch4-Pro电压相关寄存器上写入值的准备工作。

请求检测函数
存储寄存器变量的先前值

bool InnoProBase_Detect_Voltage_Request (void)
检查是否有新的电压请求

bool InnoProBase_Detect_Current_Request (void)
检查是否有新的电流请求

API写函数
用于控制InnoSwitch4-Pro的应用程序编程接口

void Inno4Pro_Initialization (void)
处理所有要写入InnoSwitch 4-Pro作为初始化的常见I2C配置

void Inno4Pro_Vbus_Switch_Control (bool bEnableVben)
Vbus开关控制（VBEN控制）

void Inno4Pro_Vbus_Switch_Control_NoReset (uint8_t u8EnableVben)
Vbus开关控制但不复位

void Inno4Pro_Bleeder_Enable (bool bEnable)
处理泄放设置

void Inno4Pro_Load_Discharge (bool bEnable)
激活Vbus负载放电

void Inno4Pro_TurnOff_PSU (bool bEnable)
关断电源

void Inno4Pro_FastVI_Disable (bool bDisable)
设置CV和CC命令速度限值

void Inno4Pro_CVOnlyMode_Enable (bool bEnable, uint16_t u16Response, uint16_t u16Timer)
设置仅恒压模式

void Inno4Pro_Write_Volts (float fSetPtCV)
无泄放控制的输出电压控制

void Inno4Pro_Write_Over_Volts (float fSetPtOVA, uint16_t u16OVL)
写入过压保护设置

void Inno4Pro_Write_Under_Volts (float fSetPtUVA, uint16_t u16Uv_FaultResp, uint16_t u16Uv_timer)
写入欠压保护设置

void Inno4Pro_Write_Cable_Drop_Comp (float fSetPtCDC)
写入输出线压降补偿(CDC)设置

void Inno4Pro_Write_Amps (float fSetPtCC)
恒流(CC)控制

void Inno4Pro_Write_Volt_Peak (float fSetPtVpk)
恒定输出功率电压阈值(VKP)控制

void Inno4Pro_Write_CCSC_Fault_Response (uint16_t u16Response)
写入输出短路故障响应设置

void Inno4Pro_Write_ISSC_Fault_Response (uint16_t u16Response, uint16_t u16Frequency, uint16_t u16CC)
写入IS引脚短路故障响应设置

void Inno4Pro_Write_Watchdog_Timer (uint16_t u16Timer)
写入看门狗计时器设置

void Inno4Pro_Write_Interupt_Mask (uint16_t u16IntMask)
写入中断掩码设置

void Inno4Pro_Write_OTP_Hysteresis (uint16_t u16Otp)
设置过温滞回

void Inno4Pro_Write_CV_Load (uint16_t u16Load)
写入恒压负载设置

void Inno4Pro_Write_Loop_Speed1 (uint16_t u16LoopSpeed)
写入环路速度1设置

void Inno4Pro_Write_Loop_Speed2 (uint16_t u16LoopSpeed)
写入环路速度2设置

bool Inno4Pro_Write_VI (float fSetPtCV, float fSetPtCC)
具有泄放控制和恒流(CC)控制的输出电压控制

void Inno4Pro_VBUSSC (uint16_t u16VSSC_response, uint16_t u16Vsamples, uint16_t u16CCThreshold)
定义器件对串联母线开关短路故障的响应方式。

void Inno4Pro_DCMonly (bool bEnable)
使能或禁止限制次级到初级开关周期请求的功能，使变换器始终工作于断续导通模式

bool Inno4Pro_Process_Voltage (bool bVoltIncrease)
处理电压递增/递减的命令序列

常见API遥测函数
这些函数是主要API读函数的基础

uint16_t InnoProBase_Telemetry (uint8_t ReadBack_Address)
处理InnoSwitch4-Pro的常见I2C读回遥测

bool InnoProBase_Read_Bit (uint8_t ReadBack_Address, uint8_t Bit)
处理InnoSwitch4-Pro的I2C读位

uint8_t InnoProBase_Read_Byte (uint8_t ReadBack_Address, bool bHighByte)

处理InnoSwitch4-Pro的I2C读字节

uint8_t InnoProBase_Read_2Bits (uint8_t ReadBack_Address, uint8_t u8ShiftCnt)
处理InnoSwitch4-Pro的I2C读2个位

float InnoProBase_Read_SetPoint (uint16_t ReadBack_Address, float fMultiplier)
处理InnoSwitch4-Pro的I2C设置点和阈值

Read1 - 主要API遥测函数
Read1的遥测API

float Inno4Pro_Read_CV_SetPoint (void)
读取遥测寄存器READ1 - 输出电压设置点

Read2 - 主要API遥测函数
Read2的遥测API

float Inno4Pro_Read_Output_CC_SetPoint (void)
读取遥测寄存器READ2 – 输出电流设置点

Read3 - 主要API遥测函数
Read3的遥测API

float Inno4Pro_Read_OV_Threshold (void)
读取遥测寄存器READ3 - 过压阈值

Read4 - 主要API遥测函数
READ4的遥测API

float Inno4Pro_Read_UV_Threshold (void)
读取遥测寄存器READ4 - 欠压阈值

Read5 - 主要API遥测函数
Read4的遥测API

float Inno4Pro_Read_CC_SetPoint (void)
读取遥测寄存器READ5 – 恒流设置点

float Inno4Pro_Read_CP_Threshold (void)
读取遥测寄存器READ5 – 恒功率设置点

Read6 - 主要API遥测函数
Read6的遥测API

uint8_t Inno4Pro_Read_OV_Fault_Response (void)
读取遥测寄存器READ6 - 过压故障响应

uint8_t Inno4Pro_Read_UV_Fault_Response (void)
读取遥测寄存器READ6 - 欠压阈值故障响应

uint8_t Inno4Pro_Read_OutputSckt_Fault_Response (void)
读取遥测寄存器READ6 - 输出短路故障响应

uint8_t Inno4Pro_Read_IsPinShort_Fault_Response (void)
读取遥测寄存器READ6 - IS引脚短路故障响应

uint8_t Inno4Pro_Read_UV_Fault_Timer (void)
读取遥测寄存器READ6 - 欠压计时器

uint8_t Inno4Pro_Read_Watchdog_Timer (void)
读取遥测寄存器READ6 - 看门狗计时器

uint8_t Inno4Pro_Read_CvMode_Fault_Response (void)
读取遥测寄存器READ6 -恒压模式故障响应

uint8_t Inno4Pro_Read_CvMode_Timer (void)
读取遥测寄存器READ6 -恒压模式计时器

Read7 - 主要API遥测函数
Read7的遥测API

bool Inno4Pro_Read_VbusSwitch (void)
读取遥测寄存器READ7上的第14位 - VBUS开关使能

bool Inno4Pro_Read_Bleeder (void)
读取遥测寄存器READ7上的第13位 - 最小负载（泄放）

bool Inno4Pro_Read_PsuOff (void)
读取遥测寄存器READ7上的第12位 - 关断电源（器件锁存）

bool Inno4Pro_Read_FastVI (void)
读取遥测寄存器READ7上的第11位 - 快速VI命令

bool Inno4Pro_Read_CvoMode (void)
读取遥测寄存器READ7上的第10位 - 仅恒压模式

bool Inno4Pro_Read_OtpFaultHyst (void)
读取遥测寄存器READ7上的第9位 - 过温保护

float Inno4Pro_Read_Cable_Drop_Comp (void)
读取遥测寄存器READ7 -输出线压降补偿

Read8 - 主要API遥测函数
Read8的遥测API

float Inno3Pro_Read_Amps (void)
读取遥测寄存器READ8 - 实测输出电流

Read9 - 主要API遥测函数
Read9的遥测API

float Inno3Pro_Read_Volts (void)
读取遥测寄存器READ9 -实测输出电压

Read10 - 主要API遥测函数
Read10的遥测API

bool Inno4Pro_Read_Status_InterruptEnable (void)
读取遥测寄存器READ10上的第15位 - 中断使能

bool Inno4Pro_Read_Status_SystemReady (void)
读取遥测寄存器READ10上的第14位 - 系统就绪信号

bool Inno4Pro_Read_Status_OutputDischarge (void)
读取遥测寄存器READ10上的第13位 - 输出放电

bool Inno4Pro_Read_Status_HighSwitchFreq (void)
读取遥测寄存器READ10上的第12位 - 开关频率高

bool Inno4Pro_Read_Status_OtpFault (void)
读取遥测寄存器READ10上的第9位 - 过温保护

bool Inno4Pro_Read_Status_Vout2pct (void)
读取遥测寄存器READ10上的第5位 - 2%泄放使能

bool Inno4Pro_Read_Status_Vout10pct (void)
读取遥测寄存器READ10上的第4位 - VOUTADC > 1.1*Vout

bool Inno4Pro_Read_Status_IsPinShort (void)
读取遥测寄存器READ10上的第3位 - 检测到IS引脚短路

bool Inno4Pro_Read_Status_OutputShorCkt (void)
读取遥测寄存器READ10上的第2位 – 检测到输出短路

bool Inno4Pro_Read_Status_UV_Fault (void)
读取遥测寄存器READ10上的第1位 - 输出电压UV故障比较器

bool Inno4Pro_Read_Status_OV_Fault (void)
读取遥测寄存器READ10上的第0位 - 输出电压OV故障比较器

Read12 - 主要API遥测函数
Read12的遥测API

float Inno4Pro_Read_AmpsAverage (void)
读取遥测寄存器READ12 - 平均输出电流

Read13 - 主要API遥测函数
Read14的遥测API

float Inno4Pro_Read_VoltsAverage (void)
读取遥测寄存器READ13 - 平均输出电压

Read14 - 主要API遥测函数
Read14的遥测API

float Inno3Pro_Read_Voltage_DAC (void)
读取遥测寄存器READ14 - 电压DAC

Read16 - 主要API遥测函数
Read16的遥测API

bool Inno4Pro_Read_Status_CvoMode_AR(void)
读取遥测寄存器READ16上的第15位 - CVO模式自动重启动(AR)

bool Inno4Pro_Read_Status_IsPinShort_AR(void)
读取遥测寄存器READ16上的第12位 – 由于IS引脚短路而出现自动重启动指示

bool Inno4Pro_Read_Status_OutputShortCkt_AR(void)
读取遥测寄存器READ16上的第11位 – 由于输出短路而出现自动重启动指示

bool Inno4Pro_Read_Status_OV_AR(void)
读取遥测寄存器READ16上的第10位 - 由于过压故障而出现自动重启动指示

bool Inno4Pro_Read_Status_UV_AR(void)
读取遥测寄存器READ16上的第9位 - 由于欠压过重而出现自动重启动指示

bool Inno4Pro_Read_Status_LatchOff(void)
读取遥测寄存器READ16上的第7位 – - 发生锁存关断(LO)

bool Inno4Pro_Read_Status_CvoMode_LO(void)
读取遥测寄存器READ16上的第6位 – CVO模式锁存关断(LO)

bool Inno4Pro_Read_Status_PsuOffCmd(void)
读取遥测寄存器READ16上的第5位 – 收到电源关断命令

bool Inno4Pro_Read_Status_IsPinShort_LO(void)
读取遥测寄存器READ16上的第4位 – 由于IS引脚短路而出现锁存关断指示

bool Inno4Pro_Read_Status_OV_LO(void)
读取遥测寄存器READ16上的第2位 - 由于过压故障而出现锁存关断指示

bool Inno4Pro_Read_Status_UV_LO(void)
读取遥测寄存器READ16上的第1位 - 由于欠压过重而出现锁存关断指示

Read17 - 主要API遥测函数
Read17的遥测API

bool Inno4Pro_Read_Interrupt_Mask_CntrlSecondary (void)
读取遥测寄存器READ17上的第14位 - 中断掩码控制次级

bool Inno4Pro_Read_Interrupt_Mask_BpsCurrentLo (void)
读取遥测寄存器READ17上的第13位 -中断掩码BPS电流锁存关断

bool Inno4Pro_Read_Interrupt_Mask_CvoPkLoadTimer (void)
读取遥测寄存器READ17上的第12位 -中断掩码BPS电流锁存关断

bool Inno4Pro_Read_Interrupt_Mask_IsPinShort (void)
读取遥测寄存器READ17上的第11位 - 中断掩码IS引脚短路

bool Inno4Pro_Read_Interrupt_Mask_OutputShortCkt (void)
读取遥测寄存器READ17上的第10位 - 中断掩码输出短路

bool Inno4Pro_Read_Interrupt_Mask_UV (void)
读取遥测寄存器READ17上的第9位 -中断掩码Vout欠压(UV)

bool Inno4Pro_Read_Interrupt_Mask_OV (void)
读取遥测寄存器READ17上的第8位 - 中断掩码Vout过压(OV)

bool Inno4Pro_Read_Interrupt_Stat_OMF(void)
读取遥测寄存器READ17上的第7位 – 中断状态工作模式已更改

bool Inno4Pro_Read_Interrupt_Stat_VBUSSC (void)
读取遥测寄存器READ17上的第7位 – 中断状态Vbus短路

bool Inno4Pro_Read_Interrupt_Stat_CntrlSecondary (void)
读取遥测寄存器READ17上的第6位 - 中断状态控制次级

bool Inno4Pro_Read_Interrupt_Stat_BpsCurrentLo (void)
读取遥测寄存器READ17上的第6位 - 控制次级的中断状态

bool Inno4Pro_Read_Interrupt_Stat_CvoPkLoadTimer (void)
读取遥测寄存器READ17上的第5位 - CVO模式峰值负载计时器的中断状态

bool Inno4Pro_Read_Interrupt_Stat_IsPinShort (void)
读取遥测寄存器READ17上的第3位 - 状态IS引脚短路的中断状态

bool Inno4Pro_Read_Interrupt_Stat_OutputShortCkt(void)
读取遥测寄存器READ17上的第2位 - 状态输出短路的中断状态

bool Inno4Pro_Read_Interrupt_Stat_UV (void)
读取遥测寄存器READ17上的第1位 - 状态Vout(UV)的中断状态

bool Inno4Pro_Read_Interrupt_Stat_OV (void)
读取遥测寄存器READ17上的第0位 - 状态Vout(OV)的中断状态

变量

局部变量

- static volatile float **fInno4Pro_CV** = (float)(5)
- static volatile float **fInno4Pro_OVA** = (float)(6.2)
- static volatile float **fInno4Pro_UVA** = (float)(3)
- static volatile float **fInno4Pro_CDC** = (float)(300)
- static volatile float **fInno4Pro_CC** = (float)(5.1)
- static volatile float **fInno4Pro_VKP** = (float)(7)
- static volatile float **fInno4Pro_VBEN** = (float)(0)
- static volatile float **fInno4Pro_UVL** = (float)(0)

InnoSwitch4-Pro标志

用于应用特定例程的示例标志。用于特定的InnoSwitch4-Pro例程

- bool **b_Lock_Timer_Is_Running** = false
- bool **b_Lock_Enable** = false
- bool **b_Setting_Update** = false
- bool **b_Request_Enable** = false
- volatile bool **b_Volt_Setting** = false

InnoSwitch4-Pro校准变量

设置计算所需的变量

- float **fInno4Pro_Rsense** = (float)(5.25)

InnoSwitch4-Pro I2C寄存器缓冲区

用于I2C通信的单个数组缓冲区分别对应一个InnoSwitch4-Pro I2C寄存器

这些数组缓冲区需要填充LSB和MSB值。这些直接用于通过I2C向InnoSwitch4-Pro写入值。这些值使用InnoSwitch4-Pro默认值进行初始化。

Buffer[0] - **LSB**
Buffer[1] - **MSB**

- volatile uint8_t **u8_Buffer_VBEN** [2] ={0}
- volatile uint8_t **u8_Buffer_BLEEDER** [2] ={0}
- volatile uint8_t **u8_Buffer_VDIS** [2] ={0}
- volatile uint8_t **u8_Buffer_TURN_OFF_PSU** [2] ={0}
- volatile uint8_t **u8_Buffer_FAST_VI_CMD** [2] ={0}
- volatile uint8_t **u8_Buffer_CVO** [2] ={0}
- volatile uint8_t **u8_Buffer_CV** [2] ={0}
- volatile uint8_t **u8_Buffer_OVA** [2] ={0}
- volatile uint8_t **u8_Buffer_UVA** [2] ={0}
- volatile uint8_t **u8_Buffer_CDC** [2] ={0}
- volatile uint8_t **u8_Buffer_CCSC** [2] ={0}
- volatile uint8_t **u8_Buffer_CC** [2] ={0}
- volatile uint8_t **u8_Buffer_VKP** [2] ={0}
- volatile uint8_t **u8_Buffer_OVL** [2] ={0}
- volatile uint8_t **u8_Buffer_UVL** [2] ={0}
- volatile uint8_t **u8_Buffer_CCSC** [2] ={0}
- volatile uint8_t **u8_Buffer_ISSC** [2] ={0}
- volatile uint8_t **u8_Buffer_UVL_TIMER** [2] ={0}
- volatile uint8_t **u8_Buffer_WATCHDOG_TIMER** [2] ={0}
- volatile uint8_t **u8_Buffer_CVOL** [2] ={0}
- volatile uint8_t **u8_Buffer_CVOL_TIMER** [2] ={0}
- volatile uint8_t **u8_Buffer_INTERRUPT** [2] ={0}
- volatile uint8_t **u8_Buffer_OTP** [2] ={0}
- volatile uint8_t **u8_Buffer_CVLOAD** [2] ={0}
- volatile uint8_t **u8_Buffer_LoopSpeed1** [2] ={0}
- volatile uint8_t **u8_Buffer_LoopSpeed2** [2] ={0}

函数文档

void InnoProBase_Set_Register_CV (float fVout)
设置CV寄存器

参数：
- fVout：以伏特(V)为单位的变量

返回值
- 无

void InnoProBase_Set_Register_OVA (float fOva)
设置过压阈值

参数：
- fOva：以伏特(V)为单位的变量

返回值
- 无

void InnoProBase_Set_Register_UVA (float fUva)
设置欠压阈值

参数：
- fUva：以伏特(V)为单位的变量

返回值
- 无

void InnoProBase_Set_Register_CC (float fCc)
设置恒流

参数：
- fCc：以安培(A)为单位的变量

返回值
- 无

void InnoProBase_Set_Register_CDC (float fCdc)
设置输出线压降补偿

参数：
- fCdc：以mV为单位的变量

返回值
- 无

void InnoProBase_Set_Register_VKP (float fVkp)
设置恒定输出功率拐点电压

参数：
- fCdc：以mV为单位的变量

返回值
- 无

void InnoProBase_Set_Register_VBEN (float fVben)
设置Vbus控制设置

参数：
- fVben：
  - 3 – 使能VBEN/禁止VDIS
  - 1 – 禁止VBEN/不复位
  - 0 – 禁止VBEN/复位

返回值
- 无

void InnoProBase_Set_Register_UVL (float fUvl)
设置欠压响应

参数：
- fUvl：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

返回值
- 无

void InnoProBase_Set_Rsense_Value (float fRsense)
设置Rsense电阻值

参数：
- fRsense：电阻(mΩ)

返回值
- 无

float InnoProBase_Get_Register_CV (void)
获取CV寄存器的值

参数：
- 无

返回值
- float：以伏特(V)为单位的值

float InnoProBase_Get_Register_OVA (void)
获取过压阈值的值

参数：
- 无

返回值
- float：以伏特(V)为单位的值

float InnoProBase_Get_Register_UVA (void)
获取欠压阈值的值

参数：
- 无

返回值
- float：以伏特(V)为单位的值

float InnoProBase_Get_Register_CC (void)
获取恒流的值

参数：
- 无

返回值
- float：以安培(A)为单位的值

float InnoProBase_Get_Register_CDC (void)
获取输出线压降补偿的值

参数：
- 无

返回值
- float：以mV为单位的值

float InnoProBase_Get_Register_VKP (void)
获取恒定输出功率拐点电压的值

参数：
- 无

返回值
- float：以伏特(V)为单位的值

float InnoProBase_Get_Register_VBEN (void)
获取VBEN设置的值

参数：
- 无

返回值
- float：
  - 3 – 使能VBEN/禁止VDIS
  - 1 – 禁止VBEN/不复位
  - 0 – 禁止VBEN/复位

float InnoProBase_Get_Register_UVL (void)
设置欠压响应

参数：
- 无

返回值
- float：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

float InnoProBase_Get_Rsense_Value (void)
获取Rsense电阻值

参数：
- 无

返回值
- float：电阻(mΩ)

float Inno4Pro_Compute_CV (float fSetPt)
InnoSwitch4-Pro输出电压(CV)计算。根据10mV/LSB分辨率计算。将值限制在3V至24V之间

参数：
- fSetPt：设置点值（3至24V）

返回值
- float：300至2400

float Inno4Pro_Compute_OV (float fSetPt)
InnoSwitch4-Pro过压阈值(OVA)计算。根据100mV/LSB分辨率计算。将值限制在6.2V至25V之间

参数：
- fSetPt：设置点值（6.2至25V）

返回值
- float：62至250

float Inno4Pro_Compute_UV (float fSetPt)
InnoSwitch4-Pro欠压阈值(UVA)计算。根据100mV/LSB分辨率计算。将值限制在3V至24V之间

参数：
- fSetPt：设置点值（3至24V）

返回值
- float：30至240

float Inno4Pro_Compute_CDC (float fSetPt)
InnoSwitch4-Pro输出线压降补偿(CDC)计算。根据50mV/LSB分辨率计算。将值限制在0mV至600mV之间

参数：
- fSetPt：设置点值（0至600mV）

返回值
- float：0至12

float Inno4Pro_Compute_CC (float fSetPt)
InnoSwitch4-Pro恒流调整(CC)计算。根据0.16mV/步长/Rsense分辨率计算

参数：
- fSetPt：以安培(A)为单位的设置点值

返回值
- float：25至192LSB

float Inno4Pro_Compute_VKP (float fSetPt)
InnoSwitch4-Pro输出功率拐点电压(VKP)计算。根据100mV/LSB分辨率计算

参数：
- fSetPt：设置点值（5.3至24V）

返回值
- float：53至240

float Inno4Pro_Compute_VBEN (float fSetPt)
InnoSwitch4-Pro串联母线开关控制(VBEN)计算应用0至3的限值。

参数：
- 无

返回值
- float：
  - 3 – 使能VBEN/禁止VDIS
  - 1 – 禁止VBEN/不复位
  - 0 – 禁止VBEN/复位

void InnoProBase_Encode_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
处理输入数据到十六进制LSB和MSB（无奇偶校验位）的转换。然后将这些值存储到缓冲区中。

参数：
- u16Temp：要转换的值
- *u8WriteBuffer：指向存储数据的内存位置的指针

返回值
- 无

bool InnoProBase _OddParity (uint8_t u8OddParity)
处理奇校验位检测。

参数：
- u8OddParity：要使用奇偶校验进行评估的值

返回值
- bool：
  - true – 奇数个1
  - false – 偶数个1

void InnoProBase _Format_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
处理输入数据到十六进制LSB和MSB（有已评估的奇偶校验位）的转换。然后将这些值存储到缓冲区中。

参数：
- u16Temp：要转换的值
- *u8WriteBuffer：指向存储数据的内存位置的指针

返回值
- 无

uint8_t InnoProBase _AddOddParity (uint16_t u16Temp)
评估一个字节的奇偶校验

参数：
- u16Temp：要评估并添加奇偶校验位的值

返回值
- uint8_t：有奇偶校验的格式化字节

void Inno4Pro_Process_Volt_Buffers (void)
处理在InnoSwitch4-Pro电压相关寄存器上写入值的准备工作，这些寄存器是过压和欠压阈值

参数：
- 无

返回值
- 无

bool InnoProBase_Detect_Voltage_Request (void)
检查是否有新的电压请求。此外，它还会发出b_Volt_Setting信号，指示电压是升高还是降低

参数：
- 无

返回值
- bool：
  - true – CV值变化
  - false – 无变化

bool InnoProBase_Detect_Current_Request (void)
检查是否有新的电流请求

参数：
- 无

返回值
- bool：
  - true – CC值变化
  - false – 无变化

void Inno4Pro_Initialization (void)
处理所有要写入InnoSwitch 4-Pro作为初始化的常见I2C配置。这将读取InnoSwitch4-Pro系统就绪信号，检查InnoSwitch4-Pro是否准备好进行通信。一旦InnoSwitch4-Pro准备就绪，该函数将配置基本操作所需的初始寄存器

参数：
- 无

返回值
- 无

void Inno4Pro_Vbus_Switch_Control (bool bEnableVben)
Vbus开关控制（VBEN控制）。禁止VBEN时，默认使能看门狗计时器。需要禁止看门狗计时器才能再次使能VBEN。

在关断Vbus开关之前，会仔细监测输出电压，以正确决定何时关断Vbus开关。

当检测到输出电压处于**HIGH**（高电平）设置（>5V）时，先将UV阈值设置为3V，CV设置为5V，然后禁止VBEN

参数：
- bEnableVben：
  - 1 – 使能VBEN
  - 0 – 禁止VBEN且复位

返回值
- 无

void Inno4Pro_Vbus_Switch_Control_NoReset (uint8_t u8EnableVben)
与**Inno4Pro_Vbus_Switch_Control**类似，但包括禁止VBEN/不复位选项

参数：
- bEnableVben：
  - 3 – 使能VBEN
  - 1 – 禁止VBEN/不复位
  - 0 – 禁止VBEN且复位

返回值
- 无

void Inno4Pro_Bleeder_Enable (bool bEnable)
处理泄放设置。BLEEDER不能长时间处于使能状态，以防止功耗过大

参数：
- bEnable：
  - 3 – 具有自动禁止特性的使能BLEEDER
  - 1 – 使能BLEEDER
  - 0 – 禁止BLEEDER

返回值
- 无

void Inno4Pro_Load_Discharge (bool bEnable)
激活Vbus负载放电(VDIS)。使能VDIS寄存器将自动禁止VBEN

参数：
- bEnable：
  - true：使能负载放电
  - false：禁止负载放电

返回值
- 无

void Inno4Pro_TurnOff_PSU (bool bEnable)
关断电源。需要交流输入重新上电来重新启动电源

参数：
- bEnable：
  - true：使能负载放电
  - false：禁止负载放电

返回值
- 无

void Inno4Pro_FastVI_Disable (bool bDisable)
设置CV和CC命令速度限值

参数：
- bEnable：
  - true：无速度限制
  - false：已使能10ms更新限制

返回值
- 无

void Inno4Pro_CVOnlyMode_Enable (bool bEnable, uint16_t u16Response, uint16_t u16Timer)
设置仅恒压模式。该函数将器件设置为仅恒压、无恒流调整模式。一旦负载电流超过CVOL计时器限值内的设定电流，CVOL故障设置即被激活

参数：
- bEnable：
  - true：CVO模式使能
  - false：CVO模式禁止
- u16Response：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应
- u16Timer：
  - 3 – 16 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

返回值
- 无

void Inno4Pro_Write_Volts (float fSetPtCV)
无泄放控制的输出电压控制。用于更新CV寄存器的值

参数：
- fSetPtCV：以伏特(V)为单位的输出电压设置点

返回值
- 无

void Inno4Pro_Write_Over_Volts (float fSetPtOVA, uint16_t u16OVL)
写入过压保护设置。

参数：
- fSetPtOVA：以伏特(V)为单位的过压阈值
- u16OVL：过压响应
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

返回值
- 无

void Inno4Pro_Write_Under_Volts (float fSetPtUVA, uint16_t u16Uv_FaultResp, uint16_t u16Uv_timer)
写入欠压保护设置

参数：
- fSetPtUVA：以伏特(V)为单位的欠压阈值
- u16UVL：欠压响应
  - 3 – 关输出

- 2 – 自动重启动
- 1 – 锁存关断
- 0 – 无响应

- u16Uv_timer：欠压响应
  - 3 – 64 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

返回值
- 无

void Inno4Pro_Write_Cable_Drop_Comp (float fSetPtCDC)
写入输出线压降补偿(CDC)设置。

参数：
- fSetPtCDC：以mV为单位的输出线压降补偿

返回值
- 无

void Inno4Pro_Write_Amps (float fSetPtCC)
恒流(CC)控制

参数：
- fSetPtCC：以安培(A)为单位的恒流设置点

返回值
- 无

void Inno4Pro_Write_Volt_Peak (float fSetPtVpk)
恒定输出功率电压阈值(VKP)控制

参数：
- fSetPtVkp：以伏特(V)为单位的恒定输出功率电压

返回值
- 无

void Inno4Pro_Write_CCSC_Fault_Response (uint16_t u16Response)
写入输出短路故障响应设置

参数：
- u16Response：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

返回值
- 无

void Inno4Pro_Write_ISSC_Fault_Response (uint16_t u16Response, uint16_t u16Frequency, uint16_t u16CC)
写入IS引脚短路故障响应、频率和电流限流点设置

参数：
- u16Response：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应
- u16Frequency：
  - 3 – 60 kHz
  - 2 – 40 kHz
  - 1 – 30 kHz
  - 0 – 50 kHz
- u16CC：
  - 7 – 112 LSB
  - 6 – 96 LSB
  - 5 – 80 LSB
  - 4 – 64 LSB
  - 3 – 48 LSB
  - 2 – 32 LSB
  - 1 – 16 LSB

返回值
- 无

void Inno4Pro_Write_Watchdog_Timer (uint16_t u16Timer)
写入看门狗计时器设置。确定器件在触发看门狗故障前将继续工作多长时间

参数：
- u16Timer：
  - 3 – 2 s
  - 2 – 1 s
  - 1 – 0.5 s
  - 0 – 无看门狗

返回值
- 无

void Inno4Pro_Write_Interupt_Mask (uint16_t u16IntMask)
写入中断掩码设置。必须为每个单独的故障条件使能中断掩码寄存器，才能激活该特性。

一旦发生故障，中断掩码将被复位，并且必须重新使能特定的相关故障以激活SCL报告方案

参数：
- u16IntMask – 中断位掩码设置

返回值
- 无

void Inno4Pro_Write_OTP_Hysteresis (uint16_t u16Otp)
设置过温滞回。当次级控制器结温升高到125 C时，主动VOUT引脚泄放功能将关断。在控制器温度降至设定滞回值以下之前，不允许重新使能泄放

参数：
- u16Otp – 过温滞回设置
  - 0 – 40 C
  - 1 – 60 C

返回值
- 无

void Inno4Pro_Write_CV_Load (uint16_t u16Load)
写入恒压负载设置。InnoSwitch4-Pro中的恒流调整模式可针对应用所需的恒压(CV)型负载进行优化。使能该命令寄存器可减少仅CV负载的输出电流纹波。仅当必须支持CV负载时才应使用该设置

参数：
- u16Load – CV负载寄存器的值

返回值
- 无

void Inno4Pro_Write_Loop_Speed1 (uint16_t u16LoopSpeed)
写入环路速度1设置。如果应用中需要更快的动态响应，InnoSwitch4-Pro中包含的命令寄存器可减少低到高输出电压转换的时间。

注：使用默认或推荐设置以外的其他值可能会导致振荡

参数：
- u16LoopSpeed – 环路速度1的值

返回值
- 无

void Inno4Pro_Write_Loop_Speed2 (uint16_t u16LoopSpeed)
写入环路速度2设置。如果应用中需要更快的动态响应，InnoSwitch4-Pro中包含的命令寄存器可减少低到高输出电压转换的时间。

注：使用默认或推荐设置以外的其他值可能会导致振荡

参数：
- u16LoopSpeed – 环路速度2的值

返回值
- 无

bool Inno4Pro_Write_VI (float fSetPtCV, float fSetPtCC)
具有泄放控制和恒流(CC)控制的输出电压控制。自动计算UVA和OVA设置。OVA设置为CV设置点的124%，UVA设置为3V。

参数：
- fSetPtCV：以伏特(V)为单位的输出电压设置点
- fSetPtCC：以安培(A)为单位的恒流设置点

返回值
- bool：
  - true – 过程完成
  - false – 过程未完成

void Inno4Pro_VBUSSC (uint16_t u16VSSC_response,
uint16_t u16Vsamples,
uint16_t u16CCThreshold)
定义器件对串联母线开关短路故障的响应方式。

参数：
- u16VSSC_response：
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应
- u16VSamples：
  - 3 – 4个采样
  - 2 – 3个采样
  - 1 – 2个采样
  - 0 – 1个采样
- u16CCThreshold：
  - 3 – d’ 72
  - 2 – d’ 64
  - 1 – d’ 32
  - 0 – d’ 48

返回值
- 无

void Inno4Pro_DCMonly (bool bEnable)
具有泄放控制和恒流(CC)控制的输出电压控制

参数：
- bEnable：
  - 1 – 使能仅DCM
  - 0 – 禁止仅DCM

返回值
- 无

bool Inno4Pro_Process_Voltage (bool bVoltIncrease)
处理电压递增/递减的命令序列。该函数遵循特定的命令序列，以避免意外触发UV或OV故障

参数：
- bVoltIncrease – 表示是电压增大还是减小的转换

返回值
- 无

uint16_t InnoProBase_Telemetry (uint8_t ReadBack_Address)
处理InnoSwitch4-Pro的常见I2C读回遥测。该功能读取特定的InnoSwitch4-Pro遥测寄存器。

参数：
- ReadBack_Address – 寄存器读回地址

返回值
- 寄存器的2字节值

bool InnoProBase_Read_Bit (uint8_t ReadBack_Address,
uint8_t Bit)
处理InnoSwitch4-Pro的I2C读位

参数：
- ReadBack_Address – 寄存器读回地址
- Bit – 位索引

返回值
- 位的值

uint8_t InnoProBase_Read_Byte (uint8_t ReadBack_Address,
bool bHighByte)
处理InnoSwitch4-Pro的I2C读字节

参数：
- ReadBack_Address – 寄存器读回地址
- bHighByte – 表示MSB或LSB
  - 1 – MSB
  - 0 – LSB

返回值
- 1字节

uint8_t InnoProBase_Read_2Bits (uint8_t ReadBack_Address,
uint8_t u8ShiftCnt)
处理InnoSwitch4-Pro的I2C读2个位。参数u8ShiftCnt决定读回值的右移次数。然后返回最后一个位[1:0]。

参数：
- ReadBack_Address – 寄存器读回地址
- u8ShiftCnt – 右移计数

返回值
- uint8_t：两位的值（0到3）

float InnoProBase_Read_SetPoint (uint16_t
ReadBack_Address,
float
fMultiplier)
处理InnoSwitch4-Pro的I2C设置点和阈值。该函数主要用于CV、OV和UV设置点

参数：
- ReadBack_Address – 寄存器读回地址
- fMultiplier – 与读回值相乘的值

返回值
- float：读回值与fMultiplier的乘积

float Inno4Pro_Read_CV_SetPoint (void)
读取遥测寄存器READ1 - 输出电压设置点

参数：
- 无

返回值
- float：以伏特(V)为单位的输出电压

float Inno4Pro_Read_Output_CC_SetPoint (void)
读取遥测寄存器READ2 – 输出电流设置点

参数：
- 无

返回值
- float：以安培(A)为单位的恒流设置点

float Inno4Pro_Read_OV_Threshold (void)
读取遥测寄存器READ3 - 过压阈值

参数：
- 无

返回值
- float：以伏特(V)为单位的过压阈值

float Inno4Pro_Read_UV_Threshold (void)
读取遥测寄存器READ4 - 欠压阈值

参数：
- 无

返回值
- float：以伏特(V)为单位的欠压阈值

float Inno4Pro_Read_CC_SetPoint (void)
读取遥测寄存器READ5 – 恒流设置点

参数：
- 无

返回值
- float：以安培(A)为单位的恒流设置点

float Inno4Pro_Read_CP_Threshold (void)
读取遥测寄存器READ5 – 恒功率设置点

参数：
- 无

返回值
- float：以伏特(V)为单位的恒定输出功率拐点电压

uint8_t Inno4Pro_Read_OV_Fault_Response (void)
读取遥测寄存器READ6 - 过压故障响应

参数：
- 无

返回值
- uint8_t：故障响应
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

uint8_t Inno4Pro_Read_UV_Fault_Response (void)
读取遥测寄存器READ6 - 欠压阈值故障响应

参数：
- 无

返回值
- uint8_t：故障响应
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

uint8_t Inno4Pro_Read_OutputSckt_Fault_Response (void)
读取遥测寄存器READ6 - 输出短路故障响应

参数：
- 无

返回值
- uint8_t：故障响应
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

uint8_t Inno4Pro_Read_IsPinShort_Fault_Response (void)
读取遥测寄存器READ6 - IS引脚短路故障响应

参数：
- 无

返回值
- uint8_t：故障响应
  - 3 – 关输出
  - 2 – 自动重启动
  - 1 – 锁存关断
  - 0 – 无响应

uint8_t Inno4Pro_Read_UV_Fault_Timer (void)
读取遥测寄存器READ6 - 欠压计时器

参数：
- 无

返回值
- uint8_t：故障计时器
  - 3 – 64 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

uint8_t Inno4Pro_Read_Watchdog_Timer (void)
读取遥测寄存器READ6 - 看门狗计时器

参数：
- 无

返回值
- uint8_t：故障计时器
  - 3 – 2 s
  - 2 – 1 s
  - 1 – 0.5 s
  - 0 – 无看门狗

uint8_t Inno4Pro_Read_CvMode_Fault_Response (void)
读取遥测寄存器READ6 -恒压模式故障响应

参数：
- 无

返回值
- uint8_t：故障响应

- 3 – 关输出
- 2 – 自动重启动
- 1 – 锁存关断
- 0 – 无响应

uint8_t Inno4Pro_Read_CvMode_Timer (void)
读取遥测寄存器READ6 -恒压模式计时器

参数：
- 无

返回值
- uint8_t：故障计时器
  - 3 – 64 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

bool Inno4Pro_Read_VbusSwitch (void)
读取遥测寄存器READ6上的第14位 - VBUS开关使能

参数：
- 无

返回值
- uint8_t：故障计时器
  - 3 – 64 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

bool Inno4Pro_Read_Bleeder (void)
读取遥测寄存器READ6上的第13位 - 最小负载（泄放）

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_PsuOff (void)
读取遥测寄存器READ6上的第12位 - 关断电源（器件锁存）

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_FastVI (void)
读取遥测寄存器READ6上的第11位 - 快速VI命令

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_CvoMode (void)
读取遥测寄存器READ6上的第10位 - 仅恒压模式

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_OtpFaultHyst (void)
读取遥测寄存器READ6上的第9位 - 过温保护

参数：
- 无

返回值
- bool：
  - true – 60 C
  - false – 40 C

float Inno4Pro_Read_Cable_Drop_Comp (void)
读取遥测寄存器READ6 -输出线压降补偿

参数：
- 无

返回值
- float：以mV为单位的输出线压降补偿

float Inno3Pro_Read_Amps (void)
读取遥测寄存器READ7 - 实测输出电流

参数：
- 无

返回值
- float：以安培(A)为单位的输出电流

float Inno3Pro_Read_Volts (void)
读取遥测寄存器READ9 -实测输出电压

参数：
- 无

返回值
- float：以伏特(V)为单位的输出电压

bool Inno4Pro_Read_Status_InterruptEnable (void)
读取遥测寄存器READ10上的第15位 - 中断使能

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_Status_SystemReady (void)
读取遥测寄存器READ10上的第14位 - 系统就绪信号。

注：READ10遥测寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 就绪
  - false – 未就绪

bool Inno4Pro_Read_Status_OutputDischarge (void)
读取遥测寄存器READ10上的第13位 - 输出放电

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 已使能
  - false – 已禁止

bool Inno4Pro_Read_Status_HighSwitchFreq (void)
读取遥测寄存器READ10上的第12位 - 开关频率高

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 高频率
  - false – 低频率

bool Inno4Pro_Read_Status_OtpFault (void)
读取遥测寄存器READ10上的第9位 - 过温保护

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 保护已使能
  - false – 保护已禁止

bool Inno4Pro_Read_Status_Vout2pct (void)
读取遥测寄存器READ10上的第5位 - 2% BLEEDER使能。InnoSwitch4-Pro将自动激活VOUT引脚上的弱电流泄放，直到输出电压稳定到CV设置点的2%以下

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 2% BLEEDER使能
  - false – 2%BLEEDER禁止

bool Inno4Pro_Read_Status_Vout10pct (void)
读取遥测寄存器READ10上的第4位 - VOUTADC > 1.1*Vout。InnoSwitch4-Pro将自动激活VOUT引脚上的弱电流泄放，直到输出电压稳定到CV设置点的10%以下

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 10% BLEEDER使能
  - false – 10% BLEEDER禁止

bool Inno4Pro_Read_Status_IsPinShort (void)
读取遥测寄存器READ10上的第3位 - 检测到IS引脚短路

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 检测到IS引脚短路
  - false – 无IS引脚短路

bool Inno4Pro_Read_Status_OutputShorCkt (void)
读取遥测寄存器READ10上的第2位 – 检测到输出短路

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 检测到输出短路
  - false – 无输出短路

bool Inno4Pro_Read_Status_UV_Fault (void)
读取遥测寄存器READ10上的第1位 - 输出电压UV故障比较器

注：该寄存器值是瞬时的，只要条件不再有效就会清零。

参数：
- 无

返回值
- bool：
  - true – 检测到欠压故障
  - false – 无欠压故障

bool Inno4Pro_Read_Status_OV_Fault (void)
读取遥测寄存器READ10上的第0位 - 输出电压OV故障比较器

参数：
- 无

返回值
- bool：
  - true – 检测到过压故障
  - false – 无过压故障

float Inno4Pro_Read_AmpsAverage (void)
读取遥测寄存器READ12 - 平均输出电流

参数：
- 无

返回值
- float：以安培(A)为单位的平均输出电流

float Inno4Pro_Read_VoltsAverage (void)
读取遥测寄存器READ13 - 平均输出电压

参数：
- 无

返回值
- float：以伏特(V)为单位的平均输出电压

float Inno3Pro_Read_Voltage_DAC (void)
读取遥测寄存器READ14 - 电压DAC

参数：
- 无

返回值
- float：以伏特(V)为单位的DAC输出电压

bool Inno4Pro_Read_Status_CvoMode_AR (void)
读取遥测寄存器READ16上的第15位 - CVO模式自动重启动(AR)

参数：
- 无

返回值
- bool：
  - true – 发生CVO模式自动重启动
  - false – 无故障

bool Inno4Pro_Read_Status_IsPinShort_AR (void)
读取遥测寄存器READ16上的第12位 - IS引脚短路自动重启动(AR)

参数：
- 无

返回值
- bool：
  - true – 发生IS引脚短路AR
  - false – 无故障

bool Inno4Pro_Read_Status_OutputShortCkt_AR (void)
读取遥测寄存器READ16上的第12位 – 输出短路自动重启动(AR)

参数：
- 无

返回值
- bool：
  - true – 发生输出短路AR
  - false – 无故障

bool Inno4Pro_Read_Status_OV_AR (void)
读取遥测寄存器READ16上的第10位 - 输出电压OV自动重启动(AR)

参数：
- 无

返回值
- bool：
  - true – 发生过压故障AR
  - false – 无故障

bool Inno4Pro_Read_Status_UV_AR (void)
读取遥测寄存器READ16上的第9位 - 输出电压UV自动重启动(AR)

参数：
- 无

返回值
- bool：
  - true – 发生欠压故障AR
  - false – 无故障

bool Inno4Pro_Read_Status_LatchOff (void)
读取遥测寄存器READ16上的第7位 -- 发生锁存关断(LO)

参数：
- 无

返回值
- bool：
  - true – 发生锁存关断
  - false – 无故障

bool Inno4Pro_Read_Status_CvoMode_LO (void)
读取遥测寄存器READ16上的第6位 - CVO模式锁存关断(LO)

参数：
- 无

返回值
- bool：
  - true – 发生CVO模式锁存关断
  - false – 无故障

bool Inno4Pro_Read_Status_PsuOffCmd (void)
读取遥测寄存器READ16上的第5位 - 收到电源关断命令

参数：
- 无

返回值
- bool：
  - true – 收到电源关断命令
  - false – 无故障

bool Inno4Pro_Read_Status_IsPinShort_LO (void)
读取遥测寄存器READ16上的第4位 - IS引脚短路锁存关断(LO)

参数：
- 无

返回值
- bool：
  - true – 发生IS引脚短路锁存关断
  - false – 无故障

bool Inno4Pro_Read_Status_OV_LO (void)
读取遥测寄存器READ16上的第2位 - 输出电压OV锁存关断(LO)

参数：
- 无

返回值
- bool：
  - true – 发生过压锁存关断
  - false – 无故障

bool Inno4Pro_Read_Status_UV_LO (void)
读取遥测寄存器READ16上的第1位 - 输出电压UV锁存关断(LO)

参数：
- 无

返回值
- bool：
  - true – 发生欠压锁存关断
  - false – 无故障

bool Inno4Pro_Read_Status_BPS_LO (void)
读取遥测寄存器READ16上的第0位 - BPS引脚锁存关断(LO)。该位表示是否在BPS引脚上检测到过压故障

参数：
- 无

返回值
- bool：
  - true – 发生BPS引脚锁存关断
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Mask_CntrlSecondary (void)
读取遥测寄存器READ17上的第15位 - 中断掩码控制次级

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – 控制次级的中断已使能
  - false – 控制次级的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_BpsCurrentLo (void)
读取遥测寄存器READ17上的第13位 -中断掩码BPS电流锁存关断

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – BPS电流锁存关断的中断已使能
  - false – BPS电流锁存关断的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_CvoPkLoadTimer (void)
读取遥测寄存器READ17上的第12位 -中断掩码BPS电流锁存关断

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – CVO模式峰值负载计时器的中断已使能
  - false – CVO模式峰值负载计时器的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_IsPinShort (void)
读取遥测寄存器READ17上的第11位 - 中断掩码IS引脚短路

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – IS引脚短路的中断已使能
  - false – IS引脚短路的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_OutputShortCkt (void)
读取遥测寄存器READ17上的第10位 - 中断掩码输出短路

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – 输出短路的中断已使能
  - false – 输出短路的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_UV (void)
读取遥测寄存器READ17上的第9位 -中断掩码Vout欠压(UV)

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – Vout (UV)的中断已使能
  - false – Vout (UV)的中断已禁止

bool Inno4Pro_Read_Interrupt_Mask_OV (void)
读取遥测寄存器READ17上的第8位 - 中断掩码Vout过压(OV)

注：一旦发生故障，中断掩码将被复位，并且必须重新使能才能激活SCL报告方案

参数：
- 无

返回值
- bool：
  - true – Vout (OV)的中断已使能
  - false – Vout (UV)的中断已禁止

bool Inno4Pro_Read_Interrupt_Stat_CntrlSecondary (void)
读取遥测寄存器READ17上的第6位 - 中断状态控制次级

参数：
- 无

返回值
- bool：
  - true – 发生控制次级故障
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_OMF(void)
如果使能中断掩码，则每当工作模式从CV变为CC或反之时，都会引发中断。

参数：
- 无

返回值
- bool：
  - true – 工作模式发生变化
  - false – 工作模式无变化

bool Inno4Pro_Read_Interrupt_Stat_VBUSSC (void)
该函数表明当VBEN被禁止时，IS引脚检测到的电流大于设置的VBUSSC寄存器

参数：
- 无

返回值
- bool：
  - true – 发生VBUSSC故障
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_BpsCurrentLo (void)
读取遥测寄存器READ17上的第6位 - BPS的中断状态

参数：
- 无

返回值
- bool：
  - true – 发生BPS故障
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_CvoPkLoadTimer (void)
读取遥测寄存器READ17上的第5位 - CVO模式峰值负载计时器的中断状态

参数：
- 无

返回值
- bool：
  - true – 发生恒压模式故障
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_IsPinShort (void)
读取遥测寄存器READ17上的第3位 - 状态IS引脚短路的中断状态

参数：
- 无

返回值
- bool：
  - true – 发生IS引脚短路
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_OutputShortCkt(void)
该函数表明已发生CCSC故障，并在SCL上生成了中断

参数：
- 无

返回值
- bool：
  - true – 发生输出短路
  - false – 无故障
  -

bool Inno4Pro_Read_Interrupt_Stat_UV (void)
读取遥测寄存器READ17上的第1位 - 状态Vout(UV)的中断状态

参数：
- 无

返回值
- bool：
  - true – 发生欠压故障
  - false – 无故障

bool Inno4Pro_Read_Interrupt_Stat_OV (void)
读取遥测寄存器READ17上的第0位 - 状态Vout(OV)的中断状态

参数：
- 无

返回值
- bool：
  - true – 发生过压故障
  - false – 无故障

修订版本	注释	日期
A	初始版本。	01/20/23

有关最新产品信息，请访问：www.power.com

Power Integrations reserves the right to make changes to its products at any time to improve reliability or manufacturability. Power Integrations does not assume any liability arising from the use of any device or circuit described herein. POWER INTEGRATIONS MAKES NO WARRANTY HEREIN AND SPECIFICALLY DISCLAIMS ALL WARRANTIES INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

Patent Information

The products and applications illustrated herein (including transformer construction and circuits external to the products) may be covered by one or more U.S. and foreign patents, or potentially by pending U.S. and foreign patent applications assigned to Power Integrations. A complete list of Power Integrations patents may be found at www.power.com. Power Integrations grants its customers a license under certain patent rights as set forth at www.power.com/ip.htm.

Life Support Policy

POWER INTEGRATIONS PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF POWER INTEGRATIONS. As used herein:

1. A Life support device or system is one which, (i) is intended for surgical implant into the body, or (ii) supports or sustains life, and (iii) whose failure to perform, when properly used in accordance with instructions for use, can be reasonably expected to result in significant injury or death to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Power Integrations, the Power Integrations logo, CAPZero, ChiPhy, CHY, DPA-Switch, EcoSmart, E-Shield, eSIP, eSOP, HiperPLC, HiperPFS, HiperTFS, InnoSwitch, Innovation in Power Conversion, InSOP, LinkSwitch, LinkZero, LYTSwitch, SENZero, TinySwitch, TOPSwitch, PI, PI Expert, PowiGaN, SCALE, SCALE-1, SCALE-2, SCALE-3 and SCALE-iDriver, are trademarks of Power Integrations, Inc. Other trademarks are property of their respective companies. ©2020, Power Integrations, Inc.

Power Integrations全球销售支持网络

全球总部
5245 Hellyer Avenue
San Jose, CA 95138, USA
Main: +1-408-414-9200
Customer Service:
Worldwide: +1-65-635-64480
Americas: +1-408-414-9621
e-mail: usasales@power.com

中国（上海）
徐汇区漕溪北路88号圣爱广场
1601-1603室
上海|中国，200030
电话：+86-21-6354-6323
电子邮箱：chinasales@power.com

中国（深圳）
南山区科技南八路二号豪威科技大厦17层
深圳|中国，518057
电话：+86-755-8672-8689
电子邮箱：chinasales@power.com

德国（AC-DC/LED业务销售）
Einsteinring 24
85609 Dornach/Aschheim
Germany
Tel: +49-89-5527-39100
e-mail: eurosales@power.com

德国（门极驱动器销售）
HellwegForum 1
59469 Ense
Germany
Tel: +49-2938-64-39990
e-mail: igbt-driver.sales@power.com

印度
#1, 14th Main Road
Vasanthanagar
Bangalore-560052 India
Phone: +91-80-4113-8020
e-mail: indiasales@power.com

意大利
Via Milanese 20, 3rd. Fl.
20099 Sesto San Giovanni (MI) Italy
Phone: +39-024-550-8701
e-mail: eurosales@power.com

日本
Yusen Shin-Yokohama 1-chome
Bldg. 1-7-9, Shin-Yokohama,
Kohoku-ku Yokohama-shi,
Kanagawa 222-0033 Japan
Phone: +81-45-471-1021
e-mail: japansales@power.com

韩国
RM 602, 6FL
Korea City Air Terminal B/D, 159-6 Samsung-Dong, Kangnam-Gu,
Seoul, 135-728, Korea
Phone: +82-2-2016-6610
e-mail: koreasales@power.com

新加坡
51 Newton Road
#19-01/05 Goldhill Plaza
Singapore, 308900
Phone: +65-6358-2160
e-mail: singaporesales@power.com

台湾地区
5F, No. 318, Nei Hu Rd., Sec. 1
Nei Hu Dist.
Taipei 11493, Taiwan R.O.C.
Phone: +886-2-2659-4570
e-mail: taiwansales@power.com

英国
Building 5, Suite 21
The Westbrook
Centre Milton Road
Cambridge
CB4 1YG
Phone: +44 (0) 7823-557484
e-mail: eurosales@power.com