# Developer Notes
# iMeter

**Version 002**

**July 24, 2013**

## Revision History

| Rev | Date | Comments |
|---|---|---|
| 001 | 2/1/11 | Initial Release |
| 002 | 7/24/13 | Updated Commands |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Table of Contents

**1 FIRMWARE DESCRIPTION .................................................. 3**

1.1 INSTEON Commands Supported .................................................. 3

1.1.1 Standard length common INSTEON commands: .................................................. 3

1.1.2 Extended length iMeter INSTEON commands: ........................................ **Error! Bookmark not defined.**

2 Checksum

3 Memory Map

4 Timing

# 1 Firmware Description

The iMeter Solo tracks energy use of any plug-in appliance and is collected by HouseLinc or other compatible INSTEON control software/hardware. Data is gathered into a .csv allowing you to create graphs and track energy consumption.

## 1.1 INSTEON Commands Supported

### 1.1.1 Standard length common INSTEON commands:

All direct commands will be ignored if the sender's ID is not in the I2CS device's database with the exceptions below. The Door Sensor will reply with a NAK and 0xFF in cmd2 to indicate that the ID is not in the database.

**Assign to ALL-Link Group Command**

Description: Sent when holding down the SET Button for 3 seconds on the device. Blinks the LED green for 4 minutes or until linked to another device.

Example (Hex): AA BB CC XX YY ZZ CF 01 DD (where AA.BB.CC is the Door Sensor's ID)

| SD Command | Message Direction | From Address (3 bytes) | To Address (3 bytes) | Message type | Cmd1 (1 byte) | Cmd2 (1 byte) | Notes |
|---|---|---|---|---|---|---|---|
| **Assign to ALL-Link Group** | From Device | Device's ID | 0xXX (DevCat),<br>0xYY (SubCat),<br>0xZZ (firmware revision) | Broadcast | 0x01 | 0xDD (hardware revision) | Sent when holding down SET Button for 3 seconds.<br>Group number for Door Sensor is 0x01 |

**Delete from ALL-Link Group Command**

Description: Blinks the LED red for 4 minutes or until unlinked from another device.

Example (Hex): AA BB CC XX YY ZZ CF 02 DD (where AA.BB.CC is the Door Sensor's ID)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Delete from ALL-Link Group** | From Device | Device's ID | 0xXX (DevCat),<br>0xYY (SubCat),<br>0xZZ (firmware revision) | Broadcast | 0x02 | 0xDD (hardware revision) | Group number for Door Sensor is 0x01 |

**INSTEON Engine Version Command**

Description: Returns the INSTEON Engine version in the acknowledgement.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **INSTEON Engine Version** | To device | Sender's ID | Device's ID | Direct | 0x0D | 0x00 | |
| | Response | Device's ID | Sender's ID | Ack | 0x0D | 0x01 | Indicate i2 engine version |

**Ping Command**

Description: Returns the exact same message with an acknowledgement.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Ping** | To device | Sender's ID | Device's ID | Direct | 0x0F | 0x00 -> 0xFF (Don't Care Value) | |
| | Response | Device's ID | Sender's ID | Ack | 0x0F | Same as sent | |

**ID Request Command**

Description: Same as holding down the SET Button for 3 seconds on the device, but without going into linking mode
Example (Hex): AA BB CC DD EE FF 0F 10 01 (where DD.EE.FF is the Sender's ID, AA.BB.CC is the Devices's Id)

| SD Command | Message Direction | From Address (3 bytes) | To Address (3 bytes) | Message type | Cmd1 (1 byte) | Cmd2 (1 byte) | Notes |
|---|---|---|---|---|---|---|---|
| **ID Request** | To device | Sender's ID | Device's ID | Direct | 0x10 | 0x00 -> 0xFF (Don't Care Value) | |
| | Response | Device's ID | Sender's ID | Ack | 0x10 | Same as sent | |
| | Sent from Device | Device's ID | 0xXX (DevCat),<br>0xYY (SubCat),<br>0xZZ (firmware revision) | Broadcast | 0x02 | 0xDD (hardware revision) | Same as holding down SET Button for 3 seconds, but device not in linking mode |

| Extended Command | Message Direction | From Address (3 bytes) | To Address (3 bytes) | Message type | Cmd1 (1 byte) | Cmd2 (1 byte) | Data 1 (1 byte) | Data 2 (1 byte) |
|---|---|---|---|---|---|---|---|---|
| **Get Database** | To device | Sender's ID | Device's ID | Extended Direct | 0x2F | 0x00 | 0x00 -> 0xFF (Don't Care Value) | See Get Database Info |
| | Response | Device's ID | Sender's ID | Standard Ack | 0x2F | 0x00 | N/A | N/A |
| | From device | Device's ID | Sender's ID | Extended Direct | 0x2F | 0x00 | Same as sent | See Returned Extended Get Database Info |

| **Get Database Info** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Data 2 (1 byte)** | **Data 3** | **Data 4** | **Data 5** | **Data 6** | **Data 7** | **Data 8** | **Data 9** | **Data 10** | **Data 11** |
| 0x00 | 0x00 -> 0xFF (Hi Byte Address) | 0x00 -> 0xFF (Lo Byte Address) | 0x00 -> 0xFF (# of Records, 0x00 dumps all records | N/A | N/A | N/A | N/A | N/A | N/A |

| **Returned Extended Get Database Info (will continue to be sent until # of records is sent or until the first never been used record is sent)** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Data 2 (1 byte)** | **Data 3** | **Data 4 (1 byte)** | **Data 5** | **Data 6** | **Data 7** | **Data 8** | **Data 9** | **…** | **Data 13** |
| 0x01 | 0x00 -> 0xFF (Hi Byte Address) | 0x00 -> 0xFF (Lo Byte Address) | 0x00 | Byte 1 of record | Byte 2 of record | Byte 3 of record | Byte 4 of record | | Byte 8 of record |

| Extended Command | Message Direction | From Address (3 bytes) | To Address (3 bytes) | Message type | Cmd1 (1 byte) | Cmd2 (1 byte) | Data 1 (1 byte) | Data 2 (1 byte) |
|---|---|---|---|---|---|---|---|---|
| **Set Database** | To device | Sender's ID | Device's ID | Extended Direct | 0x2F | 0x00 | 0x00 -> 0xFF (Don't Care Value) | See Set Database Info |
| | Response | Device's ID | Sender's ID | Standard Ack | 0x2F | 0x00 | N/A | N/A |

| **Set Database Info** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Data 2 (1 byte)** | **Data 3** | **Data 4 (1 byte)** | **Data 5** | **Data 6** | **Data 7** | **Data 8** | **Data 9** | **Data 13** | **Data 14** |
| 0x02 | 0x00 -> 0xFF (Hi Byte Address) | 0x00 -> 0xFF (Lo Byte Address) | 0x01 -> 0x08 (# of bytes to write, over 0x08 is an error and ignored) | Byte 1 of data | Byte 2 of data | Byte 3 of data | Byte 4 of data | Byte 8 of data | Checksum |

| Extended Command | Message Direction | From Address<br>(3 bytes) | To Address<br>(3 bytes) | Message type | Cmd1<br>(1 byte) | Cmd2<br>(1 byte) | Data 1<br>(1 byte) | Data 2<br>(1 byte) |
|---|---|---|---|---|---|---|---|---|
| **Get iMeter Status** | To device | Sender's ID | Device's ID | Extended Direct | 0x82 | 0x00 -> 0xFF (Don't Care Value) | N/A | See Get iMeter Status Info |
| | Response | Device's ID | Sender's ID | Standard Ack | 0x82 | 0x00 -> 0xFF (Don't Care Value) | N/A | N/A |
| | From device | Device's ID | Sender's ID | Extended Direct | 0x82 | 0x00 | For Internal Testing Only | |

| Get iMeter Status Info | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Data 2<br>(1 byte)** | **Data 3-6** | **Data 7** | **Data 8** | **Data 9** | **Data 10** | **Data 11** | **Data 12** | **Data 13** | **Data 14** |
| N/A | N/A | True Power (Hi Byte) | True Power (Lo Byte) | Accumulated Energy | Accumulated Energy | Accumulated Energy | Accumulated Energy | CRC (Hi Byte) | CRC (Lo Byte) See CRC calc |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Reset Accumulated Energy** | To device | Sender's ID | Device's ID | Direct | 0x80 | 0x00 -> 0xFF (Don't Care Value) | Notes: Clears Energy, Clears Timestamp for log, Clears log index, Sets iMeter to 0-150W range<br>See Get iMeter Status to read values |
| | Response | Device's ID | Sender's ID | Ack | 0x80 | Same as Sent | |

FAQ:
1) In the example for power and energy calculation they have:

```
IntEnergy = 256*256*256*Data9 + 256*256*Data10 + 256*Data11 + Data12;
if(Data9 < 254)
   AccumEnergy = IntEnergy * 65535.0 / (1000 * 60.0 * 60.0 * 60.0);
else
   AccumEnergy = 0.0;

IntPwr = (Data7 << 8) | Data8;
*pwr = (double)IntPwr;
if(*pwr > 32767.0)
   *pwr = *pwr - 65535.0;
```

Note the bold sections.

- If Data9 is >= 254 then energy is 0. Why? Are they storing negatives and if so wouldn't it be Data9 > 254 (high bit on).
    - **INSTEON:** It was decided to no support negative energy loads like generators.
- Also if *pwr > 32767 the power is calculated as (*pwr - 65535). It seems like there is the possibility of negative power. Normally negative numbers are stored as 2's complement, but if this were so the conversion would be (*pwr - 65536). Can we get some clarification as to the actual data encoding?
    - **INSTEON:** True power calculations support both negative and positive power. Noise can cause a load-less iMeter to have a negative power and accumulate negative energy, which we do not support.
    - **INSTEON:** We are using a modified version of 2's complement (*pwr - 65535). The difference of 1 is calibrated out.

2) If the energy field can be negative does it roll over from the max positive value to the min negative value, or does it roll over to zero?

- **INSTEON:** The accumulated energy will eventually role over. When it rolls, it rolls to a negative value. The energy level is meant to be read and reset every month. It has enough bits to accumulate for about 7 months with a maximum load before rolling over.
- **INSTEON:** The energy level is reset via INSTEON command.

3) Why is the example energy decoding testing for (data9 < 254)? Basically it is setting the energy to zero if it is greater than or equal to 0xFE000000 (254 in the high byte). This seems completely arbitrary and I need to know the rules around when this condition occurs. Do I need to reset the iMeter when this happens? Do I ignore all values larger than this? How does this tie into the rollover cycle?

- **INSTEON:** The energy level should be reset if this is encountered.

## CRC Calculation

16 bit CRC calculation of payload for checking data involves data bytes from command 1 to data 12 byte

### 1.1.1 Calculation

Sample Source Code:

```
unsigned int MyFrame::crc16(unsigned char *msgbuf,int count)
{
        int loop,bit;
        unsigned int fb;
        unsigned char byte;
        unsigned int crc;

        crc = 0;

        for(loop = 0;loop < count;loop++)
        {
                byte = msgbuf[loop];
                for(bit = 0;bit < 8;bit++)
                {
                        fb = byte & 1;
                        fb = (crc & 0x8000) ? fb ^ 1 : fb;
                        fb = (crc & 0x4000) ? fb ^ 1 : fb;
                        fb = (crc & 0x1000) ? fb ^ 1 : fb;
                        fb = (crc & 0x0008) ? fb ^ 1 : fb;
                        crc = (crc << 1) | fb ;
                        byte = byte >> 1;
                }
        }
        return crc;
}
```

## Power Calculation

```
// *pwr is returned in watts
// AccumEnergy is returned in kW-h (kilowatt hours)

IntEnergy = 256*256*256*Data9 + 256*256*Data10 + 256*Data11 + Data12;
if(Data9 < 254)
      AccumEnergy = IntEnergy * 65535.0 / (1000 * 60.0 * 60.0 * 60.0); // Convert
Accumulated Energy value into kW-h

else
            AccumEnergy = 0.0;
IntPwr = (Data7 << 8) | Data8;
*pwr = (double)IntPwr;
if(*pwr > 32767.0)
*pwr = *pwr - 65535.0;
```

## 2. Checksum Information

For Set Database, Set Properties and 0x20, Data14 will contain a 2s compliment of cmd1 through 2nd to last data record in the last data record.

Example of Checksum:

| |
|---|
| 01 02 03 04 05 06 1F 2F 00 01 02 0F FF 08 E2 01 08 B6 EA 00 1B 01 11 |
| |
| From 01.02.03 to 04.05.06 |
| a record at 0FFF (A valid boundary) |
| 08 bytes a record that 04.05.06 will control |
| Group 1 the responder is 08.B6.EA  (00 1B 01 DNC) |
| 11 is the check sum |

| Int | Hex | |
|---|---|---|
| 47 | 2F | |
| 0 | 00 | |
| 1 | 01 | |
| 2 | 02 | |
| 15 | 0F | |
| 255 | FF | |
| 8 | 08 | |
| 226 | E2 | |
| 1 | 01 | |
| 8 | 08 | |
| 182 | B6 | |
| 234 | EA | |
| 0 | 00 | |
| 27 | 1B | |
| 1 | 01 | |
| 1007 | 3EF | Sum |
| | 10 | Compliment (Last byte) |
| | 11 | Add 1 |

## 3. Memory Map

### All-Link Database (AL /L) Overview

The AL /L starts at the top of external (serial) EEPROM and grows downward. In the Micro Module Shutter, top of memory is 0x0FFF. Each AL /L Record is 8 bytes long, so the first record starts at 0x0FF8, the second record starts at 0x0FF0, and so on down to 0x0300 for a total of 416 links. In what follows, the 3-byte INSTEON Address contained in a record is called the *Device ID* or sometimes just the *ID*. The high byte (MSB) of the Device ID is *ID2*, the middle byte is *ID1*, and the low byte (LSB) is *ID0*.

### Micro Module Shutter External EEPROM Structure Overview

| Location | | **Comments** |
|---|---|---|
| 0x0FF8 | 0xA2 01 AA BB CC FF FE 00 | All-Link Database Record |
| 0x0FF0 | | |
| 0x0FD8 | | |
| …… | | |
| 0x0300 | | Last Record, 416 total links allowed |
| 0x02XX | N/A | Addressing below 0x0300 is ignored by database |

### AL /L Record Format

Micro Module Shutter AL  Record Format

Database entries with Record Control Bit 6: 0 = Responder and Group 1 will control the local load.

| **Linear ALL-Link Database (AL /L) Record Format** | | |
|---|---|---|
| **Field** | **Length (bytes)** | **Description** |
| Record Control | 1 | Record Control Flag Bits:<br>Bit 7: 1 = Record is in use, 0 = Record is available<br>Bit 6: 1 = Controller (Master) of Device ID, 0 = Responder to (Slave of) Device ID<br>Bit 5: Not used<br>Bit 4: Not used<br>Bit 3: Not used<br>Bit 2: Not used<br>Bit 1: 1 = Record has been used before, 0 = ‘High-water Mark’<br>Bit 0: Not used |
| Group | 1 | ALL-Link Group Number this Device ID belongs to |
| ID | 3 | Device ID (ID2, ID1, ID0 in that order) |
| Data 1 | 1 | Not used |
| Data 2 | 1 | Not used |
| Data 3 | 1 | Not used |

To add a record to an AL /L, you search for an existing record that is marked available. (Available means the same as empty, unused or deleted.) If none is available, you create a new record at the end of the AL /L.

An unused record will have bit 7 of the *Record Control* byte set to zero. The last record in an AL /L will have bit 1 of the *Record Control* byte set to zero.

### Overwriting an Empty AL /L Record

If you found an empty record, you simply overwrite it with your new record data.

Change bit 7 of the *Record Control* byte from zero to one to show that the record is now in use.

Set bit 6 of the *Record Control* byte to one if the device containing the AL /L is an INSTEON Controller of the INSTEON Responder Device whose *ID* is in the record. If instead the device containing the AL /L is an INSTEON Responder to the INSTEON Controller Device whose *ID* is in the record, then clear bit 6 of the *Record Control* byte to zero. In other words, within an AL /L, setting bit 6 means "I'm a Controller," and clearing bit 6 means "I'm a Responder."

Put the ALL-Link Group number in the *Group* field, and put the *Device ID* in the *ID* field. Finally, set the *Data 1*, *Data 2*, and *Data 3* fields appropriately for the *Record Class* you are storing.

### Creating a New AL /L Record

To create a new record at the end of the AL /T, find the record with bit 1 of the *Record Control* byte set to zero, indicating that it is the last record in the AL /L. Flip that bit to one.

## 4. Timing Update

Stay Awake time after an activation has been changed. Older Open/Close Sensors use to stay awake for 1-2 seconds after activation. **The stay awake time after activation is now 3-4 seconds.**