Application Note AN-110
InnoSwitch5-Pro Family

Programming Manual

Introduction

This manual describes the firmware considerations for the InnoSwitch5-Pro IC family with respect to I²C write and read protocol format, and provides example calculations for output voltage and current commands. Also addressed are parity bit implementation, effects of a system reset, recommended sequences for initialization and voltage transitions, telemetry register examples, and a high-level overview of the code libraries (PIC microcontroller and Arduino platform).

InnoSwitch5-Pro I²C Protocol Format

In this section, the InnoSwitch5-Pro I²C Write and Read protocol formats are explained with the following conventions:
[A] – Slave acknowledgement
[a] – Master acknowledgement
[na] – Master NACK
[W] – Write command (1'b0)
[r] – Read command (1'b1)
[PI_SLAVE_ADDRESS] =0x18 (7'b001 1000)
[PI_COMMAND] – PI Command Register Address
[TELEMETRY_REGISTER_ADDRESS] – Telemetry Register Address

I²C Slave Address

The InnoSwitch5-Pro device has a 7-bit slave address of 0x18 (7'b001 1000).

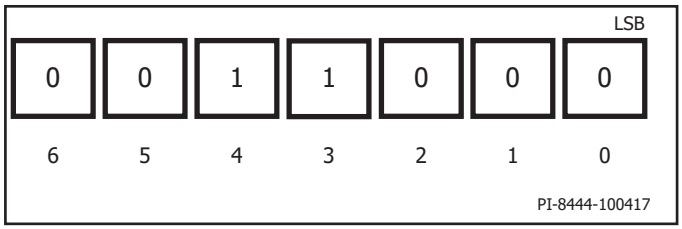


Figure 1. Slave Address.

For an I²C communication, the 7-bit slave address is shifted to the left by one bit and is appended by either a '0' or a '1' to specify whether the transaction is I²C Write or I²C Read, respectively. Therefore, the InnoSwitch5-Pro 7-bit slave address with write/read bit will be as follows:

- I²C Write: PI_SLAVE_ADDRESS + W = 0x30 (8b'0011 0000)
- I²C Read: PI_SLAVE_ADDRESS + r = 0x31 (8b'0011 0001)

I²C Write Format

An InnoSwitch5-Pro IC I²C Write Command is either a 2-byte or a 3-byte write, depending on the data length of the corresponding command register that will be written. Refer to the InnoSwitch5-Pro family datasheet to determine whether a command type has a 16-bit or 8-bit data length.
Command registers with only 8-bit data length (W_Byte) will use the 2-byte write command format:

- [PI_SLAVE_ADDRESS][W][A][PI_COMMAND][A][Low Byte][A]

Similarly, command registers with 16-bit data length (W_Word) will use the 3-byte write command format:

- [PI_SLAVE_ADDRESS][W][A][PI_COMMAND][A][Low Byte] [A] [High Byte][A]

I²C transactions start with the I²C slave address including the write/read bit. For a write command, this becomes 0x30. The succeeding bytes consist of the PI command register address and the data bytes to be written. Examples of I²C write sequences for a 2-byte and 3-byte commands are shown in Figure 2.

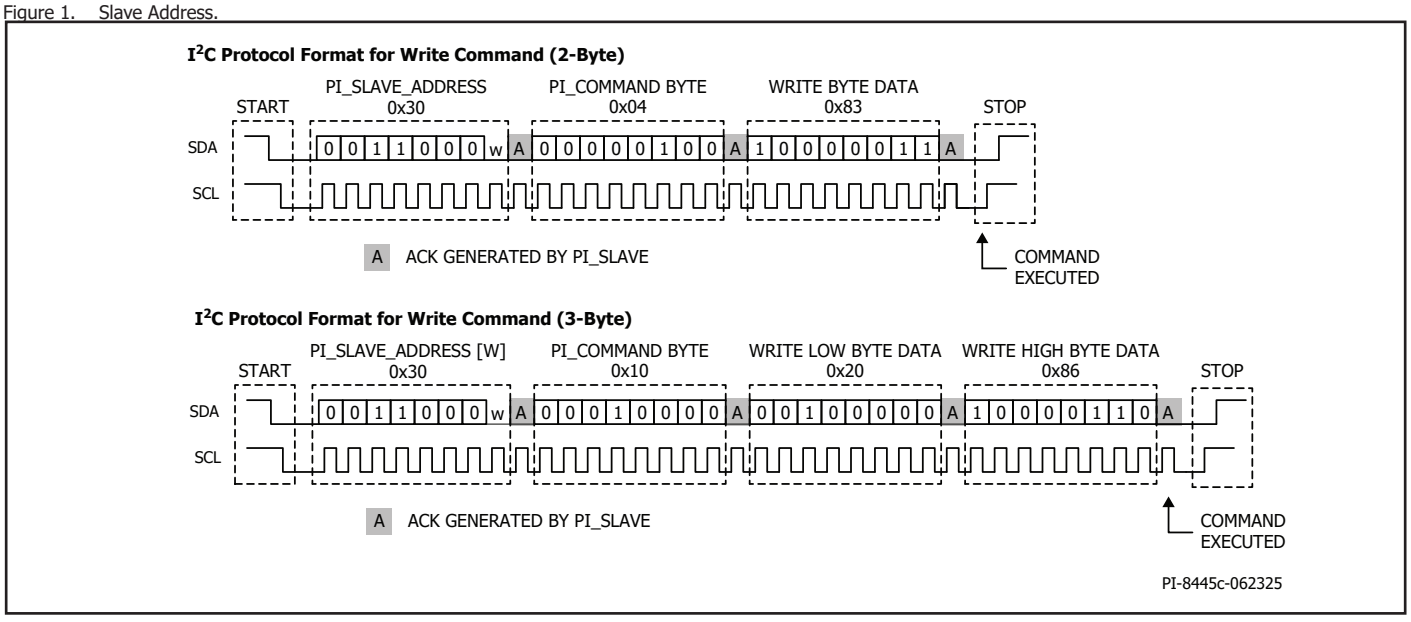


Figure 2. Example Register Write Sequences for a 2-Byte Command (VBEN register) and a 3-Byte Command (CV register).

I²C Read Format

An InnoSwitch5-Pro IC I²C read transaction is a 3-byte write followed by a 2-byte read. The first part (3-byte write) contains the PI slave address with write bit (0x30), followed by the read register address (0x80), Start-Telemetry register, and End-Telemetry register. This informs the InnoSwitch5-Pro IC that the I²C master (external micro-controller) is requesting for the values of the specified telemetry registers. When reading back the value of a single telemetry register, the Start-and End-Telemetry registers are set to the same value.

- [PI_SLAVE_ADDRESS][W][A][PI_COMMAND][A] [START_TELEMETRY_REGISTER_ADDRESS][A] [END_TELEMETRY_REGISTER_ADDRESS [A]

Once the telemetry register has been specified, the second part is performed with a 2-byte read. This contains the PI slave address with read bit (0x31), followed by the low byte read-back and high- byte readback values respectively provided by the InnoSwitch5-Pro device.

- [PI_SLAVE_ADDRESS][r][A] {PI Slave responds Low Byte}[a] {PI Slave responds High Byte}[na]

Telemetry registers with adjacent addresses can be read sequentially by setting the Start-and End-Telemetry registers correspondingly. In this case, the first 2-byte read will give the telemetry value for the Start-Telemetry register, and for each succeeding 2-byte read, the telemetry register address automatically increments until the End-Telemetry register has been reached. Note that each 2-byte read needs to terminate with an I²C Stop bit to automatically increment the telemetry register address.

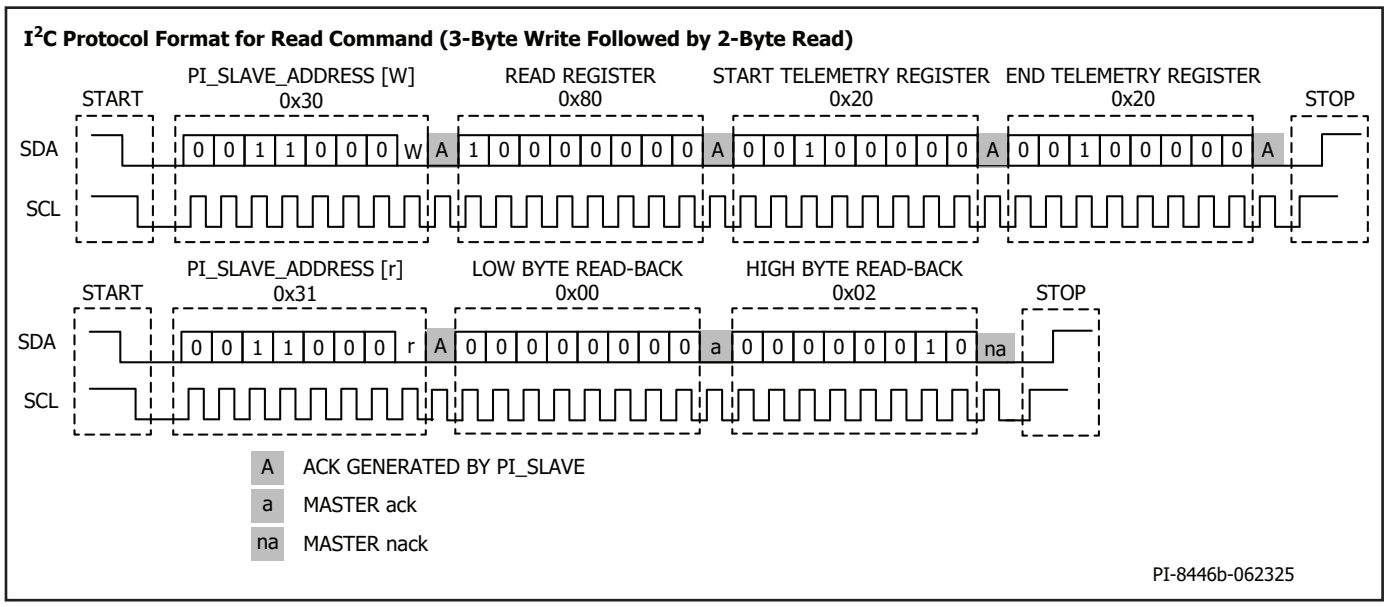


Figure 3. Example Register Read Sequence (READ16, Telemetry Register Address 0x20).

Command Register Example Calculations Output Voltage Register, CV (0x10)

The output voltage of the power supply is regulated on the VOUT pin and is configured with the CV register (address 0x10). The CV register has a valid programming range from 3 V to 30 V with a bit resolution of 10 mV/LSB.

To convert the desired output regulation voltage (volts) into CV register value (data), divide the voltage by the bit resolution and then implement odd parity on both high byte and low byte data.
Example: Convert 20 V into CV register value.

Step 1 – Divide the voltage setpoint by the bit resolution to obtain the equivalent decimal value.

$$\text{CV (decimal equivalent)} = \frac{\text{Set Point in Volts}}{\text{Resolution}} = \frac{20\text{ V}}{(10\text{ mV / LSB})} = 2000$$

Step 2 – Implement odd parity on high byte and low byte data (refer to Parity Bit Implementation section for more details).

$$\text{CV (decimal equivalent)} = 2000 = 0\text{x07D0 (hex)}$$
$$\text{CV Register Value (with Odd Parity)} = 0\text{x8FD0 (hex)}$$

After calculating the CV register value with odd parity data applied, the I²C write command protocol format described in the previous section can then be used to configure the CV register.

CV Command – Code Example

In this pseudo-code, the function input parameter u16SetPtCV is the decimal equivalent of the desired output voltage (e.g. 2000 decimal for 20 V output). Limiting the CV Setpoint within the programmable range (3 V to 30 V) is recommended. The saturation macro sig_minmax sets the lower and upper limits. Any u16SetPtCV value lower or higher than 300 or 3000 will be clamped to these values, respectively. Once the data value has been limited, an odd parity function is used to format the high byte and low byte data and the I²C write command is then sent.

```
#define INNO5PRO_CV_SET_PT_MIN 300
#define INNO5PRO_CV_SET_PT_MAX 3000
#define sig_minmax(sig, min, max) ((sig < min) ? sig =
          min : (sig > max) ? sig = max : 0)

void Inno5Pro_Write_Volts(uint16_t u16SetPtCV)
{
    uint8_t u8Buffer[2] =  {0};

    // Limit the setpoint within range
    sig_minmax (u16SetPtCV,
                INNO5PRO_CV_SET_PT_MIN,
                INNO5PRO_CV_SET_PT_MAX);

    // Format data with odd parity into buffer
    InnoProBase_Format_Buffer(u16SetPtCV,
                              u8Buffer);

    // I2C Write into CV Register

    I2C_Write16(INNOPRO_I2C_ADDRESS,
                INNO5PRO_CV,
                u8Buffer,
                WR_WORD)}
}
```

Constant Current Register, CC (0x98)

Output constant current regulation is configured with the CC register (address 0x98), which sets the IS-GND pin voltage regulation threshold. By default, the CC register is set at the full-scale value of d'192. This also corresponds to the full-scale voltage of 32 mV across the IS-GND pins ($I_{SV(TH)}$ typical), and the full-scale CC threshold is set by the current sense resistor R_{SENSE} across the IS-GND pins. The recommended values for are R_{SENSE} are either 6 mΩ or 9 Ω for designs with 5 A or 3 A rated load current, respectively.

$$\text{Full Scale cc Threshold (Amps)} = \frac{32\text{ mV}}{R_{SENSE}}$$
$$\approx 5.33\text{ A (with } R_{SENSE} \text{ 6 m}\Omega)$$
$$\approx 3.56\text{ A (with } R_{SENSE} \text{ 9 m}\Omega)$$

The CC register has a valid programming range from 15% (d'29) to 100% (d'192) of the full-scale and has a CC bit resolution (mA/LSB) dependent on R_{SENSE} as shown in the equation below. The effective R_{SENSE} seen by the IS-GND pins may be slightly different based on the PCB layout and therefore characterization on a hardware prototype is recommended.

$$\text{CC Bit Resolution} = \left(\frac{32\text{ mV}}{R_{SENSE}}\right)\left(\frac{1}{192\text{ LSB}}\right)$$
$$\approx 27.3\text{ mA/LSB} (\text{with effective } R_{SENSE} \approx 6.1\text{ m}\Omega)$$
$$\approx 18.3\text{ mA/LSB} (\text{with effective } R_{SENSE} \approx 9.1\text{m}\Omega)$$

To convert the desired CC regulation threshold (mA) into CC register value (data), divide the target current by the CC bit resolution and then implement odd parity on both the high byte and the low byte data.

Example: Convert 5.0 A into CC register value with $R_{SENSE} \approx 6\text{ m}\Omega$.

Step 1 – Divide the current setpoint by the bit resolution to obtain the equivalent decimal value. The target current setpoint is typically expressed in mA. To minimize errors, it is not recommended to round up/down the CC bit resolution (e.g. use 27.3 mA/LSB instead of 27 mA/LSB) while ensuring there is no variable overflow in the firmware calculations. Without floating point variables, this can be performed by multiplying the numerator by 10 and directly using "273" for the denominator as shown below.

$$\text{CC (decimal equivalent)} = \frac{\text{Set Point in mA}}{\text{CC Bit Resolution in mA/LSB}} = \left(\frac{(5000)\text{ mA} \times (10)}{273\text{ mA/LSB}}\right) = 183$$

Step 2 – Implement odd parity on high byte and low byte data (refer to Parity Bit Implementation section for more details).

$$\text{CC (decimal equivalent)} = 183 = 0\text{x00B7 (hex)}$$
$$\text{CC Register Value (with Odd Parity)} = 0\text{x0137 (hex)}$$

CC Command – Code Example

In this pseudo-code, the function input parameter u16SetPtCCmA is the desired current threshold in mA (e.g. 5000 for 5.0 A threshold). This is converted to CC register equivalent decimal value using the CC bit resolution. The saturation macro (sig_minmax) sets the lower and upper limits such that any u16SetPtCC value lower or higher than 29 or 192 will be clamped to these values, respectively. Once the data value has been set, an odd parity function is used to format the high byte and low byte data and then the I²C write command is sent.

```
#define INNO5PRO_CC_SET_PT_MIN 29
#define INNO5PRO_CC_SET_PT_MAX 192
// Rsense ~6.1mohm
// CC Bit Resolution (32mV/Rsense)*(1/192LSB)
// 27.3 mA/LSB for Rsense 6.1mohm
#define INNO5PRO_CC_BIT_RESOLUTION (273)

void Inno5Pro_Write_Amps(uint16_t u16SetPtCCmA)
{
    uint8_t u8Buffer[2] =  {0};
    uint16_t u16SetPtCC =  0;

    // e.g. Convert 5000 mA into CC value
    // 5000mA * 10 / 273 = 183 decimal
    u16SetPtCC = ((u16SetPtCCmA * 10)/
                  INNO5PRO_CC_BIT_RESOLUTION);

    // Limit the setpoint within range
    sig_minmax (u16SetPtCC,
                INNO5PRO_CC_SET_PT_MIN,
                INNO5PRO_CC_SET_PT_MAX);

    // Format data with odd parity into buffer
    InnoProBase_Format_Buffer(u16SetPtCC,
                              u8Buffer);

    // I2C Write into CC Register


    // I2C_Write16 (INNOPRO_I2C_ADDRESS,
                    INNO5PRO_CC,
                    u8Buffer,
                    WR_WORD);
}
```

CC Command with End of Line Calibration – Code Example

In this pseudo code, End of Line Calibration telemetry is used to adjust the CC setpoint to improve the output current tolerance between multiple InnoSwitch5-Pro devices. This method is preferred over the previous example where the there is no CC offset correction applied.

The internal CC calibration offset is stored within each device. It is available as an I²C telemetry (READ23, 0x2E), which provides the digital value of CC regulation offset (bit[2:0]) and whether it is a positive or a negative offset (bit[3]). A one-time check during initialization is sufficient to store the READ23 value into a variable, that can be used as the 2nd input parameter when using the function below.

```
#define INNO5PRO_CC_SET_PT_MIN 29
#define INNO5PRO_CC_SET_PT_MAX 192
// Rsense ~6.1mohm
// CC Bit Resolution (32mV/Rsense)*(1/192LSB)
// 27.3 mA/LSB for Rsense 6.1mohm
#define INNO5PRO_CC_BIT_RESOLUTION (273)

void Inno5Pro_Write_Amps_EOL(uint16_t u16SetPtCCmA,
                             uint16_t u16EOL)
{
    uint8_t u8Buffer[2] =  {0};
    uint16_t u16SetPtCC =  0;
    uint16_t u16Offset =  0;
    bool bSign =  0;

    // e.g. Convert 5000 mA into CC value
    // 5000mA * 10 / 273 = 183 decimal
    u16SetPtCC = ((u16SetPtCCmA * 10)/
                  INNO5PRO_CC_BIT_RESOLUTION);

    // Extract Calibration Offset and Sign from raw data
    u16Offset = (u16EOL & 0x0007);
    bSign = (u16EOL >> 3);

    if (bSign)
    {
        // Offset is positive, add to CC Setpoint
        u16SetPtCC = u16SetPtCC + u16Offset;
    }
    else
    {
        // Offset is negative, subtract to CC Setpoint
        u16SetPtCC = u16SetPtCC - u16Offset;
    }

    // Limit the setpoint within range
    sig_minmax (u16SetPtCC,
                INNO5PRO_CC_SET_PT_MIN,
                INNO5PRO_CC_SET_PT_MAX);

    // Format data with odd parity into buffer
    InnoProBase_Format_Buffer(u16SetPtCC,u8Buffer);

    // I2C Write into CC Register
    I2C_Write16 (INNOPRO_I2C_ADDRESS,
                 INNO5PRO_CC,
                 u8Buffer,
                 WR_WORD);
}
```

Cable Drop Compensation Register, CDC (0x16)

The CDC register has a valid programming range from 0 V to 600 mV (d'0 to d'12) with a bit resolution of 50 mV/LSB. The increase in output voltage regulation is dependent on the voltage across IS-GND pins (current through R_{SENSE}). At no-load there is no CDC, and the voltage compensation linearly increases as load increases and reaches the maximum programmed CDC value when IS-GND pin voltage is at $I_{SV(TH)}$ (32 mV typical).

CDC Command – Code Example

The mapping between the desired CDC setpoint (in mV) and its decimal data value can be simplified using macros. In this pseudo-code, the function can be called with the macro as the input parameter based on the desired CDC value.

```
#define INNO5PRO_CDC_SET_PT_MIN 0
#define INNO5PRO_CDC_SET_PT_MAX 12
#define INNO5PRO_CDC_SET_PT_0MV      0
#define INNO5PRO_CDC_SET_PT_50MV     1
#define INNO5PRO_CDC_SET_PT_100MV    2
#define INNO5PRO_CDC_SET_PT_150MV    3
#define INNO5PRO_CDC_SET_PT_200MV    4
#define INNO5PRO_CDC_SET_PT_250MV    5
#define INNO5PRO_CDC_SET_PT_300MV    6
#define INNO5PRO_CDC_SET_PT_350MV    7
#define INNO5PRO_CDC_SET_PT_400MV    8
#define INNO5PRO_CDC_SET_PT_450MV    9
#define INNO5PRO_CDC_SET_PT_500MV    10
#define INNO5PRO_CDC_SET_PT_550MV    11
#define INNO5PRO_CDC_SET_PT_600MV    12

void Inno5Pro_Write_Cable_Drop_Comp(uint16_t u16SetPtCDC)
{
    uint8_t u8Buffer[2] =  {0};

    // Limit the setpoint within range
    sig_minmax (u16SetPtCDC,
                INNO5PRO_CDC_SET_PT_MIN,
                INNO5PRO_CDC_SET_PT_MAX);

    // Format data into buffer
    InnoProBase_Encode_Buffer (u16SetPtCDC,
                               u8Buffer);

  // I2C Write into CDC Register
    I2C_Write16 (INNOPRO_I2C_ADDRESS,
                INNO5PRO_CDC,
                u8Buffer,
                WR_BYTE);
}
```

Undervoltage Fault Register, UVA (0x94)

The UVA register configures the following settings for output UV protection in a single command: UV threshold, fault response, timer, and timer enable/disable. The UV threshold has a valid programming range from 2.7 V to 40 V with a bit resolution of 100 mV/LSB, and the options for the remaining settings are as listed in the InnoSwitch5-Pro family data sheet.

To convert the desired UV threshold (volts) into UV register value (data), divide the voltage by the bit resolution. Append the bits for the remaining UV settings and then implement odd parity on both high byte and low byte data.

Example: Set UV register value as 4.5 V, auto-restart, 8 ms timer, and UV timer enabled.

Step 1 – Divide the UV threshold by the bit resolution to obtain the equivalent decimal value.

$$\text{UV (decimal equivalent)} = \frac{\text{Set Point in Volts}}{\text{Resolution}} = \frac{4.5\ \text{V}}{(10\ \text{mV / LSB}} = 45$$

Step 2 – Append the fault response, timer, and timer enable settings by using bit shifts and bitwise-OR operation (refer to the pseudo-code for implementation). Here, the amount of bit shifts does not include the parity bit and the settings will be properly aligned to their bit assignments once the parity bit has been inserted.

UV Threshold = d'45 (4.5 V)
UV Response = 0x02 (AR)
UV Timer = 0x00 (8 ms)
UV Timer Enable = 0 x00 {Enabled}
UV Register Value (before Odd Parity) = 0x042D (hex)

Step 3 – Implement odd parity on high byte and low byte data (refer to Parity Bit Implementation section for more details).

UV Register Value (with Odd Parity) = 0x08AD (hex)

UVA Command – Code Example

In this pseudo-code, the function's first input parameter u16SetPtUVA is the decimal equivalent of the desired UV threshold (e.g. 45 decimal for 4.5 V). The provided macros for fault response, timer, and timer enable settings can then be used for the next three input parameters when calling the function.

The saturation macro sig_minmax sets the lower and upper limits such that any u16SetPtUVA value lower or higher than 27 or 400 will be clamped to these values, respectively. Once the threshold value has been limited, the remaining UV settings are then appended. An odd parity function is used to format the high byte and low byte data and the I²C write command is then sent.

```
#define INNO5PRO_UV_SET_PT_MIN 27
#define INNO5PRO_UV_SET_PT_MAX 400
#define INNO5PRO_UVL_FAULT_RESPONSE_NORESPONSE          0x0
#define INNO5PRO_UVL_FAULT_RESPONSE_LATCHOFF 0x1
#define INNO5PRO_UVL_FAULT_RESPONSE_AUTORESTART         0x2
#define INNO5PRO_UVL_FAULT_RESPONSE_DISABLEOUTPUT       0x3
#define INNO5PRO_UVL_FAULT_TIMER_8MS            0x0
#define INNO5PRO_UVL_FAULT_TIMER_16MS           0x1
#define INNO5PRO_UVL_FAULT_TIMER_32MS           0x2
#define INNO5PRO_UVL_FAULT_TIMER_64MS           0x3
#define INNO5PRO_UVL_FAULT_TIMER_ENABLE         0x0
#define INNO5PRO_UVL_FAULT_TIMER_DISABLE        0x1

void Inno5Pro_Write_Under_Volts (uint16_t u16SetPtUVA,
                                 uint16_t u16Uv_FaultResp,
                                 uint16_t u16Uv_timer,
                                 uint16_t u16Uv_timer_En)
{
    uint8_t u8Buffer[2] =  {0};

    // Limit the setpoint within range
    sig_minmax(u16SetPtUVA,
               INNO5PRO_UV_SET_PT_MIN,
               INNO5PRO_UV_SET_PT_MAX);

    // Append UVL response, Timer, and Timer Enable bits
    // before inserting parity bit
    u16SetPtUVA = u16SetPtUV  |
    (u16Uv_FaultResp          << 9)      |
    (u16Uv_timer              << 11)     |
    (u16Uv_timer_En << 13);

    // Format data with odd parity into buffer
    InnoProBase_Format_Buffer(u16SetPtUVA,
                              u8Buffer);

    // I2C Write into UVA Register

    I2C_Write16(INNOPRO_I2C_ADDRESS,
          INNO5PRO_UVA,
          u8Buffer,
          WR_BYTE);
}
```

Parity Bit Implementation

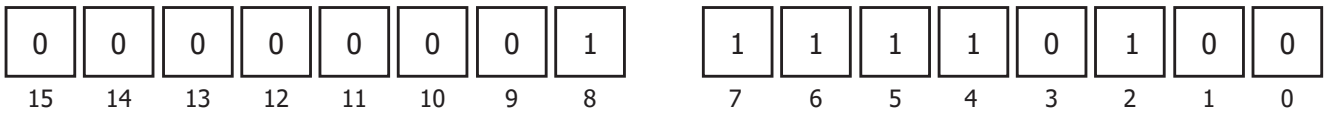
As listed in the command register assignments from the InnoSwitch5-Pro family data sheet, some of the registers requires odd parity error bits to be added to the high and low byte data. In odd parity bit error checking, the total number of 1's in the binary format of the 8-bit data (including the parity bit) must be an odd number.

For the low byte, the parity bit (bit[7]) selection will change based on the data contents from bit[6:0]. Similarly for the high byte, the parity bit (bit[15]) will be set based on existing data from bit[14:8]. This is illustrated in the example below.

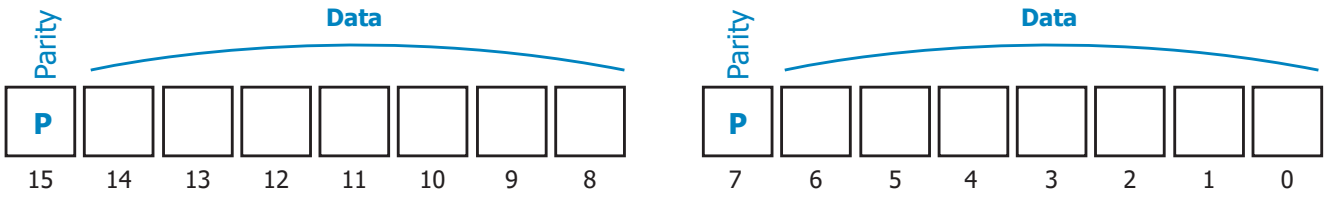
Example for CV-Setpoint = 5.00 V
(Note: CV register requires odd parity on both high and low byte data.)

Register	Output Voltage	CV (Decimal Equivalent)	Hex without Parity	Hex with Odd Parity
CV	5.00 V	500	0x01f4	0x83f4

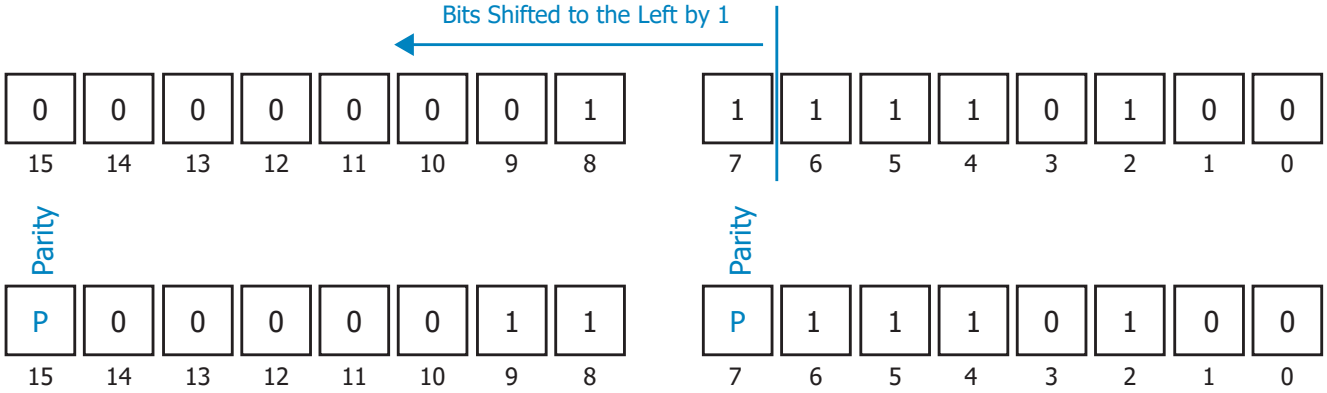
Binary Equivalent of d'500 (initial data):



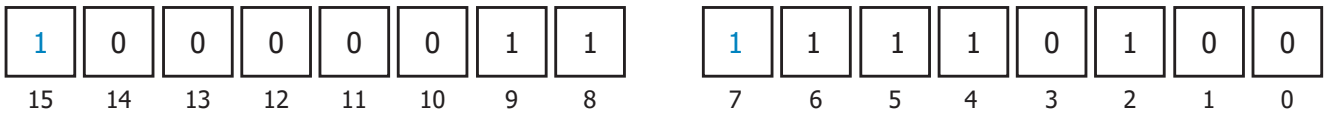
Odd parity bits will be assigned for bit[15] and bit[7]. The remaining bits will be used for the actual data.



From the initial binary data, all bits from bit[7] and above must be shifted to the left by one place to reserve the position for the parity bits.



For the low byte, if bit[6:0] has an odd number of 1's, then the parity bit[7] is cleared to '0'. Otherwise, it is set to '1'.
For the high byte, if bit[14:8] has an odd number of 1's, then the parity bit[15] is cleared to '0'. Otherwise, it is set to '1'.
In this example, a 5.00 V CV setpoint results in a CV Register value = 0x83F4.



PI-10213-071725

Odd Parity – Code Example

In this pseudo-code, the function InnoProBase_Format_Buffer has two input parameters: the initial data that needs odd parity format, and the pointer to the variable where the formatted result will be stored. The initial data has the low byte and high byte masked. Each masked data set is checked and modified accordingly to implement odd parity using the functions Inno5Pro_CheckOddParity and Inno5Pro_AddOddParity, respectively.

```
void InnoProBase_Format_Buffer (uint16_t u16Temp, uint8_t *u8WriteBuffer)
{
    // Extract the low byte bit[6:0] from initial data and then add the parity bit
    // Store the result into low byte buffer for I2C Write
    u8WriteBuffer[0] = Inno5Pro_AddOddParity(u16Temp & 0x007F);

    // Extract the high byte bit[13:7] from initial data and then add the parity bit
    // Store the result into high byte buffer for I2C Write
    u8WriteBuffer[1] = Inno5Pro_AddOddParity((u16Temp & 0x3F80) >> 7);
}

uint8_t Inno5Pro_AddOddParity(uint8_t u8Temp)
{
    // Check odd parity and update parity bit
    if (Inno5Pro_CheckOddParity(u8Temp))
    {
        // No of 1's is Odd. Keep parity bit clear.
        return u8Temp;
    }
    else
    {
        // No of 1's is Even. Assert parity bit.
        return set_bit(u8Temp,7);
    }
}

bool Inno5Pro_CheckOddParity(uint8_t u8OddParity)
{
    u8OddParity ^= (u8OddParity >> 4);
    u8OddParity ^= (u8OddParity >> 2);
    u8OddParity ^= (u8OddParity >> 1);
    // Return 1 if the data has odd number of 1's. Otherwise return 0.
    return u8OddParity & 1;
}
```

Effects of VBEN State and System Reset to Secondary Timers and Registers

Before proceeding with I²C command Sequences, it is important to understand the behavior of InnoSwitch5-Pro IC during the VBEN-Disabled state and at a system reset event.

VBEN Disabled State and System Reset

For InnoSwitch5-Pro parts, the timers described below will be 8 times the programmed setting when VBEN is disabled. These will automatically update to match the programmed setting once VBEN is enabled. Note that a system reset event will also change VBEN to disabled which will also cause the timers below to be updated.

- Watchdog Timer (0x26)
- CVO Timer (0x0E)
- UVA Timer (0x94)
- Fast VI Timer (0x8C)

Similarly, the update delay of internal voltage reference due to a change in CV setpoint (is which discussed later in the Voltage Decrement Sequence section) will also be 8 times slower during the VBEN disabled state (compared to VBEN enabled state). However, a system reset event immediately changes the output voltage reference to the default of 5 V, disables the bus switch drive, and sets all command registers to their default values at the same time. Therefore, when a system reset event occurs while VOUT is above 5 V, the power supply will stop switching (since the internal voltage reference is immediately lower than VOUT). Enabling the Strong-Bleeder function with Auto-Disable can be used if needed.

Additionally, a system reset exhibits the following behaviors.

- System reset event while VOUT is above 9.6 V will also trigger an auto-restart (AR) since the OVA register will immediately return to its default value of 9.6 V threshold and AR response.
- Average IOUT, READ12 (0x18) and average VOUT, READ13 (0x1A) are 16-sample averages of their instantaneous telemetry registers READ8 and READ9, respectively. Both the average telemetry values will reset to 0 and will re-start accumulating a 16-sample average, therefore, after a system reset event, there will be some delay before the average Telemetry registers match the actual VOUT and IOUT values.
- Faults with a response set to Disable-Output will also result into a system reset when the fault is triggered. If its corresponding Interrupt Mask is enabled, the interrupt signal on SCL pin will be uncertain. It is not recommended to enable the Interrupt Mask for the faults that are configured to Disable-Output response.

A system reset will be initiated by each of the following actions:

1. I²C Write: VBEN (0x04) = 0x80 (Disable VBEN with Reset)
2. I²C Write: VDIS (0x08) = 0x83 (Enable Discharge, Disable VBEN with Reset)
3. Watchdog timer expiry (I²C commands not received within the programmable time interval)
4. Any secondary fault triggered with the response programmed to Disable-Output: CVO (0x0E), OVA (0x92), UVA (0x94), and ISSC (0xA2).
5. BPS voltage increasing from the BPS Pin undervoltage threshold ($V_{BPS(UVLO)TH}$, 3.8 V typical) to V_{BPS} (4.5 V typical) such as during an initial power-on event
6. uVCC voltage exceeding BPS voltage during externally applied abnormal operating conditions such as component or trace shorts.

Note that auto-restart or latch-off events, or uVCC voltage dropping below its reset threshold (uVCCRST, 2.5 V typical) and recovering to nominal level (uVCC, 3.6 V typical) do not inherently open the bus switch or initiate a system reset. The I²C master controller must send a command to open the bus switch if this is the desired behavior or program the watchdog timer with the desired time-out period which would trigger a system reset when it expires.

VBEN Disable with Reset Pseudo Code

It is recommended to set the output voltage register (CV) to 5 V prior to sending a VBEN disable with reset command to avoid unwanted auto-restart due to the assertion of overvoltage protection.

```
//VBEN Disable w/ Reset

//Set Output voltage to 5V
Voltage Decrement Sequence -> 5V

//Resets all registers to their default value
Set VBEN -> Disable w/ Reset (0x00)

Set VDIS -> Enable Discharge w/ Reset

Wait Delay in miliseconds

Set VDIS -> Disable
```

Note: *It is not required to transition to 5 V when using VBEN Disable without Reset. Simply sending the VBEN Disable without reset will be sufficient.*

Watchdog Timer

The InnoSwitch5-Pro ICs will initiate a system reset if I²C commands are not received within the programmable watchdog time interval. By default, the watchdog timer is set at 0.5 seconds. The watchdog timer does not engage until the master controller issues the first I²C command (read or write). During the VBEN-Disabled state, the watchdog timer period will be 8 times of the programmed value.

Disabling the watchdog feature (write 0x00 into the watchdog register 0x26) can be useful in initial software debugging or bench checking functionality of the device on the bench. For the final application (watchdog is not disabled) ensure there is recurring valid I²C communication with the InnoSwitch5-Pro IC that is sent faster than the watchdog timer.

```
#define INNO5PRO_WATCHDOG_TIMER_NOWATCHDOG          0x0
#define INNO5PRO_WATCHDOG_TIMER_500MS               0x1
#define INNO5PRO_WATCHDOG_TIMER_1000MS              0x2
#define INNO5PRO_WATCHDOG_TIMER_2000MS              0x3

void Inno5Pro_Initialization (void)
{
  //...
  // Configure Watchdog with user-preferred timer
  Inno5Pro_Write_Watchdog_Timer(
                          INNO5PRO_WATCHDOG_TIMER_500MS);
  //...
  //(Initialization Codes)
}
```

CVO Timer

When the Constant Voltage Only mode is enabled (CVO register 0x0E, bit[0] = 1) and the sensed load current in the IS pin exceeds the programmed threshold in the CC register (0x98), the peak load timer (CVO Timer bit[4:3] of the CVO register 0x0E) is started. If this condition is sustained for the duration programmed into the CVO timer, then a CVO fault will be triggered with the corresponding programmed response.

During VBEN-Disabled state, the timer for CVO fault detection will be 8 times the programmed value; however, the VBUSSC fault will also

be actively monitoring the sensed load current while VBEN is disabled. If the sensed current by the IS pin exceeds the current threshold programmed into the VBUSSC register (0xB6, bit[5:4]) for the set number of consecutive samples (0xB6, bit[3:2]), then a VBUSSC fault will be triggered. Therefore, during a VBEN-Disabled state, a VBUSSC fault detection protects the system faster than the CVO fault monitor.

UVA Timer

Output UV protection compares the VOUT pin voltage with the programmed UVA threshold (UVA register 0x94, bit[9:8] and bit[6:0]). Once the VOUT pin drops below the threshold, the UV Timer (0x94, bit[13:12]) is started. If this condition is sustained for the duration programmed in the UV Timer period, then a UV fault will be triggered with the corresponding programmed response.

During a VBEN-Disabled state, the timer for the UV fault will be 8 times of the programmed value. Additionally for InnoSwitch5-Pro ICs, there is an option either disable or enable the by setting the UVA register bit[14] to "0" or "1" respectively, regardless of VBEN state.

Fast VI Timer

By default, the CV (0x10) and CC (0x98) commands that program the output voltage/current respectively cannot be updated faster than a maximum speed limit of 10 milliseconds. During the VBEN-Disabled state, the timer for Fast-VI will be 8 times of the programmed value.

After the CV or CC command register is updated, any additional CV or CC command sent within the time limit will be ignored. The speed limit can be removed by setting the Fast VI register (0x8C) to 0x01.

Command Sequences

This section provides the recommended sequence for InnoSwitch5-Pro IC initialization and output voltage increment / decrement processes.

InnoSwitch5-Pro Initialization

The pseudo-code example below shows the recommended initialization sequence for InnoSwitch5-Pro devices. This sequence first ensures that all the command registers are at their default values by using the VDIS-Enable with a reset command. The power supply is then configured for 5 V operation. Some of the programmable fault responses are still configured with the default values for thresholds and response, but these can be adjusted if required.

```
void Inno5Pro_Initialization (void)
 {
        // Create a System Reset Event to ensure all registers are at default value
        // When this command is used while VO Pin Voltage > 9.6V, the OV Fault with AR response will trigger
        // VDIS Enable is used to also the ensure the discharge of the voltage after the bus switch
        // Set VDIS (0x08) = 0x83
        Inno5Pro_Load_Discharge (INNO5PRO_LOAD_DISCHARGE_ENABLE_WITH_RESET);

        // Configure Watchdog with user-preferred timer
        Inno5Pro_Write_Watchdog_Timer (INNO5PRO_WATCHDOG_TIMER_500MS);

        // Remove speed limit for updating CV and CC registers
        // Set FAST_VI (0x8C) = 0x01
        Inno5Pro_FastVI (INNO5PRO_FASTVI_UPDATE_LIMIT_DISABLE);

        // Read and store into a variable the End of Line Calibration Offset for Output Current Tolerance
        Inno5Pro_Store_EOL ( INNO5PRO_Read_EOL() );

        // Set Over Current Protection Threshold when CVO is enabled
        // CC Setpoint is adjusted with EOL Calibration Offset to improve tolerance
        Inno5Pro_Write_Amps_EOL (INNO5PRO_DEFAULT_CC_ASSERT_LEVEL, Inno5Pro_Get_EOL() );

        // Set Under Voltage Fault Threshold, Fault Response, Timer, and Timer Control
        Inno5Pro_Write_Under_Volts (INNO5PRO_DEFAULT_UVA_SET_PT_LEVEL,
                                    INNO5PRO_UVL_FAULT_RESPONSE_AUTORESTART,
                                    INNO5PRO_UVL_FAULT_TIMER_8MS,
                                    INNO5PRO_UVL_TIMER_ENABLE);

        // Set Output Voltage Setpoint to 5V
        Inno5Pro_Write_Volts (INNO5PRO_DEFAULT_CV_SET_PT_LEVEL);

        // Set Over Voltage Fault Threshold and Fault Response
        Inno5Pro_Write_Over_Volts (INNO5PRO_DEFAULT_OVA_SET_PT_LEVEL,
                                   INNO5PRO_OVL_FAULT_RESPONSE_AUTORESTART);

        // Set IS Pin (current sense) Short-Circuit Fault Response, Detection Frequency, and Fault Threshold
        Inno5Pro_Write_ISSC_Fault_Response (INNO5PRO_ISSC_FAULT_RESPONSE_NORESPONSE,
                                           INNO5PRO_ISSC_FREQ_THRESHOLD_60KHZ,
                                           INNO5PRO_ISSC_CC_THRESHOLD_64);

        // Enable CVO Mode and set the fault response and timer
        Inno5Pro_CVOnlyMode_Enable (INNO5PRO_CVOL_FAULT_CV_ONLY,
                                    INNO5PRO_CVOL_FAULT_RESPONSE_AUTORESTART,
                                    INNO5PRO_CVOL_FAULT_TIMER_8MS);

        // Disable the load discharge on VBD pin
        // Set VDIS (0x08) = 0x0E
        Inno5Pro_Load_Discharge (INNO5PRO_LOAD_DISCHARGE_DISABLE);

        // Set Switching Window Optimization (0x02) = 0x1F
        Inno5Pro_Switch_Window_Optimization (INNO5PRO_SWITCH_WINDOW_RECOMMENDED);

        // Set Transient Response 1 (0x32) = 0x140A
        Inno5Pro_Write_Loop_Speed1 (INNO5PRO_LOOPSPEED1_RECOMMENDED);

        // Set Transient Response 2 (0x34) = 0x1F84
        Inno5Pro_Write_Loop_Speed2 (INNO5PRO_LOOPSPEED2_RECOMMENDED);
}
```

Voltage Increment Sequence

The figure below illustrates the recommended command sequence for output voltage increment, with the assumption that the InnoSwitch5-Pro IC has already been initialized (as described in the previous section) and the bus switch is closed (VBEN enabled).

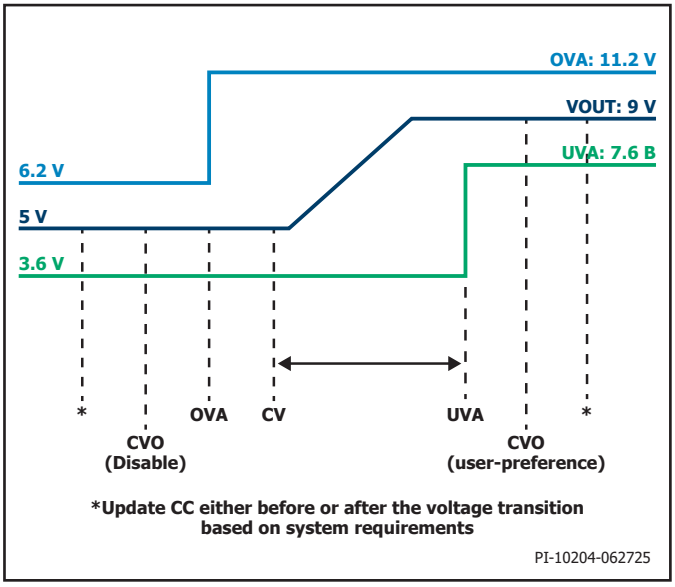


Figure 4. Recommended Command Sequence for VOUT Increment.

The CVO register (0x0E) is disabled by writing '0' into bit[0]. In contrast to the response when CVO is enabled, changing the CVO setting to disabled will slow down the rate of change of output voltage (based on the transient response registers) and may prevent unintentional CCM cycles and associated increased SR FET voltage stress during upward voltage transitions under heavy load.

Prior to increasing the CV Setpoint, the OVA Threshold must be increased to its new value (higher than the new target VOUT plus margin) to prevent the OV Fault from asserting when the output voltage rises.

After writing the new CV Setpoint, the system must wait for the output voltage to reach the new value before updating the UVA Threshold. This will prevent the UV Fault from asserting as the output rises. The simplest method is a firmware delay based on the magnitude of voltage increment (refer to Voltage Increment pseudo -code section). With the CVO disabled and with the Transient Response registers set to the recommended values, a time delay of ~5ms per volt is sufficient. Other methods can also be used, such as allowing the external microcontroller to monitor the actual output voltage to determine if the new target voltage has been reached.

Once the output voltage has settled to the new value, the UVA Threshold can be updated, and the CVO Register can be configured. For example, with a USB-PD Fixed Supply output, the CVO can be reset to enbaled to reactivate output overcurrent protection. However, with a USB-PD PPS output, the CVO needs to remain disabled to have a constant-current output profile.
The CC Setpoint, it can be adjusted either before or after the voltage transition. For example, if the output is being changed from 5 V / 3 A into 9 V / 2 A and the CC setpoint is updated after the transition has been completed, there will be an intermediate condition where the programmed setpoints are 9 V / 3 A. In contrast if the CC setpoint is updated before the transition, the intermediate condition will be 5 V / 2 A. The end-user can adjust when the CC setpoint is updated based on which behavior is suitable for the application. A similar analysis can be used for determining when to update the ISSC and CDC registers.

Voltage Increment Pseudo Code

The pseudo-code below shows the commands and delay sequence for output voltage increment. The OVA and UVA thresholds are set to 120% and 85% of the new CV setpoint respectively.

```
//Voltage Increase Routine

//CVO (0x0E) bit[0] = 0 (Disable CVO)
Set CVO -> Disable

Set OVA -> (New CV Setpoint) * 1.20

Set CV  -> New CV Setpoint

//Compute Delay in ms based on voltage difference
//Setpoints are in 10mV/LSB
//Effective delay = 5ms per volt increase
//Limit minimum delay to certain value (e.g. 6ms)
Compute Delay in miliSeconds ->
      (Previous CV Setpoint – New CV Setpoint)/20

Wait  Delay in miliSeconds

Set UVA -> (New CV Setpoint) * 0.85

//Set CVO based on user preference
Set CVO -> Enable, 8ms, AR

//Set CC at end of transition
//Adjust placement based on user preference
Set CC -> New CC Setpoint
```

Voltage Decrement Sequence

Figure 5 below illustrates the recommended command sequence for output voltage decrement, with the assumption that InnoSwitch5-Pro IC has already been initialized based on the process described in the previous section and the bus switch is closed (VBEN enabled).

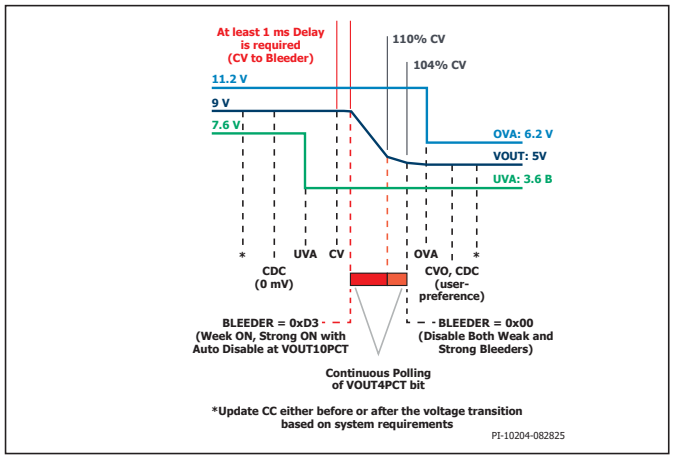


Figure 5. Recommended Command Sequence for VOUT Decrement.

The CDC Register is set to 0 mV to later simplify the later monitoring of output voltage. With CDC disabled, the output voltage becomes independent of the load current, and the microcontroller can either use the VOUT4PCT telemetry (READ10, bit[5]) or independently monitor the output voltage.

Prior to decreasing the CV Setpoint, the UVA Threshold must be decreased to its new value (lower than the new target VOUT with appropriate margin) to prevent the UV Fault from asserting before the output voltage falls.

The timing between the CV and Bleeder commands is critical to ensure proper operation for voltage decrements. The VOUT pin bleed function is only intended to discharge the flyback output capacitor close to the new voltage for large transitions (e.g. >500 mV decrements). If the Bleeder is not used, it may take a long time to discharge the output capacitor voltage and an auto-restart may be triggered or the desired time requirements for voltage transition not achieved. It is necessary to avoid overlap between power supply switching and the strong Bleeder being active. Writing to the CV register takes up to 1 ms for the internal reference to update when VBEN is enabled, which will also be the point at which the power supply stops switching (VOUT will be higher than the reference). Therefore, after updating the CV Setpoint, a delay of at least 1 ms is required before enabling the Bleeder.

The Bleeder register (0x86) must be enabled with a value of 0xD3 only. The Weak Bleeder is a ~2.5 mA current sink on VOUT pin that turns on/off when the VOUT is greater than or below ~104% of the target regulation voltage. The Strong Bleeder is a ~270 mA current sink on the VOUT pin that turns on when the appropriate I²C command is received (Bleeder 0x86 = 0xD3). It turns off when VOUT drops below ~110% of the target regulation voltage. By sending the Bleeder command after a 1 ms delay from the CV command, the overlap between Strong Bleeder and power supply switching is avoided and results in a successful downward output voltage transition.

After sending the Bleeder Enable command, the MCU must continuously monitor the output voltage tp determine if it has reached the new target voltage before adjusting the OVA threshold. This prevents unwanted triggering of OV fault protection. One method is for the MCU to indepedently monitor VOUT through its own peripherals. Another method is by polling the telemetry VOUT4PCT bit from the InnoSwitch5-Pro IC READ10 (0x14, bit[5]).

Note that using the command Bleeder 0x86 = 0xD3 will initiate a fast output discharge due to the Strong Bleeder, which will only be disabled when 110% of target voltage is reached (see region in red in Figure 5). Upon crossing VOUT10PCT, the Weak Bleeder remains on and helps further discharge VOUT to 104% of target voltage (see orange region from Figure 5). Once VOUT4PCT is reached, the next I²C commands can be written – Bleeder 0x86 = 0x00 to disable the Weak Bleeder, and then set the new target OVA threshold. The same analysis can be used whether output load is present or not. The presence of an output load simply makes the downward voltage transition faster.

Once the output voltage has settled, the CVO and CDC registers can then be configured. For example, with a USB-PD Fixed Supply output, the CVO can be set as Enabled to have output overcurrent protection together with a certain CDC value. However, with a USB-PD PPS output, the CVO needs to remain as Disabled to have a constant-current output profile and CDC is typically set to 0 mV to meet output regulation limits. For the CC Setpoint and ISSC register update, the same analysis as described in the previous section can be applied. Updates to these registers is based on the intermediate value acceptable in the end-user application.

Voltage Decrement Pseudo-Code

The pseudo-code below shows the commands for output voltage decrement. The UVA and OVA thresholds are set to 85% and 120% of the new CV setpoint respectively.

```
//Voltage Decrease Routine

//Disable CDC for simple monitoring
//when VOUT decreased to new value
Set CDC -> 0 mV

Set UVA -> (New CV Setpoint) * 0.85

Set CV  -> New CV Setpoint

//At least 1 ms delay required from CV to Bleeder
Wait Delay -> 1ms

//Set Bleeder = 0xD3 (Weak Bleeder Enable,
//                    Strong Bleeder Enable with
//                    Auto-Disable at VOUT10PCT)
Set BLEEDER -> 0xD3

Poll Read10 bit[5] -> Until VOUT4PCT bit = 0

//Set Bleeder = 0x00 (Weak Bleeder Disable,
//                    Strong Bleeder Disable)
Set BLEEDER -> 0x00

Set OVA -> (New CV Setpoint) * 1.20

//Set CVO and CDC based on user preference
Set CVO -> Enable, 8ms, AR

Set CDC -> 200 mV

//Set CC at end of transition
//Adjust placement based on user preference
Set CC -> New CC Setpoint
```

Line Sensing Sequence

The InnoSwitch5-Pro IC has a primary switch and secondary synchronous rectifier conduction-time reporting which can be used to estimate the input line voltage. The primary switch on-time is reported as TON on READ21 (0x2A) and the secondary synchronous rectifier conduction time is reported as TOFF on READ22 (0x2C).

The TON and TOFF registers will only start to be updated once 0x01 is sent to the Line Sense Trigger register (0x1C). The Line Sense Report flag will be set after 16 samples are accumulated for both registers. The line sense value can be calculated shown below.

$$V_{IN} = \frac{N_P}{N_S} \times (V_{OUT} + V_{DS(SR)}) \times \frac{T_{OFF}}{T_{ON}}$$

The TON value needs to be subtracted by b'64 to account for system delays.

The accuracy of the line sense feature is dependent on the SR conduction period which will be compromised at light loads. Before estimating values, it is recommended that the user characterize the line voltage across different operating conditions. The SR ZVS values need to be optimized before using the Line Sense Enable command during the this mode of operation.

Note: *This equation is valid for both SR ZVS and QR mode of operation*

Example 1

At 30 V output, N_P = 25T, & N_S = 5T

Obtained T_{ON} = 0x062E or d'1582
Obtained T_{OFF} = 0x03F2 or d'1010

$$V_{IN} = \frac{25}{5} \times 30\ V \times \frac{1010}{1518}$$

$$V_{IN} = 99.8\ V$$

```
//Line Sensing

//Trigger Line Sensing Trigger Register 0x1C
Set LS -> 0x01

//Poll Line Sense Ready Register bit
while (READ10 BIT [12]
{
delay_ms (5ms)
}

Set CV  -> New CV Setpoint

//Read Line Sense TON register
Read READ21 bit [11:0]

//Read Line Sense TOFF register
Read READ22 bit [15:0]
```

Telemetry / Read Back

The InnoSwitch5-Pro IC communicates with the I²C Master (external MCU) to report back the status of the power supply. The telemetry features include the CV, CC, and constant power setpoints, OV/UV thresholds, all programmable protection settings, interrupt status, and individual fault status. The InnoSwitch5-Pro family datasheet provides the complete list of available telemetry register assignments.

Prior to using the telemetry, the I²C Read/Write drivers of the microcontroller must be configured appropriately. The pseudo-codes in this section are in C programming language, which is used in the InnoSwitch5-Pro IC code library for the Microchip PIC16F family.

System Ready Signal

When starting from an initial power-up, the microcontroller should monitor the System Ready bit (READ10, bit14). A value of "1" indicates the InnoSwitch5-Pro is IC ready to communicate and accept further commands. The code example below shows a function reading back the System Ready bit.

System Ready – Code Example

```
#define READ10_Reg_CONTROL_S 14

bool Inno5Pro_Read_Status_SystemReady (void)
{
    //READ10 bit14, System Ready Signal
return InnoProBase_Read_Bit (INNO5PRO_READ10,
                             READ10_Reg_CONTROL_S);
{
```

I²C Read Back Code Examples

Read Word (16-bit) Telemetry

InnoProBase_Telemetry function is an API for reading 16-bits data from the desired telemetry register address. It uses the I²C_Read16 function which is an I²C driver created for the InnoSwitch5-Pro family and it returns the high and low bytes appended together in a 16-bits format. The return value from this function is simply the 16-bits raw data as it is, regardless of the presence of any parity bits.

```
uint16_t InnoProBase_Telemetry(uint8_t u8ReadBack_Address)
{
   uint16_t u16TempRead = 0;

  //I2C_Read16 reads 16 bits of data
  u16TempRead = I2C_Read16(INNOPRO_I2C_ADDRESS,
                           u8ReadBack_Address);

  return u16TempRead;
}
```

Read Byte (8-bit) Telemetry

InnoProBase_Read_Byte function is an API for reading either the high byte or the low byte from the desired telemetry register address.

```
uint8_t InnoProBase_Read_Byte (uint8_t u8ReadBack_Address,
                               bool bHighByte)
{
   uint16_t u16TempRead = 0;

   //I2C_Read16 reads 16 bits of data
   u16TempRead = I2C_Read16(INNOPRO_I2C_ADDRESS,
                            u8ReadBack_Address);

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

Read Bit Telemetry

InnoProBase_Read_Bit function is an API for reading the desired bit of a telemetry register. This function returns a value of either 1 or 0.

```
#define test_bit(VAR,BIT)                    (VAR & (1<<BIT))

bool InnoProBase_Read_Bit (uint8_t u8ReadBack_Address,
                           uint8_t u8Bit)
{
   uint16_t u16TempRead = 0;
   //I2C_Read16 reads 16 bits of data
   u16TempRead = I2C_Read16(INNOPRO_I2C_ADDRESS,
                            u8ReadBack_Address);

   //return the value of the bit
   return (test_bit(u16TempRead,u8Bit));
}
```

VOUT10PCT Bit – Code Example

The VOUT10PCT bit is automatically set whenever the output voltage is above 10% of the target voltage (CV Setpoint). Otherwise, VOUT10PCT bit is automatically cleared.

```
#define READ10_Reg_VOUT10PCT 4

bool Inno5Pro_Read_Status_Vout10pct (void)
{
    //READ10 bit4, VOUT > 1.1 * target regulation voltage
    return Inno5Pro_Read_Bit (INNO5PRO_READ10,
                              READ10_Reg_VOUT10PCT);
}
```

VOUT4PCT Bit – Code Example

The VOUT4PCT bit is automatically set whenever the output voltage is above 4% of the target voltage. Otherwise, VOUT4PCT bit is automatically cleared.

With InnoSwitch5-Pro devices, writing 4b'0 into the high nibble (bit[7:4]) of the Bleeder command register (address 0x86) disables the Weak Bleeder function and will result in VOUT4PCT bit to always be cleared. The VOUT4PCT bit will again become dependent on output voltage once the Weak Bleeder is enabled, e.g. writing into the Bleeder command register with the high nibble set to 0xD and the low nibble configured based on the desired Strong Bleeder operation.

```
#define READ10_Reg_VOUT4PCT 5

bool Inno5Pro_Read_Status_Vout4pct(void)
{
  //READ10 bit5, VOUT > 1.04 * target regulation voltage
  return Inno5Pro_Read_Bit (INNO5PRO_READ10,
                            READ10_Reg_VOUT4PCT);
}
```

Line Sense Reporting Ready Bit – Code Example

The line sense Reporting Ready bit is automatically set once the secondary controller line sense measurement has been completed and the line sense TON and TOFF Telemetry registers have been updated with their 16-sample accumulated values.

```
#define READ10_Reg_LS_READY 12

bool Inno5Pro_Read_Status_LineSense (void)
{
    //READ10 bit5, VOUT > 1.04 * target regulation voltage
    return Inno5Pro_Read_Bit (INNO5PRO_READ10,
                              READ10_Reg_LS_READY);
}
```

Read Setpoint and Threshold

Inno5Pro_Read_SetPoint function reads the specified telemetry. The register and returns its corresponding 16-bits data with the odd parity (bit[15] and bit[7]) removed. This function can be used to analyze the result from telemetry registers that contain odd parity in their raw data such as READ1, READ2, READ3, and READ4.

```
uint16_t InnoProBase_Read_SetPoint (uint16_t
                                    u16ReadBack_Address)
{
   uint16_t u16 TempRead Value = 0;
   uint16_t u16 Converted Value = 0;

   u16TempReadValue = InnoProBase_Telemetry (
                                   u16ReadBack_Address);

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

   return u16ConvertedValue;
}
```

Read Voltage – Instantaneous and Average

The two functions provide the decimal equivalent data from the instantaneous and average output voltage telemetry registers. As specified in InnoSwitch5-Pro family datasheet, these telemetry registers have a report-back step size that depends on the output voltage. The decimal result can be mapped to the actual output voltage by dividing it by 10 mV (e.g READ9 data = 0x01F4 -> d'500 -> 5.00 V).

```
#define INNO5PRO_READ9      0x12

uint16_t Inno5Pro_Read_Measured_Volts(void)
{
   uint16_t u16TempReadValue = 0;
   uint16_t u16ConvertedValue = 0;

   //Read instantaneous output voltage
   u16TempReadValue = InnoProBase_Telemetry(
                                          INNO5PRO_READ9);

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

   //Return the masked result, e.g. 0x01F4 -> d'500 -> 5.00V
   return u16ConvertedValue;
}
```

```
#define INNO5PRO_READ13     0x1A

uint16_t Inno5Pro_Read_VoltsAverage(void)
{
   uint16_t u16TempReadValue = 0;
   uint16_t u16ConvertedValue = 0;

   //Read average output voltage
   u16TempReadValue = InnoProBase_Telemetry(
                                          INNO5PRO_READ13);

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

   //Return the masked result, e.g. 0x01F4 -> d'500 -> 5.00V
   return u16ConvertedValue;
}
```

Read Current – Instantaneous and Average

The two functions below provide the data converted into mA units from the instantaneous and average output current telemetry registers. The macro for CC bit resolution is provided via the Constant Current register, CC (0x98) section which is set based on the effective R_{SENSE} seen by the IS-GND pins. The data from the telemetry register is mapped to the actual output current by multiplying it with the CC bit resolution.

```
#define INNO5PRO_READ8                          0x10
// Rsense ~6.1mohm
// CC Bit Resolution (32mV/Rsense)*(1/192LSB)
// 27.3 mA/LSB for Rsense 6.1mohm
#define INNO5PRO_CC_BIT_RESOLUTION (273)

uint16_t Inno5Pro_Read_Amps (void)
{
    uint16_t u16TempReadValue = 0;
    uint16_t u16ConvertedValue = 0;

    //Read instantaneous output current
    u16TempReadValue = InnoProBase_Telemetry(
                                                INNO5PRO_READ8);

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

    //Example READ8 data = 183 decimal
    //Output Current     = 183 * CC bit resolution
    //                   = 183 * 27.3mA/LSB
    //                   = 183 * 273 / 10
    //                   = 4995 mA
return (    (u16ConvertedValue *
               INNO5PRO_CC_BIT_RESOLUTION) / 10);
}
```

```
#define INNO5PRO_READ12                         0x18
// Rsense ~6.1mohm
// CC Bit Resolution (32mV/Rsense) * (1/192LSB)
// 27.3 mA/LSB for Rsense 6.1mohm
#define INNO5PRO_CC_BIT_RESOLUTION (273)

uint16_t Inno5Pro_AmpsAverage(void)
{
    uint16_t u16TempReadValue = 0;
    uint16_t u16ConvertedValue = 0;

    //Read instantaneous output current
    u16TempReadValue = InnoProBase_Telemetry (
                                          INNO5PRO_READ12);

    //Clear bits [15:8]
    u16ConvertedValue = (u16TempReadValue & 0x00FF);

    //Example READ12 data = 183 decimal
    //Output Current     = 183 * CC bit resolution
    //                   = 183 * 27.3mA/LSB
    //                   = 183 * 273 / 10
    //                   = 4995 mA
    return ( (u16ConvertedValue *
               INNO5PRO_CC_BIT_RESOLUTION) / 10);
}
```

Code Library Overview

To simplify the use of InnoSwitch5-Pro devices, a simple code library for Microchip PIC16 family and Arduino platform 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. The computation macros are presented to aid in setpoint calculations. The register default values are also included to assist writing into the required registers for device initialization.

PIC16F18325 MCU Implementation

Implementation

Header Files Inclusion

The Library header files contain all 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 "Inno5Pro.h"
#include "Inno5Pro_Config.h"
```

InnoSwitch5-Pro Initialization

Before the continuous execution of the main code, the status of System Ready Signal is monitored to ensure the InnoSwitch5-Pro IC is ready to receive I²C commands. Afterwards, initialization commands can be sent to the device to re-configure the default settings as needed.

```
void main(void)
{
    //...
    Inno5Pro_Initialization();
    //...
    // (Main Loop Codes)
}
```

Control Functions Set-up

Some example control functions are listed below, and the complete list can be found in the source code library.

Sets the Output Voltage Setpoint

```
Inno5Pro_Write_Volts (u16SetPtCV);
```

Sets the Constant Current Setpoint

```
Inno5Pro_Write_Amps(u16SetPtCCmA);
```

Sets the Over Voltage Threshold and Response

```
Inno5Pro_Write_Over_Volts (u16SetPtOVA,
                           u16Ov_FaultResp);
```

Sets the Under Voltage Threshold, Response, Timer Value and Timer Control

```
Inno5Pro_Write_Under_Volts (u16SetPtUVA,
                            u16Uv_FaultResp,
                            u16Uv_timer,
                            u16Uv_timer_En);
```

Sets the Cable Drop Compensation Value

```
Inno5Pro_Write_Cable_Drop_Comp(u16SetPtCDC);
```

Sets the Constant Output Power Threshold

```
Inno5Pro_Write_Volt_Peak (u16SetPtVKP);
```

Used for Turning On or Off the Bus Voltage Switch

```
Figure 6.Inno5Pro_Vbus_Switch_Control (u16VbenControl
```

Used for Turning On or Off the VOUT pin Weak and Strong Bleeder

```
Inno5Pro_Bleeder_Control (u16BleederControl)
```

Used to perform an output transition wherein:

- Proper sequence and timing of I²C commands are used to update CC and CV setpoint while preventing inadvertent triggering of UV or OV protection.
- VOUT pin Weak and Strong Bleeder are controlled when decreasing into a lower voltage setpoint.
- OverVoltage (OVA) and Under Voltage (UVA) thresholds are updated into new values:
    1. OVA is 120% of new CV set-point
    2. UVA is 85% of new CV set-point

```
Inno5Pro_PD_Write_VI (u16SetPtCV, u16SetPtCCmA);
```

Telemetry Functions Setup

Some example telemetry functions are listed below, and the complete list can be found in the source code library.

Used for reading the desired Register Address

```
Inno5Pro_Telemetry(u16Register_Address);
```

Used for reading the specific bit from the Register Address

```
Inno5Pro_Read_Bit(u16Register_Address, u8Bit);
```

Returns the Reg_Control_S bit value to identify when InnoSwitch5-Pro is ready to communicate and accept commands

```
Inno5Pro_Read_Status_SystemReady();
```

Returns the instantaneous measured output voltage

```
Inno5Pro_Read_Volts();
```

Returns the instantaneous measured output current

```
Inno5Pro_Read_Amps
```

Returns the VOUT10PCT bit value

```
Inno5Pro_Read_Amps();
```

Returns the VOUT4PCT bit value

```
Inno5Pro_Read_Status_Vout4pct();
```

Returns the Line Sense Reporting Ready bit value

```
Inno5Pro_Read_Status_LineSense();
```

Basic Code Example

This code example is to demonstrate the basic usage of the InnoSwitch5-Pro family code library.

- Initial commands are sent using the InnoSwitch5-Pro IC initialization routine.
- The main routine sets the output voltage to 5 V and constant current to 5 A.
- Cable Drop Compensation (CDC) is programmed to 300 mV.
- 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/Inno5Pro_Config.h"
#include "Code/Inno5Pro.h"

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

    MSSP_HostInit();

    //Step 2: Write Initialize Commands to InnoSwitch5-Pro
    Inno5Pro_Initialization();

    //Step 3: Call functions on the Main Loop
    while(1)
    {
        //Main Loop Variable Initialization
        //Initialize Output Voltage at 5V
        uint16_t u16Volts = 500;
        //Initialize Constant Current at 5A
        uint16_t u16Amps = 5000;

        //Library Call in the Mainloop
        //Set Voltage and Current
        Inno5Pro_PD_Write_VI ( u16Volts, u16Amps     );
        //Set Cable Drop Compensation
        Inno5Pro_Write_Cable_Drop_Comp(
                                    INNO5PRO_CDC_SET_PT_300MV);
        //Set Vbus Enable
        Inno5Pro_Vbus_Switch_Control      (INNO5PRO_VBUS_ENABLE);
    }
}
```

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 InnoSwitch5-Pro family datasheet for read and write transactions. I²C transactions must have at least a 150µs delay between commands.

I²C Write Code Example

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

   uint8_t writeBuffer[3];

   //Copy Command/Telemetry Register Address to buffer
   writeBuffer[0] = dataAddress;

   if (buflen > 3)
{
   buflen = 3;
}

   //Copy Data Bytes to buffer
   writeBuffer[1] = dataBuffer[0];
   writeBuffer[2] = dataBuffer[1];

   //Initiate Write to Device
   MSSP_WriteBlock (slaveAddress,
                    writeBuffer,
                    buflen);

}
```

I²C Read Code Example

```
uint16_t I2C_Read16 (uint8_t slaveAddress,
                     uint8_t dataAddress,
                     uint8_t *dataBuffer,
                     uint8_t buflen)
{
//150us delay on every I2C transaction
delayMicroseconds(150);

uint8_t buflen = 0x02;
uint8_t readDataBuffer[2];
uint16_t u16Lsb;
uint16_t u16Msb;

//Routine for I2C Write 0x80, Start , End
//followed by I2C Read LSB MSB
//Results stored in readDataBuffer
MSSP_RegisterSelectAndRead (slaveAddress,
                            dataAddress,
                            readDataBuffer,
                            buflen);

//Example  5V,  Returns F4
u16Lsb = readDataBuffer[0];

//Example  5V,  Returns 01
u16Msb = readDataBuffer[1];

//Returns 01F4
return ((u16Msb<<8)|(u16Lsb));
}
```

Arduino Implementation

Implementation

Header Files Inclusion
The library header files contain all 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 "Inno5Pro.h"
#include "Inno5Pro_Config.h
```

Class Instance Creation
Construct a class instance to call the functions inside Inno5Pro_ Application. Constructing a class instance of Inno5Pro_Rtc is optional.

```
Inno5Pro_Application Inno5ProApp;
Inno5Pro_Rtc        Inno5ProClk;
```

InnoSwitch5-Pro Initialization
Before the continuous execution of the main code, the status of System Ready Signal is monitored to ensure the InnoSwitch5-Pro IC is ready to receive I²C commands. Afterwards, initialization commands can be sent to the device to re-configure the default settings as needed.

The 400 kHz clock frequency for the I²C communication is set-up on initialization.

```
void setup()
{
    Inno5ProApp.Inno5Pro_Initialization();
}
```

Control Functions Set-up
Some example control functions are listed below, and the complete list can be found in the source code library.

Sets the Output Voltage Setpoint

```
Inno5ProApp.Inno5Pro_Write_Volts (u16SetPtCV);
```

Sets the Constant Current Setpoint

```
Inno5ProApp.Inno5Pro_Write_Amps (u16SetPtCCmA);
```

Sets the Overvoltage Threshold and Response

```
Inno5ProApp.Inno5Pro_Write_Over_Volts (u16SetPtOVA,
                                       u16Ov_FaultResp);
```

Sets the Undervoltage Threshold, Response, Timer Value and Timer Control

```
Inno5ProApp.Inno5Pro_Write_Under_Volts(
                                       u16SetPtUVA,
                                       u16Uv_FaultResp,
                                       u16Uv_timer,
                                       u16Uv_timer_En);
```

Sets the Cable Drop Compensation Value

```
Inno5ProApp.Inno5Pro_Write_Cable_Drop_Comp(
                                       u16SetPtCDC);
```

Sets the Constant Output Power Threshold

```
Inno5ProApp.Inno5Pro_Write_Volt_Peak (u16SetPtVKP);
```

Used for turning On or Off the Bus Voltage Switch

```
Inno5ProApp.Inno5Pro_Vbus_Switch_Control(
                                       u16VbenControl);
```

Used for turning On or Off the VOUT pin Weak and Strong Bleeder

```
Inno5ProApp.Inno5Pro_Bleeder_Control(
                                       u16BleederControl);
```

Used to perform an output transition wherein:
- Proper sequence and timing of I²C commands are used to update CC and CV setpoint while preventing any inadvertent triggering of UV or OV faults.
- VOUT pin weak and strong Bleeder are controlled when decreasing into a lower voltage setpoint.
- Overvoltage (OVA) and Undervoltage (UVA) thresholds are updated into new values:
  1. OVA is 120% of new CV set-point
  2. UVA is 85% of new CV set-point

```
Inno5ProApp.Inno5Pro_PD_Write_VI (u16SetPtCV,
                                       u16SetPtCCmA);
```

Telemetry Functions Setup
Some example telemetry functions are listed below, and the complete list can be found in the source code library.

Used for reading the desired register address

```
Inno5ProApp.Inno5Pro_Telemetry(
                                       u16Register_Address);
```

Used for reading the specific bit from the Register Address

```
Inno5ProApp.Inno5Pro_Read_Bit (u16Register_Address,
                                       u8Bit);
```

Returns the Reg_Control_S bit value to identify when InnoSwitch5-Pro is ready to communicate and accept commands

```
Inno5ProApp.Inno5Pro_Read_Status_SystemReady ();
```

Returns the instantaneous measured output voltage

```
Inno5ProApp.Inno5Pro_Read_Volts ();
```

Returns the instantaneous measured output current

```
Inno5ProApp.Inno5Pro_Read_Amps ();
```

Returns the VOUT10PCT bit value

```
Inno5ProApp.Inno5Pro_Read_Status_Vout10pct ();
```

Returns the VOUT4PCT bit value

```
Inno5ProApp.Inno5Pro_Read_Status_Vout4pct ();
```

Returns the Line Sense Reporting Ready bit value

```
Inno5ProApp.Inno5Pro_Read_Status_LineSense ();
```

Basic Code Example

```
//Step 1: Add Header Files
#include "Code/Drv_i2c.h"
#include "Code/Drv_Rtc.h"
#include "Code/Inno5Pro_Config.h"
#include "Code/Inno5Pro.h"

//Step 2: Create the class instance
Inno5Pro_Application Inno5Pro;

//Step 3: Write initial commands to InnnoSwitch5-Pro
void setup(void)
{
    Inno5ProApp.Inno5Pro_Initialization ();
}

//Step 4: Call the functions on the main loop
void loop()
{
   //Output Voltage and Constant Current Setpoint 5V, 5A
   Inno5ProApp.Inno5Pro_Write_VI(500, 5000);

   //Cable Drop Compensation 300 mV
   Inno5ProApp.Inno5Pro_Write_Cable_Drop_Comp(
                                INNO5PRO_CDC_SET_PT_300MV);

   //Vbus Enable
   Inno5ProApp.Inno5Pro_Vbus_Switch_Control (
                                        INNO5PRO_VBUS_ENABLE);
}
```

I²C Drivers

The I²C 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 I²C packet format in the InnoSwitch5-Pro family data sheet for write and read transactions. Every I²C transaction must have at least a 150 μs delay between commands.

I²C Write Code Example

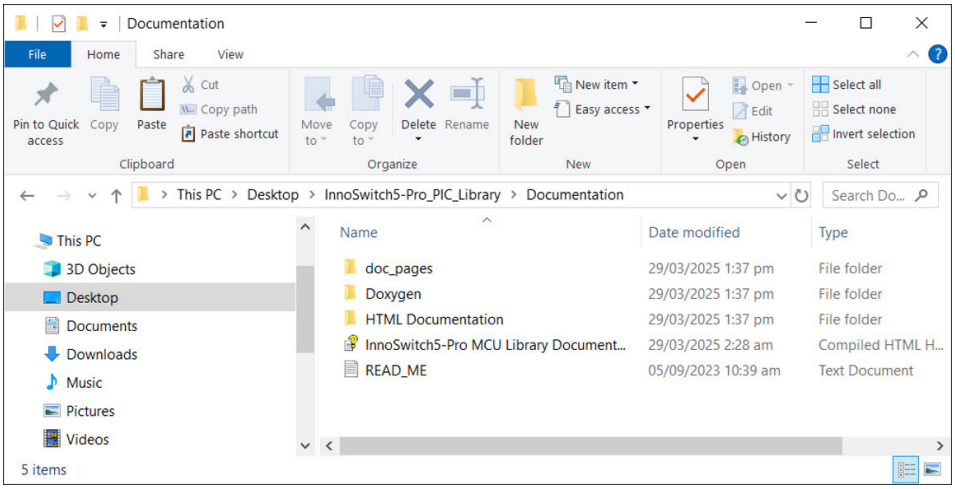
```
void Inno5Pro_I2C::I2C_Write16 (uint8_t slaveAddress,
                                uint8_t dataAddress,
                                uint8_t *dataBuffer,
                                uint8_t buflen)
{
   //150us delay on every I2C transaction
   delayMicroseconds(150);
   Wire.beginTransmission((uint8_t)slaveAddress);

//I2C driver depends on Arduino board
#if ARDUINO >= 100
   //Send register address followed by data bytes
   Wire.write((uint8_t)dataAddress);
   Wire.write((uint8_t)dataBuffer[0]);
   if (buflen == 3)
   {
                        Wire.write((uint8_t)dataBuffer[1]);
   }
#else
   //Send register address followed by data bytes
   Wire.send((uint8_t)dataAddress);
   Wire.send((uint8_t)dataBuffer[0]);
   if (buflen == 3)
   {
     Wire.send((uint8_t)dataBuffer[1]);
   }
#endif

   Wire.endTransmission ();
}
```

I²C Read Code Example

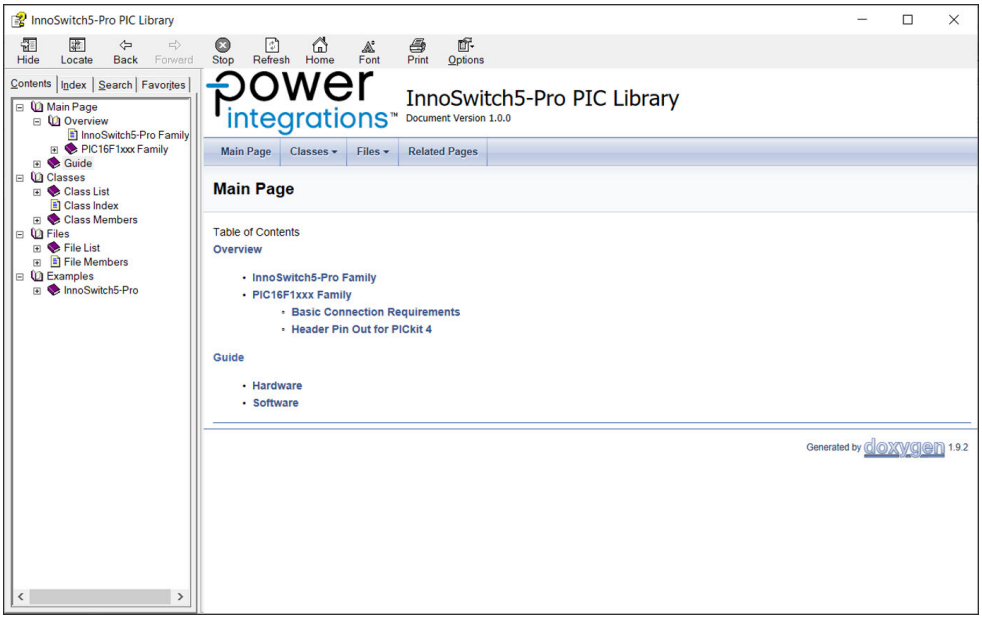
```
void Inno5Pro_I2C::I2C_Read16 (uint8_t slaveAddress,
                               uint8_t dataAddress,
                               uint8_t *dataBuffer,
                               uint8_t buflen)
{
    //150us delay on every I2C transaction
    delayMicroseconds(150);
    //Storage for LSB and MSB
    uint8_t u8Lsb;
    uint8_t u8Msb;

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

#if (ARDUINO >= 100)
   //Send 0x80 followed by Start and End address
   Wire.write(0x80);
   Wire.write(dataAddress);
   Wire.write(dataAddress);
#else
   //Send 0x80 followed by Start and End address
   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 ();

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

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

   Wire.endTransmission ();

   //Returns 0x01F4
   return ((u8Msb<<8) | (u8Lsb));
}
```

Documentation

The code documentation in .html and .chm file formats can be found in the InnoSwitch5-Pro PIC library in the Documentation folder.



The .chm file serves as a manual that provides an overview and overall setup for programming the InnoSwitch5-Pro device using the MPLAB IDE. The documentation includes all the function descriptions, macros, and code examples used in the InnoSwitch5-Pro PIC Library. The file contains a navigation panel on the left to assist in browsing through the different sections.



Revision	Notes	Date
A	Production release.	12/25

For the latest updates, visit our website: www.power.com

For patent information, Life support policy, trademark information and to access a list of Power Integrations worldwide Sales and engineering support locations and services, please use the links below.

https://www.power.com/company/sales/sales-offices