Application Note AN-104
InnoSwitch4-Pro Family

Programming Manual

Introduction

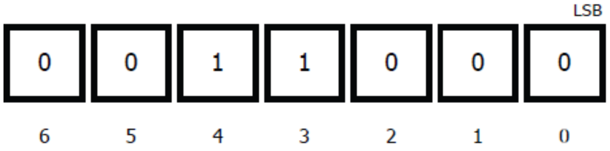
This manual describes the software implementation including driver libraries used to control InnoSwitch4-Pro operations. Important aspects of this document are calculations for the values to be programmed for various configurations such as voltage, current, cable drop compensation, constant power, I^{2}C command sequences to prevent any unexpected behaviors, device responses and code examples.

The following conventions will be used

[A] – Slave acknowledgement
[a] – Master acknowledgement
[na] – Master NACK
[W] – Write command (1'b0)
[r] – Read command (1'b1)
[PI_COMMAND] – PI Command register address assignments
[TELEMETRY_REGISTER_ADDRESS] – Telemetry register address assignments

InnoSwitch4-Pro I^{2}C Communication

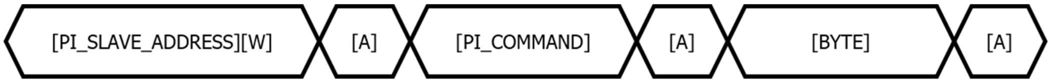
The InnoSwitch4-Pro has a 7-bit slave address of 0x18 (7'b001 1000)



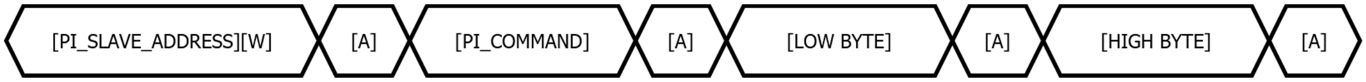
InnoSwitch4-Pro Write Operation

The I^{2}C write operation format is as follows.

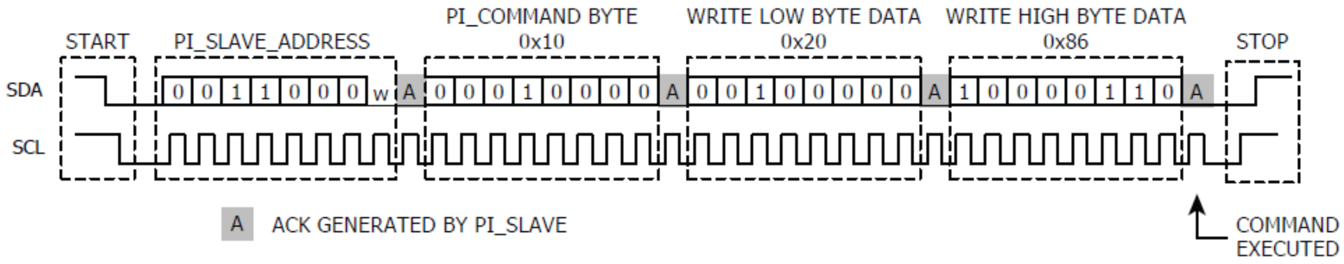
For one BYTE data writes



For one WORD or 2 BYTE data writes

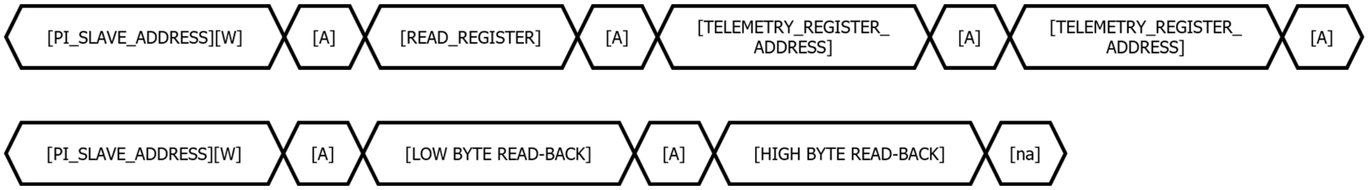


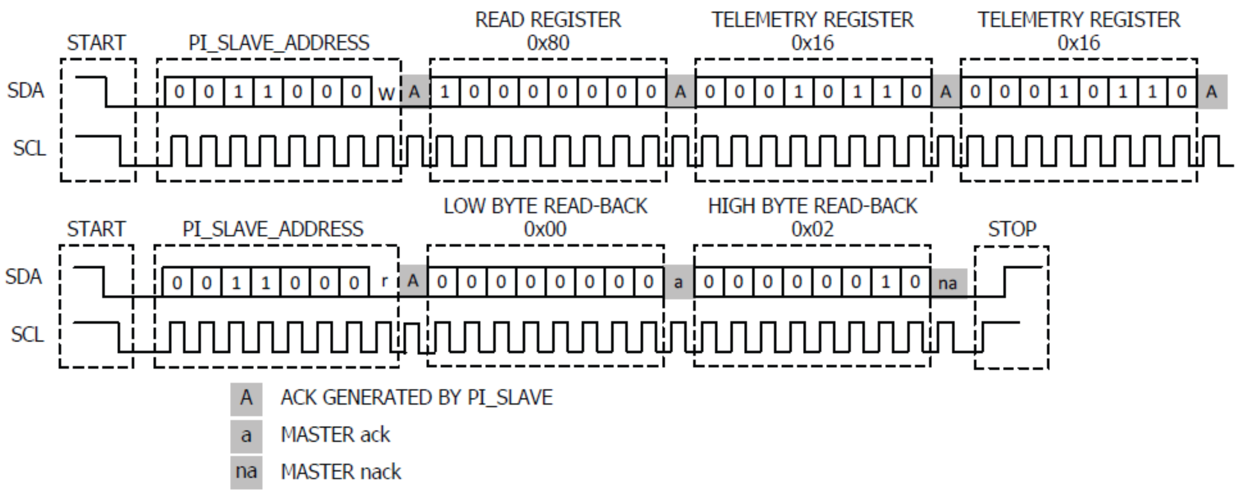
The command below illustrates the I^{2}C packets during a Write 8V to CV register. The [PI_SLAVE_ADDRESS] which originally is 0x18 (7'b001 1000) is shifted to the left by 1 bit to occupy the first 7 bits from the first byte sent through the I^{2}C communication. The LSB of this byte is for commanding R/W operation. Writing '**0**' to this bit is for writing a data to any of the [PI_COMMAND] in the slave and writing a '**1**' to this bit is for reading a data from any of the [TELEMETRY_REGISTER_ADDRESS]. So for a write operation, the [PI_SLAVE_ADDRESS] combined with the '0' at the R/W bit indicates a write operation and sends 0x30 as the first byte over I^{2}C communication as shown below.



InnoSwitch4-Pro Read Operation

The I^{2}C read operation format is as follows:



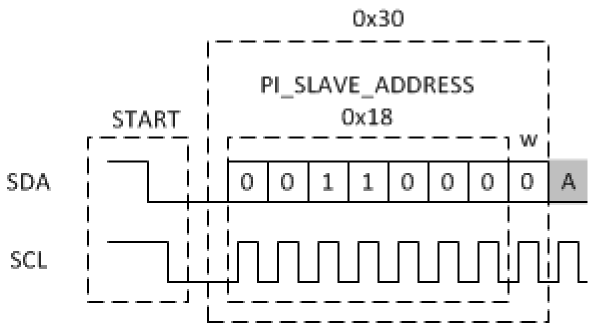


A ACK GENERATED BY PI_SLAVE

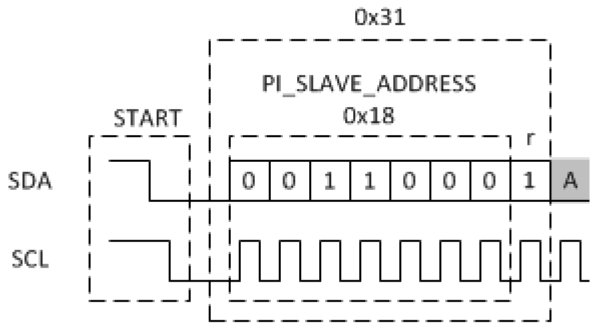
a MASTER ack

na MASTER nack

The address of the telemetry register from which data needs to be read, is first written to the Read Register which acts like a pointer to this telemetry register. The address of the Read Register is 0x80. Since the first operation while reading is a write operation, the first byte of the I^{2}C transaction is as follows



Then in the next I^{2}C operation, a read request is sent to the slave and as a response to this request, the slave sends the Low Byte followed by the High Byte stored at the respective Telemetry register whose address was written to the Read Register. The first byte during this read operation of the I^{2}C transaction is as follows



Constant Voltage Setting Calculations

Output voltage setting values are calculated based on its specific resolution (e.g. 10mV/LSB). High and low limits for this parameter in the code are recommended to ensure the device operation is within the correct range.

Output Voltage Calculation:

Register	Adjustment Range	Resolution
CV	3 V to 24 V	10 mV/LSB

Equation:

$$LSB\ Representation = \frac{Set\ Point\ in\ Volts}{Resolution}$$

Example:

x volts to be converted in decimal representation

$$LSB\ Representation = \frac{x}{\frac{10mV}{LSB}} = \frac{x}{\frac{10}{1000}V}LSB$$

$$LSB\ Representation = x * 100$$

This can be converted to equivalent hexadecimal value and odd parity must be added

Code Example

The saturation macro sets the lower and upper ends for the output voltage. Any **fTemp** value over 2400 is clamped to 2400. Likewise, any value less than 300 is clamped to 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;
}
```

Constant Current Setting Calculations

Constant current (CC) regulation set point is calculated as a percentage of the full scale CC threshold set by the sense resistor between the IS and GND pins. It can be programmed from 20% to 100%.

The ISV(TH) parameter value (with a typical value of ~32 mV) is considered as the full scale CC regulation voltage threshold. This is treated as 100% for CC set-point which in decimal representation is programmed as 192. If a 10 mΩ sense resistor is used, then 3.2 A will be considered as the full scale CC set-point. If a CC set-point value <3.2 A needs to be programmed, then it has to be calculated as a percentage of the full scale CC set-point and then the percentage value needs to be converted to its equivalent decimal representation as shown below.

Equation:

$$Percentage\ = CC\ Set\ Point\ in\ Amps\ *\ Sense\ Resistor\ in\ m\Omega\ *\ \frac{100}{Isv_{(Th)}\ in\ mV}$$

$$Decimal\ Equivalent\ = Percentage\ *\ \frac{192}{100}$$

Example:
Rsense = 10 mΩ
CC set-point = 1.6 A

$$Percentage\ = 1.6A\ *\ 10\ m\Omega\ *\ \frac{100}{32\ mV}$$

$$Percentage\ = 50\ \%$$

$$Decimal\ Equivalent\ = 50\ *\ \frac{192}{100} = 96$$

This can be converted to equivalent hexadecimal value and odd parity must be added.

Code Example

CC set-point computation

$$Setpoint\ = Current * Rsense * \frac{192}{32}$$

```
#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;
}
```

Constant Power Knee Voltage Setting Calculations

Similar to writing the Constant voltage setting, since the resolution of the constant power knee voltage setting (VKP) is 100 mV, the VKP set-point needs to get multiplied by 10 to convert to the equivalent decimal value to be written. It can be programmed from 5.3 V to 24 V.

Equation:

$$LSB\ Representation = \frac{Set\ Point\ in\ Volts}{Resolution}$$

Example:
X volts to be converted in decimal representation

$$LSB\ Representation = \frac{x}{\frac{100mV}{LSB}} = \frac{x}{\frac{100}{1000}V}LSB$$

$$LSB\ Representation =\ x\ *\ 10$$

This can be converted to equivalent hexadecimal value and odd parity must be added

Code Example

A minimum of 53 and maximum of 240 limit for the V_{KP} set-point is recommended to be used in the program.

```
#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;
}
```

Cable Drop Compensation Setting Calculations

Cable drop compensation (CDC) can be programmed from 0 to 600 mV in 50 mV increments. 600 mV corresponds to 12LSB and a 0 mV is equivalent to 0LSB.

Equation:

$$LSB\ Representation = \frac{Set\ Point\ in\ Volts}{Resolution}$$

Example:
x volts to be converted in decimal representation

$$LSB\ Representation = \frac{x}{\frac{50mV}{LSB}} = \frac{x}{\frac{50}{1000}V}LSB$$

$$LSB\ Representation =\ x\ *\ 50$$

This can be converted to hexadecimal value for programming.

Code Example

The value of the CDC set-point is limited to 0 and 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;
}
```

Parity Bit Implementation

Some registers have simple error detection mechanism using odd parity checking. In odd parity bit error checking, the total number of 1s in the binary format of the value (including the parity bit) must be an odd number. Selective registers contain parity bit on each of the Low and High Bytes. Details of all the registers are in the data sheet.

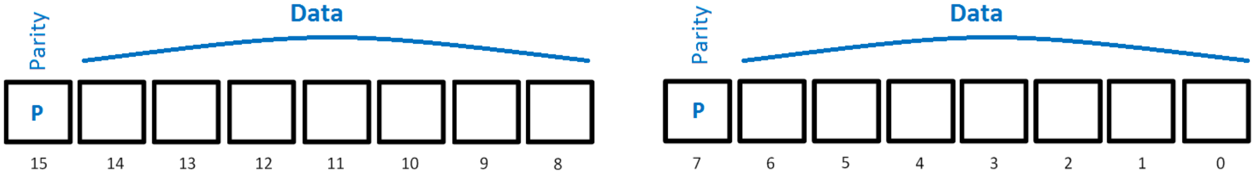
Example:

Register	Output Voltage	LSB Representation	Hex without Parity	Hex with Odd Parity
CV	5 V	500	0x01F4	0x83F4

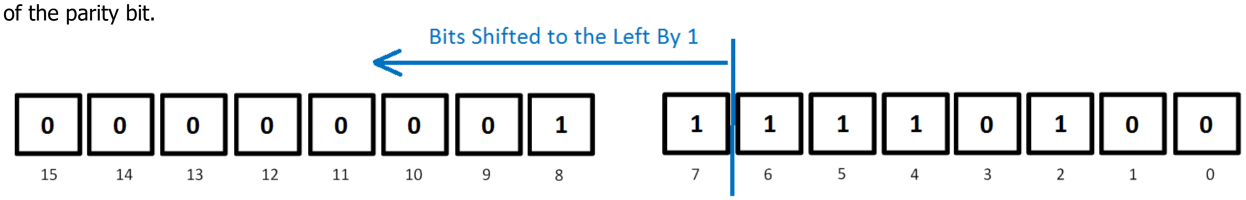
Simple Binary Conversion:

0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Some registers require Bit[15] and Bit[7] to be used for odd parity bit implementation; these bits must not contain the converted binary data.



When converting the data into binary, the binary data starting from the 7th bit of the low byte must be shifted to the left by one place to reserve the position of the parity bit.



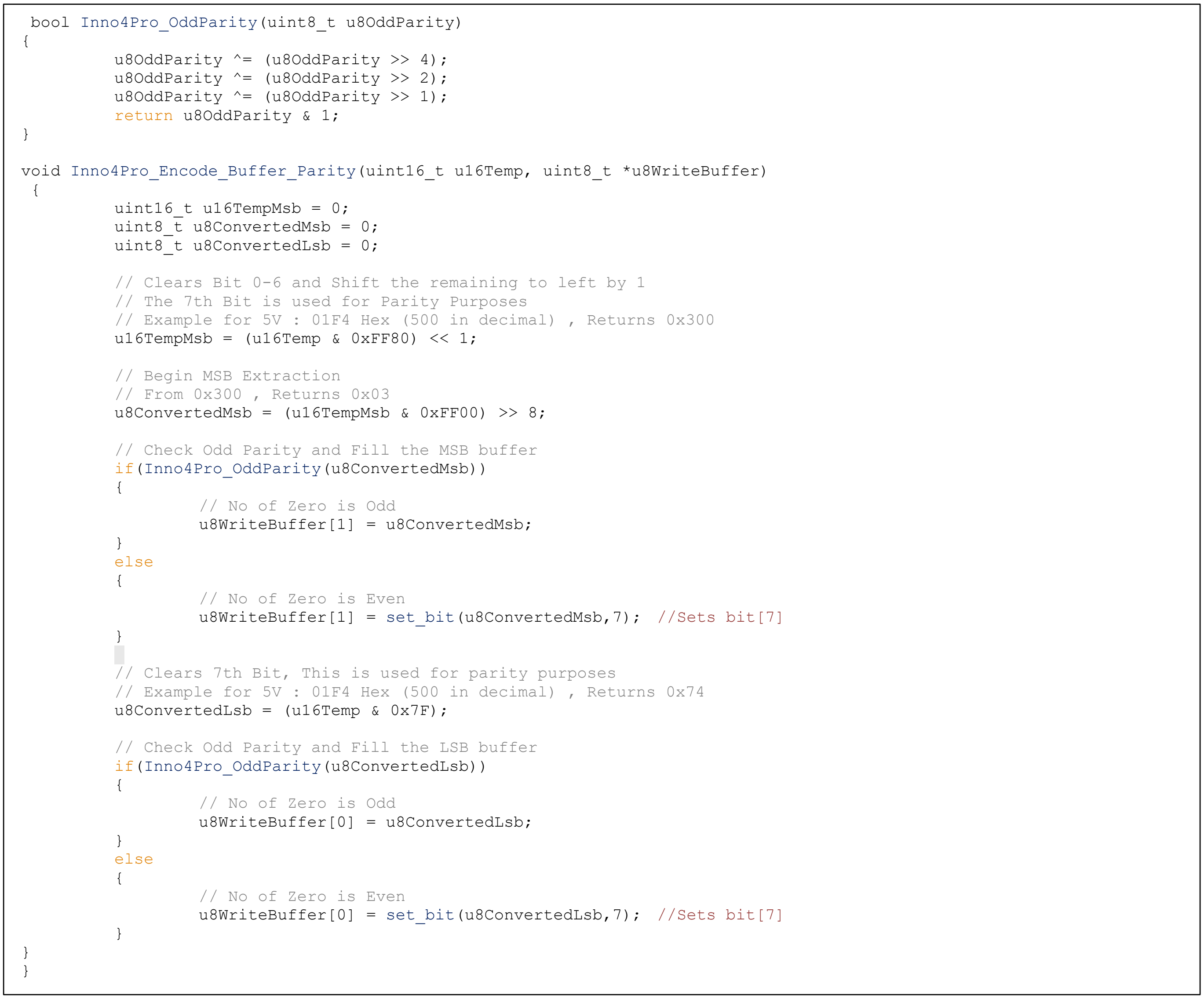
0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Parity								Parity							
P	0	0	0	0	0	1	1	P	1	1	1	0	1	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

If the data has odd number of 1s, the parity bit is **0**, otherwise it is set to **1**.

1	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Parity Code Example



```
 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]
         }
}
}
```

Command Sequences

In order to update the Output Voltage (CV) and Constant Current (CC), a certain sequence of commands is needed to be followed in order to avoid inadvertent triggering of Under Voltage (UV) and Over Voltage (OV) faults

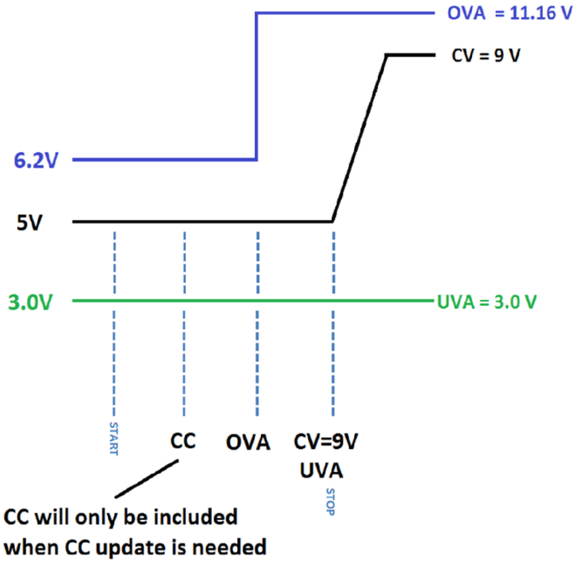
Voltage Increment Process

When the Over Voltage Set-point (OVA) is not increased to a higher value before writing the new CV value, it will trigger OV protection. The OVA needs to be programmed prior to programming the CV register.

For power supplies initially running at low output voltage and high output current that transition to high output voltage and low output current condition, if the CC set-point is not reduced before raising the voltage, then the power supply could get power limited from drawing high load at a high output voltage. The CC register needs to be programmed before increasing the output voltage as well.

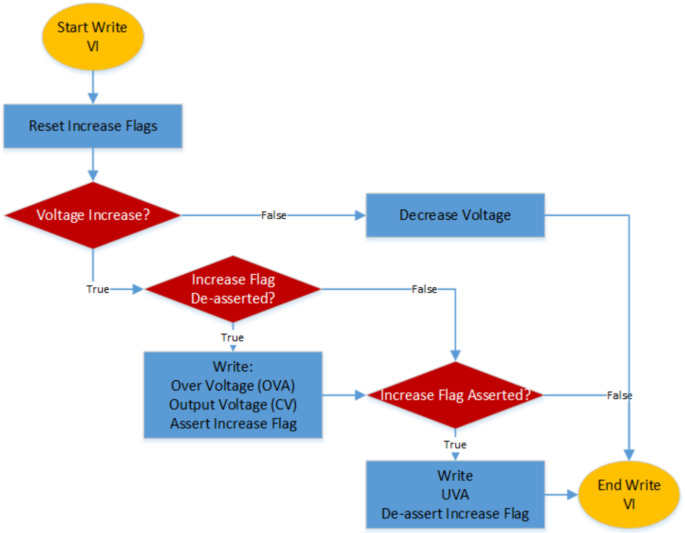
Another scenario that could happen is when the CC is set lower than the current CC set-point and cause the power supply to operate in the CC region. This sudden change may lower the output voltage below the UVA set-point and trigger the UV protection.

The figure below shows the command sequence for incrementing a voltage to a higher value, with 10ms update limit (Fast VI Command) initially disabled. A minimum of 150 us delay between consecutive commands through the I^{2}C bus is still required between all commands regardless of the status of the FAST VI Command register.



In the figure above, the command sequence is CC, OVA, CV, and UVA. The CC set-point will only be updated when requested. The example above shows that the OVA tracks the CV value. The OVA set-point is always 24% higher than the CV value.

Voltage Increment Flowchart



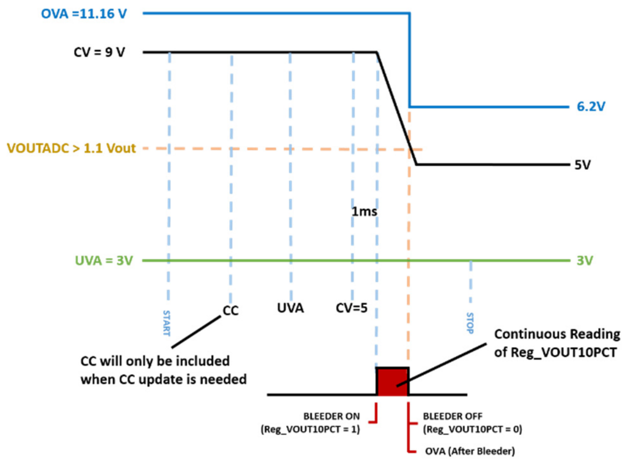
Voltage Increment Code Example

```
//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;

}
```

Voltage Decrement Process

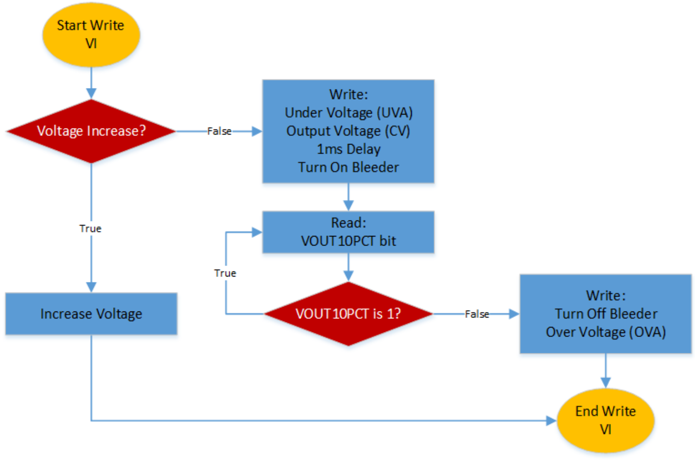
When the output voltage is set to a lower value, the BLEEDER needs to be enabled for a faster transition of the output voltage especially at no-load conditions. If the BLEEDER is not turned on during a large voltage transition, e.g. 20 V to 5 V, then the long transition time with no switching activity could trigger an auto-restart (AR) of the InnoSwitch4-Pro IC. The BLEEDER needs to be turned off as soon as the voltage reaches the desired set-point, otherwise it could lead to undesirable power dissipation and heating of the IC.

Usually high load current can be drawn at lower output voltage compared to higher output voltages for the same power ratings. Therefore, while transitioning to a lower output voltage, it is recommended to first increase the CC set-point limit before lowering the output voltage. This prevents the power supply from entering into constant current (CC) mode of operation, especially if it had been operating in constant power region before the transition. Unexpected operation of the power supply in CC mode of operation due to not increasing the CC limit before transition could also trigger UV protection if the output voltage falls below UVA.

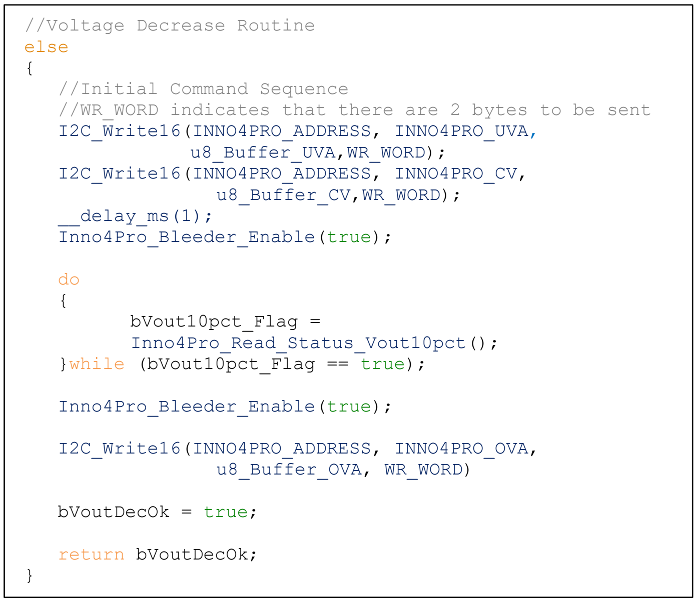


In the figure above, the CC set-point is programmed first. The UVA needs to be set to a value lower than the new CV set-point before updating the CV register. The BLEEDER should be turned on 1ms after the new CV value is written. The output voltage should be continually monitored while BLEEDER is on. After the output voltage reaches 110% of the CV set-point, the VOUT10PCT bit of the READ10 register will get cleared or set to 0. This bit notifies when the BLEEDER must be turned off to prevent unwanted power dissipation in the controller. The OVA set-point should be updated only after the bleeder is turned off in order to prevent the OVP.

Voltage Decrement Flowchart



Voltage Decrement Code Example



```
//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;
}
```

User of Timers

Fast VI Command Timer

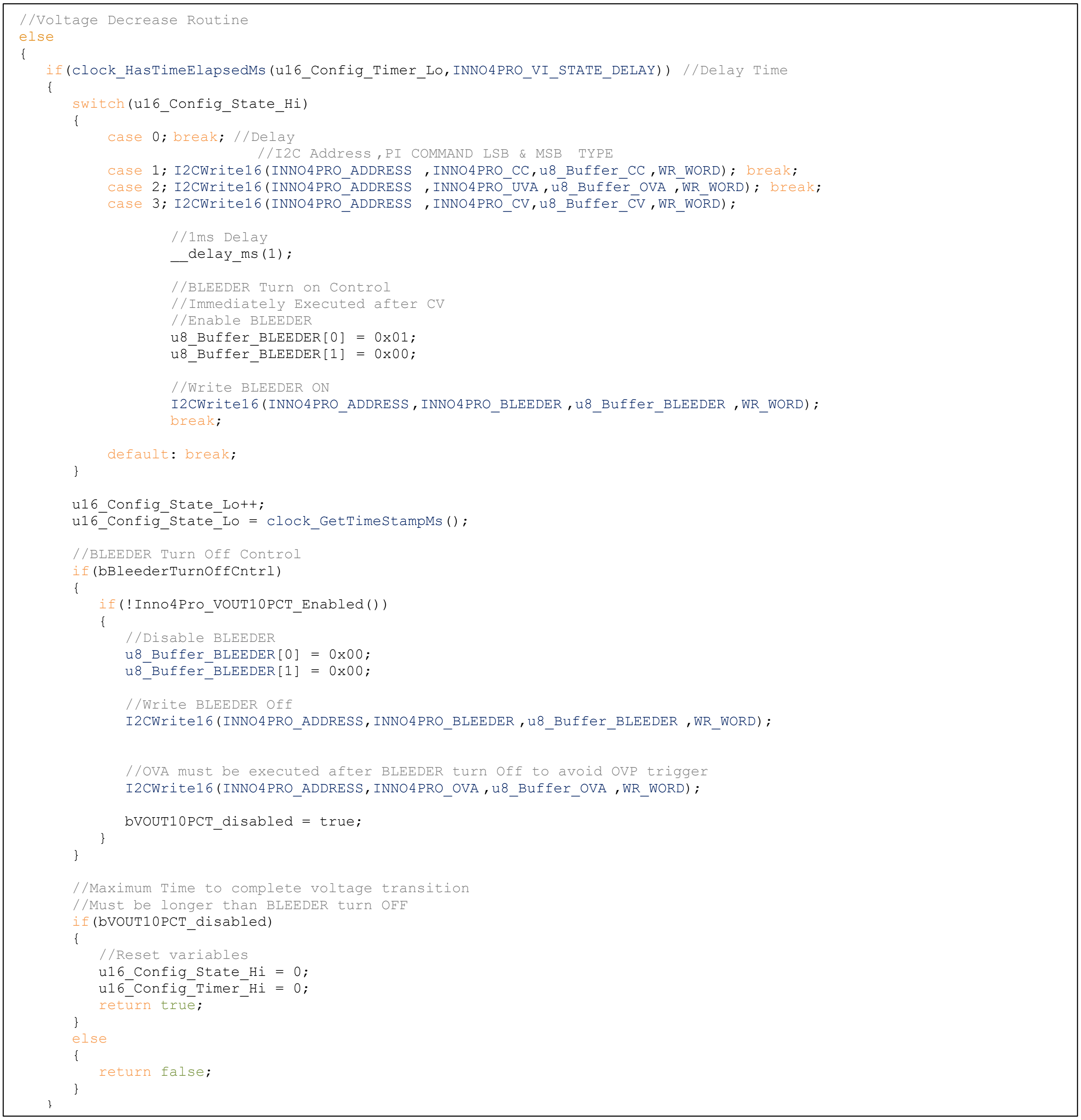
Consecutive CV and CC I^{2}C write commands cannot be sent faster than 10ms due to the FAST VI register. The 10ms update limit for consecutive commands to set the output Voltage/Current can be disabled when necessary. By default, the speed of the CV/CC update is enabled. The firmware implementation must have a **Timer Clock** that takes care of necessary timings and delays involved. This is usually a 1ms timer that runs on an interrupt

Voltage Increment with CV/CC Update Limit Enabled

The example below has a 12ms delay between CV and CC I^{2}C write commands

```
#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;
      }
   }

}
```

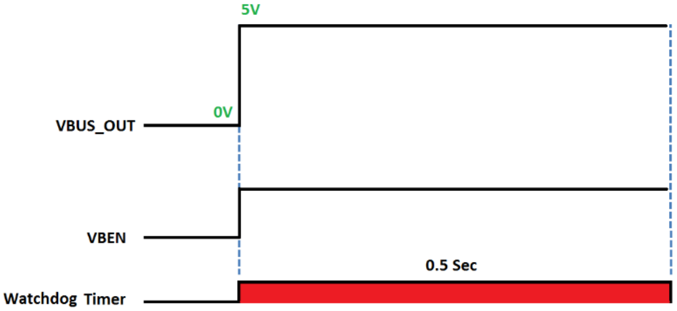
Voltage Decrement with CV/CC Update Limit Enabled



```
//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;
      }
   }
```

Watchdog Timer

The Watchdog Timer is enabled and set to 0.5 s by default. Within this time period, at least a single I²C transaction is required to prevent the InnoSwitch4-Pro from going back to the default reset.



Watchdog Code Example

```
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
 }
}
```

Telemetry / Read Back

The I²C Master can use the telemetry registers to read the user programmed registers, monitor and measure the output parameters, update the device protection features, and detect fault conditions.

Prior to using the Telemetry, I²C Read/Write drivers must be set according to the data sheet of the microcontroller being used.

System Ready Signal

The InnoSwitch4-Pro must indicate that it is ready to receive I²C commands prior to the start of any I²C transaction. This can be monitored by reading the status of the Reg_control_s bit on the READ10 register. An assertion to this bit means the InnoSwitch4-Pro is ready to communicate and accept commands.

System Ready Code Example

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

VOUT10% Signal

Whenever the output voltage transitions from a high to a low voltage set-point, InnoSwitch4-Pro asserts VOUT10PCT at the start of each transition. The device monitors the Output Voltage ADC and then clears the Reg_VOUT10PCT register when the output voltage settles to less than 10% of the set regulation threshold.

VOUT10% Code Example

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

I²C Read Back Code Example

Inno4Pro_Telemetry function is an API for reading the desired register address. It uses I²C_Read16 function which is an I²C driver created for the InnoSwitch4-Pro. When this function is used to read the value of a telemetry register, it returns a value of the MSB and then followed by the LSB. An example is that the value of the voltage DAC for a 5V reading is 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;
}
```

Read Bit Telemetry

Inno4Pro_Read_Bit function is an API for reading the desired bit of a telemetry register. This implementation also uses I²C_Read16 function. This function returns a value of either 1 or 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;
 }
}
```

Read Byte Telemetry

Inno4Pro_Read_Byte function is an API for reading the desired byte of a telemetry register. The user will have the option to select the either the MSB or 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);
 }
}
```

Read 2 Bits Telemetry

Inno4Pro_Read_2Bits function is an API for reading 2 bits in a byte of a telemetry register.

```
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;
}
```

Read Set-point and Threshold

Inno4Pro_Read_SetPoint function is used for reading voltage related read-backs.

```
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;
}
```

Read Voltage Code Example

The measured output voltage reading uses the Inno4Pro_Telemetry function to get the value of READ9. The default value of the output voltage is set to 5V during startup. When a read command is executed, we expect the value of READ9 to be 0x01 and 0xF4 for MSB and LSB respectively. This translates to a value of 500 which, based on the InnoSwitch4-Pro datasheet, translates to a 5V reading.

```
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);
}
```

Read Current Example

The value returned by both Inno4Pro_Read_Amps and Inno4Pro_Read_AmpsAverage is based on the Rsense value set in the firmware. Make sure that the Rsense value in the firmware is close to the value of the actual Rsense.

```
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));
}
```

Code Library

To simplify the technicalities on controlling the InnoSwitch4-Pro, a simple code library is provided as a reference. The library contains all the registers needed for controlling the device. These registers are organized as Command registers and Telemetry registers. Command registers are sent to the device for performance control and Telemetry registers are for reading back values. The computation macros are presented to aid in set-point calculations. The Register default values are also defined to simplify writing to the required registers at device initialization.

PIC16F18325 MCU Implementation

Implementation

Header Files Inclusion

The Library header files contain all of the function declarations and macro definitions. This must be included in the main page as shown.

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

InnoSwitch4-Pro Initialization

Before the continuous execution of the main code, the status of System Ready Signal is monitored to ensure the InnoSwitch4-Pro is ready to receive I^{2}C commands. Afterwards, initialization commands can be sent to the device to re-configure the default settings as needed. This initialization routine disables the watchdog timer and Fast VI Limit. UVL timer is also initialized to 64ms.

```
Inno4Pro_Initialization();

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

Control Functions Set-up

Updates the Output Voltage and Constant Current setting

These functions do the following:

- Follow a certain sequence of I2C commands in order to avoid inadvertent triggering of UV or OV faults
- Control the VOUT-pin strong bleeder when decreasing the voltage from High to Low setting
- Automatically updates the Over Voltage (OVA) and Under Voltage (UVA) settings:
    1. OVA is 124% of CV set-point
    2. UVA is fixed to 3V

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

Updates the Output Voltage without Bleeder Control

```
Inno4Pro_Write_Volts(Volts)
```

Sets the Constant Current Setting

```
Inno4Pro_Write_Amps(Amps)
```

Sets the Over Voltage Setting

```
Inno4Pro_Write_Overer_Volts(Value)
```

Sets the Under Voltage Setting

```
Inno4Pro_Write_Under_Volts(Volts)
```

Sets the Cable Drop Compensation Value

```
Inno4Pro_Write_Cable_Drop_Comp(Value)
```

Sets the Constant Output Power Threshold

```
Inno4Pro_Write_Volt_Peak(Value)
```

Used for Turning On or Off the Bus Voltage Switch

```
Inno4Pro_Vbus_Switch_Control(Value)
```

Used for Turning On or Off the VOUT pin strong bleeder

The BLEEDER must not be enabled for extended period of time to prevent excessive power dissipation in the controller

```
Inno4Pro_Bleeder_Enable(Value)
```

Telemetry Functions Setup

Use the Telemetry functions on the main loop to read the registers of InnoSwitch4-Pro.

Used for reading the desired Register Address

```
Inno4Pro_Telemetry(Register_Address)
```

Used for reading the specific bit of Telemetry Register

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

Tells when InnoSwitch4-Pro is ready to communicate and accept commands

```
Inno4Pro_Read_Status_SystemReady()
```

Returns the measured output voltage

```
Inno4Pro_Read_Volts()
```

Returns the measured output current

```
Inno4Pro_Read_Amps()
```

Returns the VOUT10PCT status information

VOUT10PCT status is used to disable the VOUT pin strong bleeder

```
Inno4Pro_Read_Status_Vout10pct()
```

Returns the VOUT2PCT status information

VOUT10PCT status is also used to disable the VOUT pin strong bleeder

```
Inno4Pro_Read_Status_Vout2pct()
```

Basic Code Example

This code example is to demonstrate the basic usage of InnoSwitch4-Pro Code Library.

- Initial commands are sent using the InnoSwitch4-Pro initialization routine.
- The main routine sets the output voltage to 5V and constant current to 6A.
- Cable Drop Compensation is programmed to 300mV.
- Constant power is knee voltage is set to 7V
- VBUS Switch is turned ON.

```
//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 Drivers

I²C drivers must be correctly configured depending on the microcontroller being used. This must be configured to meet the I²C packet format on the InnoSwitch4-Pro datasheet for read and write transactions. Every I²C transaction has at least a 150µs delay between commands

I²C Write Code Example

```
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;
}
```

I^{2}C Read Code Example

```
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 Implementation

Implementation

Header Files Inclusion

The library header files contain all of the function declarations and macro definitions. This must be included in the main page as shown.

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

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

Class Instance Creation

Construct a class instance to call the functions inside Inno4Pro_Application. Constructing a class instance of Inno4Pro_Rtc is optional.

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

InnoSwitch4-Pro Initialization

Before the continuous execution of the main code, the status of System Ready Signal is monitored to ensure the InnoSwitch4-Pro is ready to receive I^{2}C commands. Afterwards, initialization commands can be sent to the device to re-configure the default settings as needed. This initialization routine disables the watchdog timer and Fast VI Limit. UVL timer is also initialized to 64ms.

The 400kHz clock frequency for the I^{2}C communication is set-up on initialization.

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

Control Functions Set-up

Updates the Output Voltage and Constant Current setting

These functions do the following:

- Follow a certain sequence of I2C commands in order to avoid inadvertent triggering of UV or OV faults
- Control the VOUT-pin strong bleeder when decreasing the voltage from High to Low setting
- Automatically updates the Over Voltage (OVA) and Under Voltage (UVA) settings:
  3. OVA is 124% of CV set-point
  4. UVA is fixed to 3V

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

Updates the Output Voltage without Bleeder Control

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

Sets the Constant Current Setting

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

Sets the Over Voltage Setting

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

Sets the Under Voltage Setting

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

Sets the Cable Drop Compensation Value

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

Sets the Constant Output Power Threshold

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

Used for Turning On or Off the Bus Voltage Switch

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

Used for Turning On or Off the VOUT pin strong bleeder
The BLEEDER must not be enabled for extended period of time to prevent excessive power dissipation in the controller

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

Telemetry Functions Setup
Use the Telemetry functions on the main loop to read the registers of InnoSwitch4-Pro.

Used for reading the desired Register Address

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

Used for reading the specific bit of Telemetry Register

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

Tells when InnoSwitch4-Pro is ready to communicate and accept commands

```
Inno4ProApp.Inno4Pro_Read_Status_SystemReady()
```

Returns the measured output voltage

```
Inno4ProApp.Inno4Pro_Read_Volts()
```

Returns the measured output current

```
Inno4ProApp.Inno4Pro_Read_Amps()
```

Returns the VOUT10PCT status information
VOUT10PCT status is used to disable the VOUT pin strong bleeder

```
Inno4ProApp.Inno4Pro_Read_Status_Vout10pct()
```

Returns the VOUT2PCT status information
VOUT10PCT status is also used to disable the VOUT pin strong bleeder

```
Inno4ProApp.Inno4Pro_Read_Status_Vout2pct()
```

Basic Code Example

```
//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);
}
```

I^{2}C Drivers

The I^{2}C drivers must be correctly configured based on the Arduino Wire Library.

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

This must be configured to meet the I^{2}C packet format in the InnoSwitch4-Pro data sheet for write and read transactions. Every I^{2}C transaction must have at least a 150µs delay between commands.

I^{2}C Write Code Example

```
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();
}
```

I^{2}C Read Code Example

```
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));
}
```

Documentations

Configurations

This is the header file containing all the library macros and configuration for InnoSwitch4-Pro

Macros

Firmware Revision Macro

Defines the revision number of InnnoSwitch3-Pro Code Library to to track changes on each release.

Note:

Version Format: v00.00.00

```
#define INNO4PRO_FW_VERSION_MAJOR ‘0’,’1’
#define INNO4PRO_FW_VERSION_MINOR ‘0’.’2’
#define INNO4PRO_FW_VERSION_TEST  ‘0’.’0’
```

Saturation Macros

Used for setting limits to a certain parameter

```
#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)
```

Bit Manipulation Macros

Used for manipulating bits in a certain byte

```
#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 Macros

Defines the slave address of InnoSwitch4-Pro and I2C write sizes

Note: 0x0011000 (0x18) – 7-bit address scheme

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

InnoSwitch4-Pro Response and Timer Macros

Used for response and timer settings of registers

Note: use these macros for updating the PI Command register settings

```
#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 Register Address Assignments

Defines the command registers to control

```
#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
```

Telemetry (Read-back) Register Address Assignments

Defines the telemetry report-back registers to read

```
#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 Read Register Bit-Shift Count Macros

Defines the register bit-shift position used in reading register values

```
#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 Read Register Bit-Shift Count Macros

Defines the register bit-shift position used in reading register values

```
#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
```

Telemetry READ7 Bit Assignments

Defines the bit assignments on READ7 register

```
#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
```

Telemetry READ10 Bit Assignments

Defines the bit assignments on READ10 register

```
#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
```

Telemetry READ16 Bit Assignments

Defines the bit assignments on READ16 register

```
#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
```

Telemetry READ17 Bit Assignments

Defines the bit assignments on READ17 register

```
#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 Computation Macros

Defines the constants needed for the computation of individual registers

```
#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 Max and Min Configuration Settings

Defines the minimum and maximum values to be written to InnoSwitch4-Pro registers

```
#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)
```

Macro Definition

INNO4PRO_ADDRESS **0x18**
InnoSwitch4-Pro I^{2}C 7-bit slave address

WR_WORD **0x03**
Signals the I^{2}C write function to send 2 bytes of data (excluding the slaveAddress and registerAddress)

WR_BYTE **0x02**
Signals that the I^{2}C write function to send 1 byte of data (excluding the slaveAddress and registerAddress)

RD_MSB **1**
Signals the I^{2}C read function to read the MSB of the received data

RD_LSB **0**
Signals the I^{2}C read function to read the LSB of the received data

set_bit(address,bit) **(address |= (1<<bit))**
Sets the bit of the address to 1

clear_bit(address,bit) **(address &= ~ (1<<bit))**
Sets the bit of the address to 0

toggle_bit(address,bit) **(address ^= (1<<bit))**
Switches the bit of the address from 1 to 0 or 0 to 1

test_bit(address,bit) **(address & (1<<bit))**
Gets the value of a bit from the address

sig_minmax(sig,min,max) **((sig < min)? sig = min:**
(sig > max)? sig = max:0)
Limits the value of sig to between or equal to the min and max

sig_max(sig, max) **((sig > max) ? sig = max : 0)**
Limits the value of sig to below or equal to the max

sig_min(sig, min) **((sig < min) ? sig = min : 0)**
Limits the value of sig to above or equal to the min

INNO4PRO_READ_OV_FAULT_BITSHIFT **14**
Over-voltage fault bit-shift count for READ6

INNO4PRO_READ_UV_FAULT_BITSHIFT **12**
Under-voltage fault bit-shift count for READ6

INNO4PRO_READ_CCSC_OUTPUT_SHORT_BITSHIFT **10**
CCSC fault bit-shift count for READ6

INNO4PRO_READ_ISSC_SHORT_BITSHIFT **8**
ISSC fault bit-shift count for READ6

INNO4PRO_READ_UVL_TIMER_BITSHIFT **6**
UVL Timer bit-shift count for READ6

INNO4PRO_READ_WATCHDOG_TIMER_BITSHIFT **4**
Watchdog Timer bit-shift count for READ6

INNO4PRO_READ_CV_MODE_BITSHIFT **2**
CV Mode bit-shift count for READ6

INNO4PRO_READ_CV_MODE_TIMER_BITSHIFT **0**
CV Mode Timer bit-shift count for READ6

INNO4PRO_VBUS_ENABLE **3**
Series bus switch enable

INNO4PRO_VBUS_DISABLE_NORESET **1**
Series bus switch disable with no reset

INNO4PRO_VBUS_DISABLE **0**
Series bus switch disable with reset

INNO4PRO_BLEEDER_ENABLE_AUTODISABLE **3**
Active bleeder enable with auto-disable

INNO4PRO_BLEEDER_ENABLE **1**
Active bleeder enable

INNO4PRO_BLEEDER_DISABLE **0**
Active bleeder disable

INNO4PRO_LOAD_DISCHARGE_DISABLE **12**
Load discharge disable

INNO4PRO_LOAD_DISCHARGE_ENABLE **3**
Load discharge enable

INNO4PRO_LOAD_DISCHARGE_ENABLE_NORESET **2**
Load discharge enable with no-reset

INNO4PRO_TURN_OFF_PSU_ENABLE **true**
Latch-off device enable

INNO4PRO_TURN_OFF_PSU_DISABLE **false**
Latch-off device disable

INNO4PRO_FASTVI_UPDATE_LIMIT_DISABLE **true**
Fast VI command disable

INNO4PRO_FASTVI_UPDATE_LIMIT_ENABLE **false**
Fast VI command enable

INNO4PRO_OVL_FAULT_RESPONSE_NORESPONSE **0**
Over-voltage fault response set to no-response

INNO4PRO_OVL_FAULT_RESPONSE_LATCHOFF **1**
Over-voltage fault response set to latch-off

INNO4PRO_OVL_FAULT_RESPONSE_AUTORESTART **2**
Over-voltage fault response set to auto-restart

INNO4PRO_OVL_FAULT_RESPONSE_DISABLEOUTPUT **3**
Over-voltage fault response set to disable-output

INNO4PRO_UVL_FAULT_RESPONSE_NORESPONSE **0**
Under-voltage fault response set to no-response

INNO4PRO_UVL_FAULT_RESPONSE_LATCHOFF **1**
Under-voltage fault response set to latch-off

INNO4PRO_UVL_FAULT_RESPONSE_AUTORESTART **2**
Under-voltage fault response set to auto-restart

INNO4PRO_UVL_FAULT_RESPONSE_DISABLEOUTPUT **3**
Under-voltage fault response set to disable-output

INNO4PRO_CCSC_FAULT_RESPONSE_NORESPONSE **0**
CCSC fault response set to no-response

INNO4PRO_CCSC_FAULT_RESPONSE_LATCHOFF **1**
CCSC fault response set to latch-off

INNO4PRO_CCSC_FAULT_RESPONSE_AUTORESTART **2**
CCSC fault response set to auto-restart

INNO4PRO_CCSC_FAULT_RESPONSE_DISABLEOUTPUT **3**
CCSC fault response set to disable-output

INNO4PRO_ISSC_FAULT_RESPONSE_NORESPONSE **0**
ISSC fault response set to no-response

INNO4PRO_ISSC_FAULT_RESPONSE_LATCHOFF **1**
ISSC fault response set to latch-off

INNO4PRO_ISSC_FAULT_RESPONSE_AUTORESTART **2**
ISSC fault response set to auto-restart

INNO4PRO_ISSC_FAULT_RESPONSE_DISABLEOUTPUT **3**
ISSC fault response set to disable-output

INNO4PRO_ISSC_FREQ_THRESHOLD_60KHZ **0**
ISSC frequency threshold set to 60kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_30KHZ **1**
ISSC frequency threshold set to 30kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_90KHZ **2**
ISSC frequency threshold set to 90kHz

INNO4PRO_ISSC_FREQ_THRESHOLD_120KHZ **3**
ISSC frequency threshold set to 120kHz

INNO4PRO_ISSC_CC_THRESHOLD_16 **1**
ISSC CC threshold set to 16 LSB

INNO4PRO_ISSC_CC_THRESHOLD_32 **2**
ISSC CC threshold set to 32 LSB

INNO4PRO_ISSC_CC_THRESHOLD_48 **3**
ISSC CC threshold set to 48 LSB

INNO4PRO_ISSC_CC_THRESHOLD_64 **4**
ISSC CC threshold set to 64 LSB

INNO4PRO_ISSC_CC_THRESHOLD_80 **5**
ISSC CC threshold set to 80 LSB

INNO4PRO_ISSC_CC_THRESHOLD_96 **6**
ISSC CC threshold set to 96 LSB

INNO4PRO_ISSC_CC_THRESHOLD_112 **7**
ISSC CC threshold set to 112 LSB

INNO4PRO_UVL_FAULT_TIMER_8MS **0**
Under-voltage fault timer set to 8 ms

INNO4PRO_UVL_FAULT_TIMER_16MS **1**
Under-voltage fault timer set to 16 ms

INNO4PRO_UVL_FAULT_TIMER_32MS **2**
Under-voltage fault timer set to 32 ms

INNO4PRO_UVL_FAULT_TIMER_64MS **3**
Under-voltage fault timer set to 64 ms

INNO4PRO_WATCHDOG_TIMER_NOWATCHDOG **0**
Watchdog timer disable

INNO4PRO_WATCHDOG_TIMER_500MS **1**
Watchdog timer set to 0.5 s

INNO4PRO_WATCHDOG_TIMER_1000MS **2**
Watchdog timer set to 1.0 s

INNO4PRO_WATCHDOG_TIMER_2000MS **3**
Watchdog timer set to 2.0 s

INNO4PRO_CVOL_FAULT_RESPONSE_NORESPONSE **0**
CV Only mode fault response set to no-response

INNO4PRO_CVOL_FAULT_RESPONSE_LATCHOFF **1**
CV Only mode fault response set to latch-off

INNO4PRO_CVOL_FAULT_RESPONSE_AUTORESTART **2**
CV Only mode fault response set to auto-restart

INNO4PRO_CVOL_FAULT_RESPONSE_DISABLEOUTPUT **3**
CV Only mode fault response set to disable-output

INNO4PRO_CVOL_FAULT_TIMER_8MS **0**
CV Only mode fault timer set to 8 ms

INNO4PRO_CVOL_FAULT_TIMER_16MS 1
CV Only mode fault timer set to 16 ms

INNO4PRO_CVOL_FAULT_TIMER_32MS 2
CV Only mode fault timer set to 32 ms

INNO4PRO_CVOL_FAULT_TIMER_64MS 3
CV Only mode fault timer set to 64 ms

INNO4PRO_INTERRUPT_CONTROL_S_MASK 0x40
Interrupt mask for control secondary fault

INNO4PRO_INTERRUPT_BPS_CURR_LO_FAULT_MASK 0x20
Interrupt mask for BPS current latch-off fault

INNO4PRO_INTERRUPT_CVO_PKLOAD_TIMER_MASK 0x10
Interrupt mask for CVO mode peak load timer fault

INNO4PRO_INTERRUPT_ISSC_MASK 0x08
Interrupt mask for IS-pin short fault

INNO4PRO_INTERRUPT_CCSC_MASK 0x04
Interrupt mask for output short circuit fault

INNO4PRO_INTERRUPT_VOUT_UV_MASK 0x02
Interrupt mask for under-voltage fault

INNO4PRO_INTERRUPT_VOUT_OV_MASK 0x01
Interrupt mask for over-voltage fault

INNO4PRO_OTP_FAULT_HYST_40DEG 0
Over-temperature fault hysteresis set to 40 degrees Celsius

INNO4PRO_OTP_FAULT_HYST_60DEG 1
Over-temperature fault hysteresis set to 60 degrees Celsius

INNO4PRO_CVLOAD_DEFAULT 0x20
Constant voltage load default settings

INNO4PRO_CVLOAD_RECOMMENDED 0x80
Constant voltage load recommended settings

INNO4PRO_LOOPSPEED1_DEFAULT 0x281E
Loop speed 1 default settings

INNO4PRO_LOOPSPEED1_RECOMMENDED 0x140A
Loop speed 1 recommended settings

INNO4PRO_LOOPSPEED2_DEFAULT 0x08C8
Loop speed 2 default settings

INNO4PRO_LOOPSPEED2_RECOMMENDED 0x1F84
Loop speed 2 recommended settings

PI Command Register Address Assignments

INNO4PRO_VBEN 0x04
Series bus switch control register

INNO4PRO_BLEEDER 0x86
Active bleeder function register

INNO4PRO_VDIS 0x08
Load (VBUS) discharge register

INNO4PRO_TURN_OFF_PSU 0x8A
Latch-off device register

INNO4PRO_FAST_VI_CMD 0x8C
Speed of output CV/CC update register

INNO4PRO_CVO 0x0E
Constant Voltage Only, timer, and response register

INNO4PRO_CV 0x10
Output voltage register

INNO4PRO_OVA 0x92
Over-voltage threshold and response register

INNO4PRO_UVA 0x94
Under-voltage threshold, response, and timer register

INNO4PRO_CDC 0x16
Cable drop compensation register

INNO4PRO_CC 0x98
Constant current regulation register

INNO4PRO_VKP 0x1A
Constant output power knee voltage register

INNO4PRO_CCSC 0x20
Output short-circuit fault detection register

INNO4PRO_ISSC 0xA2
IS-pin short fault response, current, and frequency register

INNO4PRO_WATCHDOG_TIMER 0x26
Communication rate monitor register

INNO4PRO_INTERRUPT 0x2C
Interrupt mask manager

INNO4PRO_OTP 0xAE
Secondary over-temperature fault hysteresis register

INNO4PRO_CV_LOAD 0xB0
Constant voltage load register

INNO4PRO_LOOP_SPEED_1 0x32
Loop speed 1 register

INNO4PRO_LOOP_SPEED_2 0x34
Loop speed 2 register

INNO4PRO_VBUSSC 0xB6
Series bus switch short circuit fault

INNO4PRO_DCM 0xBA
DCM only register

INNO4PRO_SRZVS **0x3E**
SRFET zero voltage switching register

Telemetry Register Address Assignments

INNO4PRO_READ0 **0x00**
Revision ID telemetry register

INNO4PRO_READ1 **0x02**
Output voltage set-point telemetry register

INNO4PRO_READ2 **0x04**
Output current set-point telemetry register

INNO4PRO_READ3 **0x06**
Over-voltage threshold and response telemetry register

INNO4PRO_READ4 **0x08**
Under-voltage threshold, response, and timer telemetry register

INNO4PRO_READ5 **0x0A**
Constant Current and Constant Power telemetry register

INNO4PRO_READ6 **0x0C**
OVL, UVL, CCSC, ISSC, UVLTIMER, WDTIMER, CVMODE, and CVTimer fault and setting telemetry register

INNO4PRO_READ7 **0x0E**
VBUS switch enable, Min load, PSU off, Fast VI, CV mode, OTP hyst, and CDC telemetry registers

INNO4PRO_READ8 **0x10**
Measured output current telemetry register

INNO4PRO_READ9 **0x12**
Measured output voltage telemetry register

INNO4PRO_READ10 **0x14**
INTERRUPT, CONTROL_S, VDIS, HIGH_FSW, OTP, VOUT2PCT, VOUT10PCT, ISSC, CCSC, VOUT_UV, & VOUT_OV telemetry register

INNO4PRO_READ11 **0x16**
OMF Flag

INNO4PRO_READ12 **0x18**
Average output current telemetry registers

INNO4PRO_READ13 **0x1A**
Average output voltage telemetry register

INNO4PRO_READ14 **0x1C**
Voltage DAC telemetry register

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, and BPS_OV telemetry register

INNO4PRO_READ17 **0x22**
Interrupts telemetry register

INNO4PRO_READ_LOOP_SPEED_1 **0x26**
Loop speed 1 telemetry register

INNO4PRO_READ_LOOP_SPEED_2 **0x28**
Loop speed 2 telemetry register

Telemetry READ7 Bit Assignments

READ7_Reg_VBEN **14**
VBUS switch bit

READ7_Reg_BLEEDER **13**
Minimum load bit

READ7_Reg_PSUOFF **12**
Turn PSU off bit

READ7_Reg_FSTVIC **11**
Fast VI command bit

READ7_Reg_CVO **10**
Constant Voltage Only mode bit

READ7_Reg_OTP **9**
Over-temperature protection hysteresis bit

Telemetry READ10 Bit Assignments

READ10_Reg_INTERRUPT_EN **15**
Interrupt enable bit

READ10_Reg_CONTROL_S **14**
System ready signal bit

READ10_Reg_VDIS **13**
Output discharge bit

READ10_Reg_HIGH_FSW **12**
High switching frequency bit

READ10_Reg_OTP **9**
Over-temperature protection bit

READ10_Reg_VOUT2PCT **5**
2% BLEEDER enabled bit

READ10_Reg_VOUT10PCT **4**
10% BLEEDER enabled bit

READ10_Reg_ISSC **3**
IS-pin short-circuit detected bit

READ10_Reg_CCSC **2**
Output short-circuit detected bit

READ10_Reg_VOUT_UV **1**
Output voltage UV fault bit

READ10_Reg_VOUT_OV **0**

Output voltage OV fault bit

Telemetry READ16 Bit Assignments

READ16_Reg_ar_CV	**15**	CVO mode auto-restart bit
READ16_Reg_ar_ISSC	**12**	ISSC auto-restart bit
READ16_Reg_ar_CCSC	**11**	CCSC auto-restart bit
READ16_Reg_ar_VOUT_OV	**10**	Output voltage OV auto-restart
READ16_Reg_ar_VOUT_UV	**9**	Output voltage UV auto-restart
READ16_Reg_LO	**7**	Latch-off occurred bit
READ16_Reg_Lo_CVO	**6**	CVO mode auto-restart bit
READ16_Reg_PSUOFF	**5**	PSU off command received bit
READ16_Reg_Lo_ISSC	**4**	ISSC latch-off bit
READ16_Reg_Lo_VOUT_OV	**2**	Output voltage OV latch-off bit
READ16_Reg_Lo_VOUT_UV	**1**	Output voltage UV latch-off bit
READ16_Reg_BPS_OV	**0**	BPS-pin latch-off bit

Telemetry READ17 Bit Assignments

READ17_Reg_CONTROL_S_MASK	**15**	Control_S interrupt mask
READ17_Reg_LO_Fault_MASK	**14**	Latch-off interrupt mask
READ17_Reg_CVO_MASK	**13**	CVO interrupt mask
READ17_Reg_ISSC_MASK	**12**	ISSC interrupt mask
READ17_Reg_CCSC_MASK	**11**	CCSC interrupt mask
READ17_Reg_VOUT_UV_MASK	**10**	Output voltage UV interrupt mask
READ17_Reg_VOUT_OV_MASK	**9**	Output voltage OV interrupt mask
READ17_Reg_OMF	**8**	OMF flag interrupt mask
READ17_Reg_VBUSSC	**7**	VBUS-pin short-circuit mask
READ17_Reg_CONTROL_S_STATUS	**6**	Control_S interrupt status
READ17_Reg_LO_FAULT_STATUS	**5**	Latch-off fault interrupt status
READ17_Reg_CVO_STATUS	**4**	CVO interrupt status
READ17_Reg_ISSC_STATUS	**3**	ISSC interrupt status
READ17_Reg_CCSC_STATUS	**2**	CCSC interrupt status
READ17_Reg_VOUT_UV_STATUS	**1**	Output voltage UV interrupt status
READ17_Reg_VOUT_OV_STATUS	**0**	Output voltage OV interrupt status

InnoSwitch4-Pro Computation Macros

INNO4PRO_RSENSE	**(float)(9)**	Current resistor in mΩ
INNO4PRO_FULL_RANGE_RSENSE_VOLTAGE	**(float)(32)**	Current sense resistor full range voltage in mV
INNO4PRO_ADC_FULL_RANGE	**(float)(192)**	Analog to digital converter full range
INNO4PRO_CC_SET_PT_FACTOR	**(float)(6)**	CC set-point factor: (192 / 32)
INNO4PRO_CC_SET_PT_MULT	**(float)** **(INNO4PRO_RSENSE *** **INNO4PRO_CC_SET_POINT_FACTOR)**	CC set-point computation: (ADC voltage * Rsense * 192 / 32)
INNO4PRO_CV_SET_PT_MULT	**(float)(100)**	Multiplier for output voltage step size of 10mV/LSB
INNO4PRO_OV_PERCENTAGE_MULT	**(float)(1.15)**	OV multiplier set to 115% of CV
INNO4PRO_UV_PERCENTAGE_MULT	**(float)(0.85)**	UV multiplier set to 85% of CV
INNO4PRO_OV_SET_PT_MULT	**(float)(10)**	Multiplier for over-voltage write step size of 100mV/LSB
INNO4PRO_UV_SET_PT_MULT	**(float)(10)**	Multiplier for under-voltage write step size of 100mV/LSB

INNO4PRO_OV_READ_MULT (float)(100)
Multiplier for over-voltage read step size of 100mV/LSB

INNO4PRO_UV_READ_MULT (float)(100)
Multiplier for under-voltage read step size of 100mV/LSB

INNO4PRO_CDC_SET_PT_DIV (float)(0.02)
Divider for cable drop compensation step size of 50mV/LSB

INNO4PRO_CDC_SET_PT_MULT (float)(50)
Multiplier for cable drop compensation step size of 50mV/LSB

INNO4PRO_VKP_SET_PT_MULT (float)(10)
Multiplier for constant output power knee voltage step size of 100mV/LSB

InnoSwitch4-Pro Maximum and Minimum Limits

INNO4PRO_CV_SET_PT_MAX (2400)
Maximum output voltage set-point at 10mV/LSB

INNO4PRO_CV_SET_PT_MIN (300)
Minimum output voltage set-point at 10mV/LSB

INNO4PRO_OV_SET_PT_MAX (250)
Maximum over-voltage set-point at 100mV/LSB

INNO4PRO_OV_SET_PT_MIN (33)
Minimum over-voltage set-point at 100mV/LSB

INNO4PRO_UV_SET_PT_MAX (240)
Maximum under-voltage set-point at 100mV/LSB

INNO4PRO_UV_SET_PT_MIN (27)
Minimum under-voltage set-point at 100mV/LSB

INNO4PRO_CDC_SET_PT_MAX (12)
Maximum cable drop compensation set-point

INNO4PRO_CDC_SET_PT_MIN (0)
Minimum cable drop compensation set-point

INNO4PRO_CC_SET_PT_MAX (192)
Maximum constant current set-point

INNO4PRO_CC_SET_PT_MIN (25)
Minimum constant current set-point

INNO4PRO_VKP_SET_PT_MAX (240)
Maximum constant output power knee voltage set-point at 100mV/LSB

INNO4PRO_VKP_SET_PT_MIN (53)
Minimum constant output power knee voltage set-point at 100mV/LSB

INNO4PRO_VBEN_SET_PT_MAX (3)
Maximum VBEN set-point

INNO4PRO_VBEN_SET_PT_MIN (0)
Minimum VBEN set-point

INNO4PRO_RSENSE_MAX_LIMIT (20)
Maximum Rsense value in mΩ

INNO4PRO_RSENSE_MIN_LIMIT (1)
Minimum Rsense value in mΩ

INNO4PRO_CV_MAX_LIMIT (24)
Maximum output voltage in V

INNO4PRO_CV_MIN_LIMIT (3)
Minimum output voltage in V

INNO4PRO_OV_MAX_LIMIT (25)
Maximum over-voltage set-point in V

INNO4PRO_OV_MIN_LIMIT (3.3)
Minimum over-voltage set-point in V

INNO4PRO_UV_MAX_LIMIT (24)
Maximum under-voltage set-point in V

INNO4PRO_UV_MIN_LIMIT (2.7)
Minimum under-voltage set-point in V

INNO4PRO_CDC_MAX_LIMIT (600)
Maximum cable drop compensation set-point in mV

INNO4PRO_CDC_MIN_LIMIT (0)
Minimum cable drop compensation set-point in mV

INNO4PRO_CC_MAX_LIMIT (192)
Maximum constant current limit

INNO4PRO_CC_MIN_LIMIT (25)
Minimum constant current limit

INNO4PRO_VKP_MAX_LIMIT (24)
Maximum constant output power knee voltage in V

INNO4PRO_VKP_MIN_LIMIT (5.3)
Minimum constant output power knee voltage in V

I2C Drivers

This source file contains the drivers APIs for I^{2}C

I^{2}C Functions

int I2C_Write16 (uint16_t slaveAddress, uint8_t dataAddress, uint8_t *dataBuffer, uint8_t buflen)
Handles one I^{2}C master write transaction with the supplied parameters

uint16_t I2C_Read16 (uint16_t slaveAddress, uint8_t dataAddress)
Handles one I^{2}C master read transaction with the supplied parameters

uint8_t I2C_Read8 (uint16_t slaveAddress, uint8_t dataAddress)
Handles one I^{2}C master read transaction with the supplied parameters

Function Documentation

int I2C_Write16 (uint16_t slaveAddress, uint8_t dataAddress, uint8_t *dataBuffer, uint8_t buflen)
Handles one i2c master write transaction with the supplied parameters. This writes 1 to 2 bytes of data to the slave device

Parameters:
- slaveAddress: The address of the I^{2}C device to be accessed. Uses 7-bit address scheme.
- dataAddress: The register address to be accessed
- dataBuffer: A pointer to the block of data to be sent
- buflen: The length of the data block to be sent

Returns
- None

uint16_t I2C_Read16 (uint16_t slaveAddress, uint8_t dataAddress)
Handles one I^{2}C master read transaction with the supplied parameters. This reads 2 bytes of data from the slave device

Parameters:
- slaveAddress: The address of the I^{2}C device to be accessed. Uses 7-bit address scheme.
- dataAddress: The register address to be accessed

Returns
- Merged LSB and MSB from the slave device

uint8_t I2C_Read8 (uint16_t slaveAddress, uint8_t dataAddress)
Handles one I^{2}C master read transaction with the supplied parameters. This reads 2 bytes of data from the slave device

Parameters:
- slaveAddress: The address of the I^{2}C device to be accessed. Uses 7-bit address scheme.
- dataAddress: The register address to be accessed

Returns
- 1 byte from the slave device

Clock Driver

This is the source file containing the driver APIs for clock signals.

Functions

void clock_TimeUpdate (void)
Used for calculating

uint16_t clock_GetElapsedTimeUs (uint16_t u16TimeStamp)
Used for calculating the elapsed time in micro-seconds

uint16_t clock_GetElapsedTimeMs (uint16_t u16TimeStamp)
Used for calculating the elapsed time in milli-seconds

uint16_t clock_GetElapsedTimeSec (uint16_t u16TimeStamp)
Used for calculating the elapsed time in seconds

uint16_t clock_GetTimeStampUs (void)
Get the current timestamp in micro-seconds

uint16_t clock_GetTimeStampMs (void)
Get the current timestamp in milli-seconds

uint16_t clock_GetTimeStampSec (void)
Get the current timestamp in seconds

bool clock_HasTimeElapsedUs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
Used for generating micro-seconds delay

bool clock_HasTimeElapsedMs (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
Used for generating milli-seconds delay

bool clock_HasTimeElapsedSec (uint16_t u16TimeStamp, uint16_t u16TimeDurationCheck)
Used for generating seconds delay

Function Documentation

void clock_TimeUpdate (void)
This a simple clock interface that must run on an interrupt to generate clock signal (milli seconds, seconds) which are used to create delays based on the system clock

Note:
- This function must run on a 200µs interrupt

Parameters:
- None

Returns
- None

uint16_t clock_GetElapsedTimeUs (uint16_t u16TimeStamp)
Used for calculating the elapsed time in micro-seconds

Parameters:
- u16TimeStamp: Micro-second time stamp variable

Returns
- Returns time that has elapsed since u16TimeStamp

uint16_t clock_GetElapsedTimeMs (uint16_t u16TimeStamp)
Used for calculating the elapsed time in milli-seconds

Parameters:
- u16Timestamp: Milli-second time stamp variable

Returns
- Returns time that has elapsed since u16Timestamp

uint16_t clock_GetElapsedTimeSec (uint16_t u16TimeStamp)
Used for calculating the elapsed time in seconds

Parameters:
- u16Timestamp: Second time stamp variable

Returns
- Returns time that has elapsed since u16Timestamp

uint16_t clock_GetTimeStampUs (void)
Get the current timestamp in micro-seconds

Parameters:
- None

Returns
- Current timestamp in µs

uint16_t clock_GetTimeStampMs (void)
Get the current timestamp in milli-seconds

Parameters:
- None

Returns
- Current timestamp in ms

uint16_t clock_GetTimeStampSec (void)
Get the current timestamp in seconds

Parameters:

- None

Returns

- Current timestamp in s

bool clock_HasTimeElapsedUs (uint16_t u16TimeStamp,
uint16_t u16TimeDurationCheck)
Used for generating micro-seconds delay

Parameters:

- u16TimeStamp: Micro-second time stamp variable
- u16TimeDurationCheck: Duration to compare against the u16TimeStamp

Returns

- 1 after the delay time has elapsed

bool clock_HasTimeElapsedMs (uint16_t u16TimeStamp,
uint16_t u16TimeDurationCheck)
Used for generating milli-seconds delay

Parameters:

- u16TimeStamp: Milli-second time stamp variable
- u16TimeDurationCheck: Duration to compare against the u16TimeStamp

Returns

- 1 after the delay time has elapsed

bool clock_HasTimeElapsedSec (uint16_t u16TimeStamp,
uint16_t u16TimeDurationCheck)
Used for generating seconds delay

Parameters:

- u16TimeStamp: Second time stamp variable
- u16TimeDurationCheck: Duration to compare against the u16TimeStamp

Returns

- 1 after the delay time has elapsed’

InnoSwitch4-Pro

Functions

Setter Functions
Functions for Setting the values of the Register Variables

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 Functions
Functions for Getting the contents of the Register Variables

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)

Computation Functions
Threshold calculations and adjustment range for specific registers

float Inno4Pro_Compute_CV (float fSetPt)
InnoSwitch4-Pro computation for Output Voltage (CV)

float Inno4Pro_Compute_OV (float fSetPt)
InnoSwitch4-Pro computation for Over-Voltage Threshold (OVA)

float Inno4Pro_Compute_UV (float fSetPt)
InnoSwitch4-Pro computation for Under-Voltage Threshold (UVA)

float Inno4Pro_Compute_CDC (float fSetPt)
InnoSwitch4-Pro computation for Cable Drop Compensation (CDC)

float Inno4Pro_Compute_CC (float fSetPt)
InnoSwitch4-Pro computation for Constant Current Regulation (CC)

float Inno4Pro_Compute_VKP (float fSetPt)
InnoSwitch4-Pro computation for Output Power Knee Voltage (VKP)

float Inno4Pro_Compute_VBEN (float fSetPt)
InnoSwitch4-Pro computation for Series Bus Switch Control (VBEN)

Buffer Related Functions
Buffer and Parity Handling

void InnoProBase_Encode_Buffer (uint16_t u16Temp,
uint8_t *u8WriteBuffer)
Handles conversion of input data to hexadecimal LSB and MSB without parity bits.

bool InnoProBase _OddParity (uint8_t u8OddParity)
Handles odd parity bit detection.

void InnoProBase _Format_Buffer (uint16_t u16Temp,
uint8_t *u8WriteBuffer)
Handles conversion of input data to hexadecimal LSB and MSB with parity bits.

bool InnoProBase _AddOddParity (uint16_t u16Temp)
Adds parity bit to the LSB

void Inno4Pro_Process_Volt_Buffers (void)
Handles preparation for values to be written on InnoSwitch4-Pro voltage related registers.

Request Detection Functions
Stores the Previous value of the Register Variables

bool InnoProBase_Detect_Voltage_Request (void)
Checks if there's a new Voltage Request

bool InnoProBase_Detect_Current_Request (void)
Checks if there's a new Current Request

API Write Functions
Application Programming Interface to control InnoSwitch4-Pro

void Inno4Pro_Initialization (void)
Handles all common I2C configurations to be written to InnoSwitch4-Pro as initialization

void Inno4Pro_Vbus_Switch_Control (bool bEnableVben)
Vbus Switch Control (VBEN Control)

void Inno4Pro_Vbus_Switch_Control_NoReset (uint8_t u8EnableVben)
Vbus Switch Control with No-reset

void Inno4Pro_Bleeder_Enable (bool bEnable)
Handles Bleeder setting

void Inno4Pro_Load_Discharge (bool bEnable)
Activates Vbus Load Discharge

void Inno4Pro_TurnOff_PSU (bool bEnable)
Turns off the power supply

void Inno4Pro_FastVI_Disable (bool bDisable)
Sets CV and CC commands speed limit

void Inno4Pro_CVOnlyMode_Enable (bool bEnable, uint16_t u16Response, uint16_t u16Timer)
Sets constant voltage only mode

void Inno4Pro_Write_Volts (float fSetPtCV)
Output Voltage Control without Bleeder control

void Inno4Pro_Write_Over_Volts (float fSetPtOVA, uint16_t u16OVL)
Writes over-voltage protection settings

void Inno4Pro_Write_Under_Volts (float fSetPtUVA, uint16_t u16Uv_FaultResp, uint16_t u16Uv_timer)
Writes under-voltage protection settings

void Inno4Pro_Write_Cable_Drop_Comp (float fSetPtCDC)
Writes Cable Drop Compensation (CDC) settings

void Inno4Pro_Write_Amps (float fSetPtCC)
Constant Current (CC) control

void Inno4Pro_Write_Volt_Peak (float fSetPtVpk)
Constant Output Power Voltage Threshold (VKP) control

void Inno4Pro_Write_CCSC_Fault_Response (uint16_t u16Response)
Writes output shirt-circuit fault response setting

void Inno4Pro_Write_ISSC_Fault_Response (uint16_t u16Response, uint16_t u16Frequency, uint16_t u16CC)
Writes IS-pin short fault response setting

void Inno4Pro_Write_Watchdog_Timer (uint16_t u16Timer)
Writes watchdog timer setting

void Inno4Pro_Write_Interupt_Mask (uint16_t u16IntMask)
Writes interrupt mask setting

void Inno4Pro_Write_OTP_Hysteresis (uint16_t u16Otp)
Sets over-temperature hysteresis

void Inno4Pro_Write_CV_Load (uint16_t u16Load)
Writes Constant Voltage Load setting

void Inno4Pro_Write_Loop_Speed1 (uint16_t u16LoopSpeed)
Writes loop speed 1 settings

void Inno4Pro_Write_Loop_Speed2 (uint16_t u16LoopSpeed)
Writes loop speed 2 settings

bool Inno4Pro_Write_VI (float fSetPtCV, float fSetPtCC)
Output voltage control with bleeder control and constant current (CC) Control

void Inno4Pro_VBUSSC (uint16_t u16VSSC_response, uint16_t u16Vsamples, uint16_t u16CCThreshold)
Defines how the device will response to a series BUS switch short-circuit fault.

void Inno4Pro_DCMonly (bool bEnable)
Enables or disables the feature to limit the switching cycle requests from the secondary to the primary such that the converter is always operating in Discontinuous Conduction Mode

bool Inno4Pro_Process_Voltage (bool bVoltIncrease)
Handles command sequences for voltage increment/decrement

Common API Telemetry Functions
These functions are used as base for the main API Read functions

uint16_t InnoProBase_Telemetry (uint8_t ReadBack_Address)
Handles InnoSwitch4-Pro common I2C read back telemetry

bool InnoProBase_Read_Bit (uint8_t ReadBack_Address, uint8_t Bit)

Handles InnoSwitch4-Pro I2C read bit

uint8_t InnoProBase_Read_Byte (uint8_t ReadBack_Address, bool bHighByte)
Handles InnoSwitch4-Pro I2C read byte

uint8_t InnoProBase_Read_2Bits (uint8_t ReadBack_Address, uint8_t u8ShiftCnt)
Handles InnoSwitch4-Pro I2C read 2 bits

float InnoProBase_Read_SetPoint (uint16_t ReadBack_Address, float fMultiplier)
Handles InnoSwitch4-Pro I2C set-point and threshold

Read1 - Main API Telemetry Functions
Telemetry API for Read1

float Inno4Pro_Read_CV_SetPoint (void)
Reads the telemetry register READ1 - Output voltage set-point

Read2 - Main API Telemetry Functions
Telemetry API for Read2

float Inno4Pro_Read_Output_CC_SetPoint (void)
Reads the telemetry register READ2 – Output current set-point

Read3 - Main API Telemetry Functions
Telemetry API for Read3

float Inno4Pro_Read_OV_Threshold (void)
Reads the Telemetry register READ3 – Over-voltage threshold

Read4 - Main API Telemetry Functions
Telemetry API for READ 4

float Inno4Pro_Read_UV_Threshold (void)
Reads the Telemetry register READ 4 – Under-voltage threshold

Read5 - Main API Telemetry Functions
Telemetry API for Read4

float Inno4Pro_Read_CC_SetPoint (void)
Reads the Telemetry register READ 5 – Constant current set-point

float Inno4Pro_Read_CP_Threshold (void)
Reads the Telemetry register READ 5 – Constant power set-point

Read6 - Main API Telemetry Functions
Telemetry API for Read6

uint8_t Inno4Pro_Read_OV_Fault_Response (void)
Reads the telemetry register READ6 – Over-voltage fault response

uint8_t Inno4Pro_Read_UV_Fault_Response (void)
Reads the telemetry register READ6 – Under-voltage fault response

uint8_t Inno4Pro_Read_OutputSckt_Fault_Response (void)
Reads the telemetry register READ6 – Output short-circuit fault response

uint8_t Inno4Pro_Read_IsPinShort_Fault_Response (void)
Reads the telemetry register READ6 - IS-pin short fault response

uint8_t Inno4Pro_Read_UV_Fault_Timer (void)
Reads the telemetry register READ6 – Under-voltage timer

uint8_t Inno4Pro_Read_Watchdog_Timer (void)
Reads the telemetry register READ6 - Watchdog timer

uint8_t Inno4Pro_Read_CvMode_Fault_Response (void)
Reads the telemetry register READ6 - Constant Voltage Mode fault response

uint8_t Inno4Pro_Read_CvMode_Timer (void)
Reads the telemetry register READ6 - Constant Voltage Mode Timer

Read7 - Main API Telemetry Functions
Telemetry API for Read7

bool Inno4Pro_Read_VbusSwitch (void)
Reads bit 14 on telemetry register READ7 - VBUS Switch Enable

bool Inno4Pro_Read_Bleeder (void)
Reads bit 13 on telemetry register READ7 - Minimum Load (Bleeder)

bool Inno4Pro_Read_PsuOff (void)
Reads bit 12 on telemetry register READ7 - Turn PSU off (Latch Off Device)

bool Inno4Pro_Read_FastVI (void)
Reads bit 11 on telemetry register READ7 - Fast VI Commands

bool Inno4Pro_Read_CvoMode (void)
Reads bit 10 on telemetry register READ7 - Constant-Voltage Mode Only

bool Inno4Pro_Read_OtpFaultHyst (void)
Reads bit 9 on telemetry register READ7 - Over-Temperature Protection

float Inno4Pro_Read_Cable_Drop_Comp (void)
Reads the telemetry register READ7 - Cable Drop Compensation

Read8 - Main API Telemetry Functions
Telemetry API for Read8

float Inno3Pro_Read_Amps (void)
Reads the telemetry register READ8 - Measured output current

Read9 - Main API Telemetry Functions
Telemetry API for Read9

float Inno3Pro_Read_Volts (void)
Reads the telemetry register READ9 - Measured output voltage

Read10 - Main API Telemetry Functions
Telemetry API for Read10

bool Inno4Pro_Read_Status_InterruptEnable (void)
Reads bit 15 on telemetry register READ10 - Interrupt Enable

bool Inno4Pro_Read_Status_SystemReady (void)
Reads bit 14 on telemetry register READ10 - System Ready Signal

bool Inno4Pro_Read_Status_OutputDischarge (void)
Reads bit 13 on telemetry register READ10 - Output Discharge

bool Inno4Pro_Read_Status_HighSwitchFreq (void)
Reads bit 12 on telemetry register READ10 - Switching Frequency High

bool Inno4Pro_Read_Status_OtpFault (void)
Reads bit 9 on telemetry register READ10 - Over-Temperature Protection

bool Inno4Pro_Read_Status_Vout2pct (void)
Reads bit 5 on telemetry register READ10 - 2% Bleeder Enabled

bool Inno4Pro_Read_Status_Vout10pct (void)
Reads bit 4 on telemetry register READ10 - VOUTADC > 1.1*Vout

bool Inno4Pro_Read_Status_IsPinShort (void)
Reads bit 3 on telemetry register READ10 - IS-pin Short Circuit Detected

bool Inno4Pro_Read_Status_OutputShorCkt (void)
Reads bit 2 on telemetry register READ10 – Output Short Circuit Detected

bool Inno4Pro_Read_Status_UV_Fault (void)
Reads bit 1 on telemetry register READ10 - Output Voltage UV Fault Comparator

bool Inno4Pro_Read_Status_OV_Fault (void)
Reads bit 0 on telemetry register READ10 - Output Voltage OV Fault Comparator

Read12 - Main API Telemetry Functions
Telemetry API for Read12

float Inno4Pro_Read_AmpsAverage (void)
Reads the Telemetry register READ12 - Average output current

Read13 - Main API Telemetry Functions
Telemetry API for Read14

float Inno4Pro_Read_VoltsAverage (void)
Reads the Telemetry register READ13 - Average output voltage

Read14 - Main API Telemetry Functions
Telemetry API for Read14

float Inno3Pro_Read_Voltage_DAC (void)
Reads the Telemetry register READ14 - Voltage DAC

Read16 - Main API Telemetry Functions
Telemetry API for Read16

bool Inno4Pro_Read_Status_CvoMode_AR(void)
Reads bit 15 on telemetry register READ16 - CVO Mode auto-restart(AR)

bool Inno4Pro_Read_Status_IsPinShort_AR(void)
Reads bit 12 on telemetry register READ16 – Auto-restart indicator due to Is-pin short

bool Inno4Pro_Read_Status_OutputShortCkt_AR(void)
Reads bit 11 on telemetry register READ16 – Auto-restart indicator due to Output short-circuit

bool Inno4Pro_Read_Status_OV_AR(void)
Reads bit 10 on telemetry register READ16 - Auto-restart indicator due to over-voltage fault

bool Inno4Pro_Read_Status_UV_AR(void)
Reads bit 9 on telemetry register READ16 - Auto-restart indicator due to under-voltage fault

bool Inno4Pro_Read_Status_LatchOff(void)
Reads bit 7 on telemetry register READ16 – Latch-off (LO) occurred

bool Inno4Pro_Read_Status_CvoMode_LO(void)
Reads bit 6 on telemetry register READ16 – CVO Mode latch-off(LO)

bool Inno4Pro_Read_Status_PsuOffCmd(void)
Reads bit 5 on telemetry register READ16 – PSU turn off command received

bool Inno4Pro_Read_Status_IsPinShort_LO(void)
Reads bit 4 on telemetry register READ16 – Latch-off indicator due to Is-pin short

bool Inno4Pro_Read_Status_OV_LO(void)
Reads bit 2 on telemetry register READ16 - Latch-off indicator due to over-voltage fault

bool Inno4Pro_Read_Status_UV_LO(void)
Reads bit 1 on telemetry register READ16 - Latch-off indicator due to under-voltage fault

Read17 - Main API Telemetry Functions
Telemetry API for Read17

bool Inno4Pro_Read_Interrupt_Mask_CntrlSecondary (void)
Reads bit 14 on telemetry register READ17 - Interrupt Mask Control Secondary

bool Inno4Pro_Read_Interrupt_Mask_BpsCurrentLo (void)
Reads bit 13 on telemetry register READ17 - Interrupt Mask BPS Current latch-off

bool Inno4Pro_Read_Interrupt_Mask_CvoPkLoadTimer (void)
Reads bit 12 on telemetry register READ17 - Interrupt Mask BPS Current latch-off

bool Inno4Pro_Read_Interrupt_Mask_IsPinShort (void)
Reads bit 11 on telemetry register READ17 - Interrupt Mask IS-pin short

bool Inno4Pro_Read_Interrupt_Mask_OutputShortCkt (void)
Reads bit 10 on telemetry register READ17 - Interrupt Mask Output short-circuit

bool Inno4Pro_Read_Interrupt_Mask_UV (void)
Reads bit 9 on telemetry register READ17 - Interrupt Mask Vout under-voltage(UV)

bool Inno4Pro_Read_Interrupt_Mask_OV (void)
Reads bit 8 on telemetry register READ17 - Interrupt Mask Vout over-voltage(OV)

bool Inno4Pro_Read_Interrupt_Stat_OMF(void)

Reads bit 7 on telemetry register READ17 – Interrupt Status Operating mode has changed

bool Inno4Pro_Read_Interrupt_Stat_VBUSSC (void)
Reads bit 7 on telemetry register READ17 – Interrupt Status Vbus short-circuit

bool Inno4Pro_Read_Interrupt_Stat_CntrlSecondary (void)
Reads bit 6 on telemetry register READ17 - Interrupt Status Control Secondary

bool Inno4Pro_Read_Interrupt_Stat_BpsCurrentLo (void)
Reads bit 6 on telemetry register READ17 - Interrupt Status for Control Secondary

bool Inno4Pro_Read_Interrupt_Stat_CvoPkLoadTimer (void)
Reads bit 5 on telemetry register READ17 - Interrupt Status for CVO Mode Peak load timer

bool Inno4Pro_Read_Interrupt_Stat_IsPinShort (void)
Reads bit 3 on telemetry register READ17 - Interrupt Status for Status IS-pin short

bool Inno4Pro_Read_Interrupt_Stat_OutputShortCkt(void)
Reads bit 2 on telemetry register READ17 - Interrupt Status for Status Output short-circuit

bool Inno4Pro_Read_Interrupt_Stat_UV (void)
Reads bit 1 on telemetry register READ17 - Interrupt Status for Status Vout(UV)

bool Inno4Pro_Read_Interrupt_Stat_OV (void)
Reads bit 0 on telemetry register READ17 - Interrupt Status for Status Vout(OV)

Variables

Local Variables

- 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 Flags

Example Flags Used for Application specific routines. Used for Specific InnoSwitch4-Pro Routines

- 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 Calibration Variables

Sets the variables needed for computations

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

InnoSwitch4-Pro I2C Register Buffers

Individual array buffers used for I2C communication each corresponds to an InnoSwitch4-Pro I2C register

These array buffers needs to be filled with LSB and MSB values. These are used directly for Writing values to InnoSwitch4-Pro via I2C. These values are initialized with InnoSwitch4-Pro default values.

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_VK**P [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_TIME**R [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}

Function Documentation

void InnoProBase_Set_Register_CV (float fVout)
Sets the CV register

Parameters:
- fVout: variable in terms of volts

Returns
- None

void InnoProBase_Set_Register_OVA (float fOva)
Sets the over-voltage threshold

Parameters:
- fOva: variable in terms of volts

Returns
- None

void InnoProBase_Set_Register_UVA (float fUva)
Sets the under-voltage threshold

Parameters:
- fUva: variable in terms of volts

Returns
- None

void InnoProBase_Set_Register_CC (float fCc)
Sets the constant current

Parameters:
- fCc: variable in terms of amps

Returns
- None

void InnoProBase_Set_Register_CDC (float fCdc)
Sets the cable drop compensation

Parameters:
- fCdc: variable in terms of mV

Returns
- None

void InnoProBase_Set_Register_VKP (float fVkp)
Sets the constant output power knee voltage

Parameters:
- fCdc: variable in terms of mV

Returns
- None

void InnoProBase_Set_Register_VBEN (float fVben)
Sets Vbus control setting

Parameters:
- fVben:
  - 3 – Enable VBEN/Disable VDIS
  - 1 – Disable VBEN/No Reset
  - 0 – Disable VBEN/Reset

Returns
- None

void InnoProBase_Set_Register_UVL (float fUvl)
Sets the under-voltage response

Parameters:
- fUvl:
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

Returns
- None

void InnoProBase_Set_Rsense_Value (float fRsense)
Sets the Rsense resistor value

Parameters:
- fRsense: resistance in mΩ

Returns
- None

float InnoProBase_Get_Register_CV (void)
Gets the value of the CV register

Parameters:
- None

Returns
- float: value in volts

float InnoProBase_Get_Register_OVA (void)
Gets the value of the over-voltage threshold

Parameters:
- None

Returns
- float: value in volts

float InnoProBase_Get_Register_UVA (void)
Gets the value of the under-voltage threshold

Parameters:
- None

Returns
- float: value in volts

float InnoProBase_Get_Register_CC (void)
Gets the value of the constant current

Parameters:
- None

Returns
- float: value in amps

float InnoProBase_Get_Register_CDC (void)
Gets the value of the cable drop compensation

Parameters:
- None

Returns
- float: value in mV

float InnoProBase_Get_Register_VKP (void)
Gets the value of the constant output power knee voltage

Parameters:
- None

Returns
- float: value in volts

float InnoProBase_Get_Register_VBEN (void)
Gets the value of the VBEN setting

Parameters:
- None

Returns
- float:
  - 3 – Enable VBEN/Disable VDIS
  - 1 – Disable VBEN/No Reset
  - 0 – Disable VBEN/Reset

float InnoProBase_Get_Register_UVL (void)
Sets the under-voltage response

Parameters:
- None

Returns
- float:
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

float InnoProBase_Get_Rsense_Value (void)
Gets the Rsense resistor value

Parameters:
- None

Returns
- float: resistance in mΩ

float Inno4Pro_Compute_CV (float fSetPt)
InnoSwitch4-Pro computation for Output Voltage (CV). Calculates based on a 10mV/LSB resolution. Limits the value from 3 V to 24 V

Parameters:

- fSetPt: Set-point value (3 to 24V)

Returns

- float: 300 to 2400

float Inno4Pro_Compute_OV (float fSetPt)
InnoSwitch4-Pro computation for Over-Voltage Threshold (OVA). Calculates based on a 100mV/LSB resolution. Limits the value form 6.2 V to 25 V

Parameters:

- fSetPt: Set-point value (6.2 to 25V)

Returns

- float: 62 to 250

float Inno4Pro_Compute_UV (float fSetPt)
InnoSwitch4-Pro computation for Under-Voltage Threshold (UVA). Calculates based on a 100mV/LSB resolution. Limits the value form 3 V to 24 V

Parameters:

- fSetPt: Set-point value (3 to 24V)

Returns

- float: 30 to 240

float Inno4Pro_Compute_CDC (float fSetPt)
InnoSwitch4-Pro computation for Cable Drop Compensation (CDC). Calculates based on a 50mV/LSB resolution. Limits the value form 0 mV to 600 mV

Parameters:

- fSetPt: Set-point value (0 to 50 mV)

Returns

- float: 0 to 12

float Inno4Pro_Compute_CC (float fSetPt)
InnoSwitch4-Pro computation for Constant Current Regulation (CC). Calculates based on a 0.16mV/Step/Rsense resolution

Parameters:

- fSetPt: Set-point value in amps

Returns

- float: 25 to 192 LSB

float Inno4Pro_Compute_VKP (float fSetPt)
InnoSwitch4-Pro computation for Output Power Knee Voltage (VKP). Calculates based on a 100mV/LSB resolution

Parameters:

- fSetPt: Set-point value (5.3 to 24 V)

Returns

- float: 53 to 240

float Inno4Pro_Compute_VBEN (float fSetPt)
InnoSwitch4-Pro computation for Series Bus Switch Control (VBEN) Applies limits from 0 to 3.

Parameters:

- None

Returns

- float:
  - 3 – Enable VBEN/Disable VDIS
  - 1 – Disable VBEN/No Reset
  - 0 – Disable VBEN/Reset

void InnoProBase_Encode_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
Handles conversion of input data to hexadecimal LSB and MSB without parity bits. The values are then stored onto a buffer.

Parameters:

- u16Temp: Value to be converted
- *u8WriteBuffer: Pointer to a memory location to where the data is stored

Returns

- None

bool InnoProBase _OddParity (uint8_t u8OddParity)
Handles odd parity bit detection.

Parameters:

- u8OddParity: Value to be evaluated with parity

Returns

- bool:
  - true – Odd number of 1's
  - false – Even number of 1's

void InnoProBase _Format_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
Handles conversion of input data to hexadecimal LSB and MSB with the evaluated parity bits. The values are then stored onto a buffer.

Parameters:

- u16Temp: Value to be converted
- *u8WriteBuffer: Pointer to a memory location to where the data is stored

Returns

- None

uint8_t InnoProBase _AddOddParity (uint16_t u16Temp)
Evaluates parity to a byte

Parameters:

- u16Temp: Value to be evaluated and added a parity bit

Returns

- uint8_t: Formatted byte with parity

void Inno4Pro_Process_Volt_Buffers (void)
Handles preparation for values to be written on InnoSwitch4-Pro voltage related registers which are over-voltage and under-voltage thresholds

Parameters:

- None

Returns

- None

bool InnoProBase_Detect_Voltage_Request (void)
Checks if there's a new voltage request. In addition, it signals b_Volt_Setting to whether it is an increase or decrease in voltage

Parameters:

- None

Returns

- bool:
  - true – Change in CV value
  - false – No change

bool InnoProBase_Detect_Current_Request (void)
Checks if there's a new current request

Parameters:

- None

Returns

- bool:

  - true – Change in CC value
  - false – No change

void Inno4Pro_Initialization (void)
Handles all common I2C configurations to be written to InnoSwitch4-Pro as initialization. This reads the InnoSwitch4-Pro system ready signal to check if InnoSwitch4-Pro is ready to communicate. Once InnoSwitch4-Pro is ready, this function configures the initial registers needed for basic operation

Parameters:

- None

Returns

- None

void Inno4Pro_Vbus_Switch_Control (bool bEnableVben)
Vbus Switch Control (VBEN Control). When VBEN is disabled, Watchdog Timer is enabled as a default. The Watchdog Timer needs to be disabled in order to enable VBEN again.

Before disabling he Vbus switch, the output voltage is carefully monitored to properly decide when to turn off the Vbus switch.

When the output voltage is detected to be at a **HIGH** setting (>5V), UV threshold is set to 3V and the CV is set to 5V first before disabling VBEN

Parameters:

- bEnableVben:
  - 1 – Enable VBEN
  - 0 – Disable VBEN with reset

Returns

- None

void Inno4Pro_Vbus_Switch_Control_NoReset (uint8_t u8EnableVben)
Similar to **Inno4Pro_Vbus_Switch_Control** but includes the Disable VBEN/No reset option

Parameters:

- bEnableVben:
  - 3 – Enable VBEN
  - 1 – Disable VBEN/No reset
  - 0 – Disable VBEN with reset

Returns

- None

void Inno4Pro_Bleeder_Enable (bool bEnable)
Handles Bleeder setting. The bleeder must not be enabled for an extended period of time to prevent excessive power dissipation

Parameters:

- bEnable:
  - 3 – Enable Bleeder with auto disable
  - 1 – Enable Bleeder
  - 0 – Disable Bleeder

Returns

- None

void Inno4Pro_Load_Discharge (bool bEnable)
Activates Vbus Load Discharge (VDIS). Enabling VDIS register will automatically disable VBEN

Parameters:

- bEnable:
  - true: Enable load discharge
  - false: Disable load discharge

Returns

- None

void Inno4Pro_TurnOff_PSU (bool bEnable)
Turns off the power supply. AC power cycling is required to restart the power supply

Parameters:

- bEnable:
  - true: Enable load discharge
  - false: Disable load discharge

Returns

- None

void Inno4Pro_FastVI_Disable (bool bDisable)
Sets CV and CC commands speed limit

Parameters:

- bEnable:
  - true: No speed limit
  - false: 10 ms update limit enabled

Returns

- None

void Inno4Pro_CVOnlyMode_Enable (bool bEnable, uint16_t u16Response, uint16_t u16Timer)
Sets constant voltage only mode. This function sets the device to constant voltage only and no constant current regulation mode. Once the load current exceeds the programmed current within the CVOL timer limit, CVOL fault setting is activated

Parameters:

- bEnable:
  - true: CVO mode enable
  - false: CVO mode disable
- u16Response:
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response
- u16Timer:
  - 3 – 16 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

Returns

- None

void Inno4Pro_Write_Volts (float fSetPtCV)
Output Voltage Control without Bleeder control. Used to update the value of the CV register

Parameters:

- fSetPtCV: Output voltage set-point value in volts

Returns

- None

void Inno4Pro_Write_Over_Volts (float fSetPtOVA, uint16_t u16OVL)
Writes over-voltage protection settings.

Parameters:

- fSetPtOVA: Over-voltage threshold in volts
- u16OVL: Over-voltage response
  - 3 – Disable-output

    - 2 – Auto-restart
    - 1 – Latch-off
    - 0 – No-response

Returns

- None

**void Inno4Pro_Write_Under_Volts (float fSetPtUVA,
uint16_t u16Uv_FaultResp,
uint16_t u16Uv_timer)**

Writes under-voltage protection settings

Parameters:

- fSetPtUVA: Under-voltage threshold in volts
- u16UVL: Under-voltage response
    - 3 – Disable-output
    - 2 – Auto-restart
    - 1 – Latch-off
    - 0 – No-response
- u16Uv_timer: Under-voltage response
    - 3 – 64 ms
    - 2 – 32 ms
    - 1 – 16 ms
    - 0 – 8 ms

Returns

- None

void Inno4Pro_Write_Cable_Drop_Comp (float fSetPtCDC)

Writes Cable Drop Compensation (CDC) settings.

Parameters:

- fSetPtCDC: Cable drop compensation in mV

Returns

- None

void Inno4Pro_Write_Amps (float fSetPtCC)

Constant Current (CC) control

Parameters:

- fSetPtCC: Constant current set-point in amps

Returns

- None

void Inno4Pro_Write_Volt_Peak (float fSetPtVpk)

Constant Output Power Voltage Threshold (VKP) control

Parameters:

- fSetPtVkp: Constant output power voltage in volts

Returns

- None

**void Inno4Pro_Write_CCSC_Fault_Response (uint16_t
u16Response)**

Writes output shirt-circuit fault response setting

Parameters:

- u16Response:
    - 3 – Disable-output
    - 2 – Auto-restart
    - 1 – Latch-off
    - 0 – No-response

Returns

- None

**void Inno4Pro_Write_ISSC_Fault_Response (uint16_t
u16Response,
uint16_t
u16Frequency,
uint16_t
u16CC)**

Writes IS-pin short fault response, frequency, and current limit setting

Parameters:

- u16Response:
    - 3 – Disable-output
    - 2 – Auto-restart
    - 1 – Latch-off
    - 0 – No-response
- 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

Returns

- None

void Inno4Pro_Write_Watchdog_Timer (uint16_t u16Timer)

Writes watchdog timer setting. Determines how long will the device continue to operate before triggering the watchdog fault

Parameters:

- u16Timer:
    - 3 – 2 s
    - 2 – 1 s
    - 1 – 0.5 s
    - 0 – No Watchdog

Returns

- None

void Inno4Pro_Write_Interupt_Mask (uint16_t u16IntMask)

Writes interrupt mask setting. The interrupt mask register must be enabled for each of the individual fault conditions to activate this feature.

Once a fault occurs, the interrupt mask is reset and the particular faults of interest must be re-enabled to activate SCL reporting scheme

Parameters:

- u16IntMask – Interrupt bit mask settings

Returns

- None

void Inno4Pro_Write_OTP_Hysteresis (uint16_t u16Otp)

Sets over-temperature hysteresis. As secondary controller die temperature increases beyond 125 °C, the active VOUT pin bleeder function will be turned off. The bleeder will not be permitted to be re-enabled until the controller temperature falls below the programmed hysteresis value

Parameters:

- u16Otp – Over-temperature hysteresis setting
    - 0 – 40°C
    - 1 – 60°C

Returns

- None

void Inno4Pro_Write_CV_Load (uint16_t u16Load)

Writes Constant Voltage Load setting. The constant current regulation mode in the InnoSwitch4-Pro can be optimized for constant voltage (CV) type load required by the application. Enabling this command register reduces the output current ripple for CV load only. This setting should only be used if CV load must be supported

Parameters:

- u16Load – Value of the CV load register

Returns

- None

void Inno4Pro_Write_Loop_Speed1 (uint16_t u16LoopSpeed)

Writes loop speed 1 settings. If faster transient response is required in the application the InnoSwitch4-Pro includes command registers to reduce the time low to high output voltage transitions.

Note: Using values other than the default or recommended settings could lead to oscillatory behavior

Parameters:

- u16LoopSpeed – Value of the Loop Speed 1

Returns

- None

void Inno4Pro_Write_Loop_Speed2 (uint16_t u16LoopSpeed)

Writes loop speed 2 settings. If faster transient response is required in the application the InnoSwitch4-Pro includes command registers to reduce the time low to high output voltage transitions.

Note: Using values other than the default or recommended settings could lead to oscillatory behavior

Parameters:

- u16LoopSpeed – Value of the Loop Speed 2

Returns

- None

bool Inno4Pro_Write_VI (float fSetPtCV, float fSetPtCC)

Output voltage control with bleeder control and constant current (CC) Control. Automatically computes for UVA and OVA settings. OVA is set to 124% of CV set-point and UVA is set to 3V.

Parameters:

- fSetPtCV: Output voltage set-point value in volts
- fSetPtCC: Constant current set-point in amps

Returns

- bool:
  - true – Process complete
  - false – Process not complete

void Inno4Pro_VBUSSC (uint16_t u16VSSC_response,
uint16_t u16Vsamples,
uint16_t u16CCThreshold)

Defines how the device will response to a series BUS switch short-circuit fault.

Parameters:

- u16VSSC_response:
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response
- u16VSamples:
  - 3 – 4 Samples
  - 2 – 3 Samples
  - 1 – 2 Samples
  - 0 – 1 Sample
- u16CCThreshold:
  - 3 – d’72
  - 2 – d’64
  - 1 – d’32
  - 0 – d’48

Returns

- None

void Inno4Pro_DCMonly (bool bEnable)

Output voltage control with bleeder control and constant current (CC) Control

Parameters:

- bEnable:
  - 1 – Enable DCM Only
  - 0 – Disable DCM Only

Returns

- None

bool Inno4Pro_Process_Voltage (bool bVoltIncrease)

Handles command sequences for voltage increment/decrement This function follows a certain sequence of commands in order to avoid unwanted triggering of UV or OV faults

Parameters:

- bVoltIncrease – Indicates if it is a voltage increase or decrease transition

Returns

- None

uint16_t InnoProBase_Telemetry (uint8_t ReadBack_Address)

Handles InnoSwitch4-Pro common I2C read back telemetry. This function reads a specific InnoSwitch4-Pro telemetry register.

Parameters:

- ReadBack_Address – Register read back address

Returns

- 2 Byte value of the register

bool InnoProBase_Read_Bit (uint8_t ReadBack_Address,
uint8_t Bit)

Handles InnoSwitch4-Pro I2C read bit

Parameters:

- ReadBack_Address – Register read back address
- Bit – Bit index

Returns

- Value of the bit

uint8_t InnoProBase_Read_Byte (uint8_t ReadBack_Address,
bool bHighByte)

Handles InnoSwitch4-Pro I2C read byte

Parameters:

- ReadBack_Address – Register read back address
- bHighByte – Indicates either MSB or LSB
  - 1 – MSB
  - 0 – LSB

Returns

- 1 Byte

uint8_t InnoProBase_Read_2Bits (uint8_t ReadBack_Address,
uint8_t u8ShiftCnt)

Handles InnoSwitch4-Pro I2C read 2 bits. The parameter u8ShiftCnt determines the number of right shifts of read back value. The last Bit[1:0] are then returned.

Parameters:

- ReadBack_Address – Register read back address

- u8ShiftCnt – Right shift count

Returns

- uint8_t: Value of the two bits (0 to 3)

float InnoProBase_Read_SetPoint (uint16_t ReadBack_Address, float fMultiplier)

Handles InnoSwitch4-Pro I2C set-point and threshold. This function is mainly used for CV, OV, and UV set-points

Parameters:

- ReadBack_Address – Register read back address
- fMultiplier – Value multiplied to the read back value

Returns

- float: Product of the read back and the fMultiplier

float Inno4Pro_Read_CV_SetPoint (void)

Reads the telemetry register READ1 - Output voltage set-point

Parameters:

- None

Returns

- float: Output voltage in volts

float Inno4Pro_Read_Output_CC_SetPoint (void)

Reads the telemetry register READ2 – Output current set-point

Parameters:

- None

Returns

- float: Constant current set-point in amps

float Inno4Pro_Read_OV_Threshold (void)

Reads the Telemetry register READ3 – Over-voltage threshold

Parameters:

- None

Returns

- float: Over-voltage threshold in volts

float Inno4Pro_Read_UV_Threshold (void)

Reads the Telemetry register READ 4 – Under-voltage threshold

Parameters:

- None

Returns

- float: Under-voltage threshold in volts

float Inno4Pro_Read_CC_SetPoint (void)

Reads the Telemetry register READ 5 – Constant current set-point

Parameters:

- None

Returns

- float: Constant current set-point in amps

float Inno4Pro_Read_CP_Threshold (void)

Reads the Telemetry register READ 5 – Constant power set-point

Parameters:

- None

Returns

- float: Constant output power knee voltage in volts

uint8_t Inno4Pro_Read_OV_Fault_Response (void)

Reads the telemetry register READ6 – Over-voltage fault response

Parameters:

- None

Returns

- uint8_t: Fault response
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

uint8_t Inno4Pro_Read_UV_Fault_Response (void)

Reads the telemetry register READ6 – Under-voltage fault response

Parameters:

- None

Returns

- uint8_t: Fault response
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

uint8_t Inno4Pro_Read_OutputSckt_Fault_Response (void)

Reads the telemetry register READ6 – Output short-circuit fault response

Parameters:

- None

Returns

- uint8_t: Fault response
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

uint8_t Inno4Pro_Read_IsPinShort_Fault_Response (void)

Reads the telemetry register READ6 - IS-pin short fault response

Parameters:

- None

Returns

- uint8_t: Fault response
  - 3 – Disable-output
  - 2 – Auto-restart
  - 1 – Latch-off
  - 0 – No-response

uint8_t Inno4Pro_Read_UV_Fault_Timer (void)

Reads the telemetry register READ6 – Under-voltage timer

Parameters:

- None

Returns

- uint8_t: Fault timer
  - 3 – 64 ms
  - 2 – 32 ms
  - 1 – 16 ms
  - 0 – 8 ms

uint8_t Inno4Pro_Read_Watchdog_Timer (void)

Reads the telemetry register READ6 - Watchdog timer

Parameters:

- None

Returns

- uint8_t: Fault timer
  - 3 – 2 s

        - 2 – 1 s
        - 1 – 0.5 s
        - 0 – No Watchdog

uint8_t Inno4Pro_Read_CvMode_Fault_Response (void)
Reads the telemetry register READ6 - Constant Voltage Mode fault response

Parameters:

- None

Returns

- uint8_t: Fault response
    - 3 – Disable-output
    - 2 – Auto-restart
    - 1 – Latch-off
    - 0 – No-response

uint8_t Inno4Pro_Read_CvMode_Timer (void)
Reads the telemetry register READ6 - Constant Voltage Mode Timer

Parameters:

- None

Returns

- uint8_t: Fault timer
    - 3 – 64 ms
    - 2 – 32 ms
    - 1 – 16 ms
    - 0 – 8 ms

bool Inno4Pro_Read_VbusSwitch (void)
Reads bit 14 on telemetry register READ6 - VBUS Switch Enable

Parameters:

- None

Returns

- uint8_t: Fault timer
    - 3 – 64 ms
    - 2 – 32 ms
    - 1 – 16 ms
    - 0 – 8 ms

bool Inno4Pro_Read_Bleeder (void)
Reads bit 13 on telemetry register READ6 - Minimum Load (Bleeder)

Parameters:

- None

Returns

- bool:
    - true – Enabled
    - false – Disabled

bool Inno4Pro_Read_PsuOff (void)
Reads bit 12 on telemetry register READ6 - Turn PSU off (Latch Off Device)

Parameters:

- None

Returns

- bool:
    - true – Enabled
    - false – Disabled

bool Inno4Pro_Read_FastVI (void)
Reads bit 11 on telemetry register READ6 - Fast VI Commands

Parameters:

- None

Returns

- bool:
    - true – Enabled
    - false – Disabled

bool Inno4Pro_Read_CvoMode (void)
Reads bit 10 on telemetry register READ6 - Constant-Voltage Mode Only

Parameters:

- None

Returns

- bool:
    - true – Enabled
    - false – Disabled

bool Inno4Pro_Read_OtpFaultHyst (void)
Reads bit 9 on telemetry register READ6 - Over-Temperature Protection

Parameters:

- None

Returns

- bool:
    - true – 60°C
    - false – 40°C

float Inno4Pro_Read_Cable_Drop_Comp (void)
Reads the telemetry register READ6 - Cable Drop Compensation

Parameters:

- None

Returns

- float: Cable drop compensation in millivolts

float Inno3Pro_Read_Amps (void)
Reads the telemetry register READ7 - Measured output current

Parameters:

- None

Returns

- float: Output current in amps

float Inno3Pro_Read_Volts (void)
Reads the telemetry register READ9 - Measured output voltage

Parameters:

- None

Returns

- float: Output voltage in volts

bool Inno4Pro_Read_Status_InterruptEnable (void)
Reads bit 15 on telemetry register READ10 - Interrupt Enable

Parameters:

- None

Returns

- bool:
    - true – Enabled
    - false – Disabled

bool Inno4Pro_Read_Status_SystemReady (void)
Reads bit 14 on telemetry register READ10 - System Ready Signal.

Note: Read10 telemetry register values are instantaneous and are cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – Ready
  - false – Not ready

bool Inno4Pro_Read_Status_OutputDischarge (void)
Reads bit 13 on telemetry register READ10 - Output Discharge

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – Enabled
  - false – Disabled

bool Inno4Pro_Read_Status_HighSwitchFreq (void)
Reads bit 12 on telemetry register READ10 - Switching Frequency High

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – High frequency
  - false – Low frequency

bool Inno4Pro_Read_Status_OtpFault (void)
Reads bit 9 on telemetry register READ10 - Over-Temperature Protection

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – Protection enabled
  - false – Protection disabled

bool Inno4Pro_Read_Status_Vout2pct (void)
Reads bit 5 on telemetry register READ10 - 2% Bleeder Enabled. The InnoSwitch4-Pro will automatically activate a weak current bleeder on the VOUT-pin until the output voltage settles to less than 2% of the CV set-point

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – 2% bleeder enabled
  - false – 2% bleeder disabled

bool Inno4Pro_Read_Status_Vout10pct (void)
Reads bit 4 on telemetry register READ10 - VOUTADC > 1.1*Vout. The InnoSwitch4-Pro will automatically activate a weak current bleeder on the VOUT-pin until the output voltage settles to less than 10% of the CV set-point

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – 10% bleeder enabled
  - false – 10% bleeder disabled

bool Inno4Pro_Read_Status_IsPinShort (void)
Reads bit 3 on telemetry register READ10 - IS-pin Short Circuit Detected

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – IS-pin short-circuit detected
  - false – No IS-pin short-circuit

bool Inno4Pro_Read_Status_OutputShorCkt (void)
Reads bit 2 on telemetry register READ10 – Output Short Circuit Detected

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – Output short-circuit detected
  - false – No output short-circuit

bool Inno4Pro_Read_Status_UV_Fault (void)
Reads bit 1 on telemetry register READ10 - Output Voltage UV Fault Comparator

Note: This register value is instantaneous and is cleared whenever the condition is no longer valid.

Parameters:

- None

Returns

- bool:
  - true – Under-voltage fault detected
  - false – No under-voltage fault

bool Inno4Pro_Read_Status_OV_Fault (void)
Reads bit 0 on telemetry register READ10 - Output Voltage OV Fault Comparator

Parameters:

- None

Returns

- bool:
  - true – Over-voltage fault detected
  - false – No over-voltage fault

float Inno4Pro_Read_AmpsAverage (void)
Reads the Telemetry register READ12 - Average output current

Parameters:

- None

Returns

- float: Average output current in amps

float Inno4Pro_Read_VoltsAverage (void)
Reads the Telemetry register READ13 - Average output voltage

Parameters:

- None

Returns

- float: Average output voltage in volts

float Inno3Pro_Read_Voltage_DAC (void)
Reads the Telemetry register READ14 - Voltage DAC

Parameters:

- None

Returns

- float: DAC output voltage in volts

bool Inno4Pro_Read_Status_CvoMode_AR (void)
Reads bit 15 on telemetry register READ16 - CVO Mode auto-restart(AR)

Parameters:

- None

Returns

- bool:
    - true – CVO Mode auto-restart occurred
    - false – No fault

bool Inno4Pro_Read_Status_IsPinShort_AR (void)
Reads bit 12 on telemetry register READ16 - IS-pin Short Circuit auto-restart(AR)

Parameters:

- None

Returns

- bool:
    - true – IS-pin short-circuit AR occurred
    - false – No fault

bool Inno4Pro_Read_Status_OutputShortCkt_AR (void)
Reads bit 12 on telemetry register READ16 – Output short-circuit auto-restart(AR)

Parameters:

- None

Returns

- bool:
    - true – Output short-circuit AR occurred
    - false – No fault

bool Inno4Pro_Read_Status_OV_AR (void)
Reads bit 10 on telemetry register READ16 - Output Voltage OV auto-restart(AR)

Parameters:

- None

Returns

- bool:
    - true – Over-voltage fault AR occurred
    - false – No fault

bool Inno4Pro_Read_Status_UV_AR (void)
Reads bit 9 on telemetry register READ16 - Output Voltage UV auto-restart(AR)

Parameters:

- None

Returns

- bool:
    - true – Under-voltage fault AR occurred
    - false – No fault

bool Inno4Pro_Read_Status_LatchOff (void)
Reads bit 7 on telemetry register READ16 - Latch-Off (LO) Occurred

Parameters:

- None

Returns

- bool:
    - true – Latch-off occurred
    - false – No fault

bool Inno4Pro_Read_Status_CvoMode_LO (void)
Reads bit 6 on telemetry register READ16 - CVO Mode Latch-Off (LO)

Parameters:

- None

Returns

- bool:
    - true – CVO mode latch-off occurred
    - false – No fault

bool Inno4Pro_Read_Status_PsuOffCmd (void)
Reads bit 5 on telemetry register READ16 - PSU Turn-Off Command Received

Parameters:

- None

Returns

- bool:
    - true – PSU turn-off command received
    - false – No fault

bool Inno4Pro_Read_Status_IsPinShort_LO (void)
Reads bit 4 on telemetry register READ16 - IS-pin Short Circuit Latch-Off (LO)

Parameters:

- None

Returns

- bool:
    - true – IS-pin short-circuit latch-off occurred
    - false – No fault

bool Inno4Pro_Read_Status_OV_LO (void)
Reads bit 2 on telemetry register READ16 - Output Voltage OV Latch-Off (LO)

Parameters:

- None

Returns

- bool:
    - true – Over-voltage latch-off occurred
    - false – No fault

bool Inno4Pro_Read_Status_UV_LO (void)
Reads bit 1 on telemetry register READ16 - Output Voltage UV Latch-Off (LO)

Parameters:

- None

Returns

- bool:
  - true – Under-voltage latch-off occurred
  - false – No fault

bool Inno4Pro_Read_Status_BPS_LO (void)
Reads bit 0 on telemetry register READ16 - BPS-pin Latch-Off (LO). This bit indicates whether or not an over-voltage fault was detected on the BPS-pin

Parameters:

- None

Returns

- bool:
  - true – BPS-pin latch-off occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Mask_CntrlSecondary (void)
Reads bit 15 on telemetry register READ17 - Interrupt Mask Control Secondary

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for Control Secondary enabled
  - false – Interrupt for Control Secondary disabled

bool Inno4Pro_Read_Interrupt_Mask_BpsCurrentLo (void)
Reads bit 13 on telemetry register READ17 - Interrupt Mask BPS Current latch-off

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for BPS current latch-off enabled
  - false – Interrupt for BPS current latch-off disabled

bool Inno4Pro_Read_Interrupt_Mask_CvoPkLoadTimer (void)
Reads bit 12 on telemetry register READ17 - Interrupt Mask BPS Current latch-off

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for CVO mode peak load timer is enabled
  - false – Interrupt for CVO mode peak load timer is disabled

bool Inno4Pro_Read_Interrupt_Mask_IsPinShort (void)
Reads bit 11 on telemetry register READ17 - Interrupt Mask IS-pin short

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for IS-pin short enabled
  - false – Interrupt for IS-pin short disabled

bool Inno4Pro_Read_Interrupt_Mask_OutputShortCkt (void)
Reads bit 10 on telemetry register READ17 - Interrupt Mask Output short-circuit

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for output short-circuit enabled
  - false – Interrupt for output short-circuit disabled

bool Inno4Pro_Read_Interrupt_Mask_UV (void)
Reads bit 9 on telemetry register READ17 - Interrupt Mask Vout under-voltage(UV)

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for Vout (UV) enabled
  - false – Interrupt for Vout (UV) disabled

bool Inno4Pro_Read_Interrupt_Mask_OV (void)
Reads bit 8 on telemetry register READ17 - Interrupt Mask Vout over-voltage(OV)

Note: Once fault occurs, the interrupt mask is reset and must be re-enabled to activate the SCL reporting scheme

Parameters:

- None

Returns

- bool:
  - true – Interrupt for Vout (OV) enabled
  - false – Interrupt for Vout (UV) disabled

bool Inno4Pro_Read_Interrupt_Stat_CntrlSecondary (void)
Reads bit 6 on telemetry register READ17 - Interrupt Status Control Secondary

Parameters:

- None

Returns

- bool:
  - true – Control Secondary fault has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_OMF(void)
If interrupt mask is enabled, this interrupt is raised whenever the operating mode changes from CV to CC and vice-versa.

Parameters:

- None

Returns

- bool:

  - true – Operating mode change occurred
  - false – No change in the operating mode

bool Inno4Pro_Read_Interrupt_Stat_VBUSSC (void)
This function tells that that the current sensed by the IS-pin is greater than the set VBUSSC register when VBEN is disabled

Parameters:
- None

Returns
- bool:
  - true – VBUSSC fault has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_BpsCurrentLo (void)
Reads bit 6 on telemetry register READ17 - Interrupt Status for BPS

Parameters:
- None

Returns
- bool:
  - true – BPS fault has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_CvoPkLoadTimer (void)
Reads bit 5 on telemetry register READ17 - Interrupt Status for CVO Mode Peak load timer

Parameters:
- None

Returns
- bool:
  - true – Constant Voltage Mode fault has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_IsPinShort (void)
Reads bit 3 on telemetry register READ17 - Interrupt Status for Status IS-pin short

Parameters:
- None

Returns
- bool:
  - true – IS-pin short-circuit has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_OutputShortCkt(void)
This function tells that CCSC fault has occurred and an interrupt on the SCL was generated

Parameters:
- None

Returns
- bool:
  - true – Output short-circuit has occurred
  - false – No fault
  -

bool Inno4Pro_Read_Interrupt_Stat_UV (void)
Reads bit 1 on telemetry register READ17 - Interrupt Status for Status Vout(UV)

Parameters:
- None

Returns
- bool:
  - true – Under-voltage fault has occurred
  - false – No fault

bool Inno4Pro_Read_Interrupt_Stat_OV (void)
Reads bit 0 on telemetry register READ17 - Interrupt Status for Status Vout(OV)

Parameters:
- None

Returns
- bool:
  - true – Over-voltage fault has occurred
  - false – No fault

Revision	Notes	Date
A	Initial release.	01/20/23

For the latest updates, visit our website: 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 Worldwide Sales Support Locations

World Headquarters
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

China (Shanghai)
Rm 2410, Charity Plaza, No. 88
North Caoxi Road
Shanghai, PRC 200030
Phone: +86-21-6354-6323
e-mail: chinasales@power.com

China (Shenzhen)
17/F, Hivac Building, No. 2, Keji Nan
8th Road, Nanshan District,
Shenzhen, China, 518057
Phone: +86-755-8672-8689
e-mail: chinasales@power.com

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

Germany (Gate Driver Sales)
HellwegForum 1
59469 Ense
Germany
Tel: +49-2938-64-39990
e-mail: igbt-driver.sales@power.com

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

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

Japan
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

Korea
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

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

Taiwan
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

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