# Cluster Message Interface Control Document (ICD)

# US\_SPM/SMPC\_PTB3 ICD

python implementation: https://github.com/upliftsolar/uplift-pygateway

dart implementation: upliftsolar/lookup

Abreviations:

"SN" = serial number

"ASDH" = Al Sol De Hoy ... agricultural coop in Barranquitas.

"PII". = personally identifing iinformation

"CCCV" = constant current constant voltage

Source and Destination: universal serial number identifier -- or 0x00000 if routing to a network gateway. or 0xfffff if broadcasting to everyone. On initial startup, before being assigned a identifier number, devices think their serial number is 0x000000.

Time -- u48 to avoid Y2K of 2038.

DevInfo -- TODO: define individual bits (look in code?)

For ASDH:

Mode ('m') is broadcast repeatedly by gateway to all DC conversion equipment to tell them to keep converting. If a Controller does not hear a 'mode' command every 30 seconds, it resets the bluetooth module. This is a safety feature.

For 40W, we probably need:

* "l" (location)
* "c" setpoint
* "v" version (contains both hardware and software versions... 'firmware version' wouldn't apply
* serial number (one of the ones below. Maybe 'n')
* "d" for debugging. for existing product (prototypes), this is always on. Not sure if there's any reason to make this hard to turn on. PII might be derived if debugging information has a bunch of extraneous data (specific version number)

**Colorcode for messages:**

Blue = Down (to product)

Gold = Up (from product to phone)

**Brown = Both**

0 6 12. 22 23 24

|source | dest |. time | device| controlCharacter | [variable data]

U48 U48 u48+u32 U8 U8

0 [efr] | p | va float

| vb float

| vp float

| devinfo u8

| gasgauge float

| amp float

| volt float

| temperature float

| h | uint64\_t MessageID

| uint8\_t Message[]

| H | uint64\_t

| n | u8 // set the SN

**| l | latitude float**

**| longitude float**

**| altitude float**

**| time u48**

| t | null

1 [MCU] | a | 100 byte analog (See below)

| d | string (utf8) \*\* NOTE! | D | uint8\_t 0x0: DebugOff

| uint8\_t 0x1: DebugOn

| l | serial number

**| m | mode (on = 1 / off = 0)**

| v | version string

| : | [line of firmware]

| c | SetpointCommand

| C | ConfigCommand

| F | fault bitfields + timestamp

| f | [empty] Clears any faults that may exist, so that unit will operate

Control numbers (values PROBABLY power, but amperage also fine)

|10 | 0-24% values input 1

|11 | 25-49% values input 1

|12 | 50-74%

|13 | 75-99%

|20 | 0-24% values input 2

|21 | 25-49% values input 2

|22 |

|23 |

|30 |... etc Output

|31 |

|32 |

|33 |

Control 'numbers' (sent as ints, not String or hex) will be decoded using the following code:

static List<double> toDoublesFromBinaries(List<int> long\_list\_of\_int) {

Uint8List a\_of\_int;

List<double> out = [];

double tmp;

while (long\_list\_of\_int.isNotEmpty) {

a\_of\_int = Uint8List.fromList(long\_list\_of\_int.getRange(0, 4).toList());

tmp = PowerConvert.toDoubleFromBinary(a\_of\_int);

long\_list\_of\_int.removeRange(0, 4);

out.add(tmp);

}

return out;

}

**Special 'link watchdog' behaviors:**

't' is necessary every 15 seconds

'm' absence triggers 'gateway interval shutdown' is necessary every 1 minute (doesn't matter if 'on' or 'off')

**Questions / indeterminate behavior:**

Do things fail correctly when null is sent? e.g., to 'm'

Device Info:

12 SPM (Solar Power Monitor)

8 PTB3

NOTE: For header information that will not be used, please send '0x11111111' (u256) as an 'out-of-address' indication. E.g., which 'subsystem' is irrelevant for messages directed to the gateway/app, so subsystem should be u256, arbitrarily, for better inspectability of messages

Analog 'a' reading above deconstructs like this:

struct PhaseSense {  
 float voltage;  
 float currentp;  
 float currentn;  
 float currentd;  
 float temperature;  
 };  
 struct OutSense {  
 float voltage;  
 float currentp; // p current  
 float currentn; // n current  
 float currentd; // differential  
 float temperature;  
 };  
 struct IMUSense {  
 float AX;  
 float AY;  
 float AZ;  
 };  
 struct Sense {  
 struct PhaseSense A1;//5\*4=20  
 struct PhaseSense A2;//5\*4=20  
 struct PhaseSense A3;//5\*4=20  
 struct OutSense Out;//5\*4=20  
 struct IMUSense IMU;//3\*2=6  
 float VoltageA;//4  
 float TemperatureMCU;//4  
};

Faults:

#define VA1\_overvoltage 0

#define VA1\_undervoltage 1

#define IA1\_OverCurrent 2

#define IA1\_GroundLoop 3

#define IA1\_ArcFault 4

#define TA1\_OverTemperature 5

#define VA2\_overvoltage 6

#define VA2\_undervoltage 7

#define IA2\_OverCurrent 8

#define IA2\_GroundLoop 9

#define IA2\_ArcFault 10

#define TA2\_OverTemperature 11

#define VA3\_overvoltage 12

#define VA3\_undervoltage 13

#define IA3\_OverCurrent 14

#define IA3\_GroundLoop 15

#define IA3\_ArcFault 16

#define TA3\_OverTemperature 17

#define IO\_GroundLoop 18

#define IO\_ArcFault 19

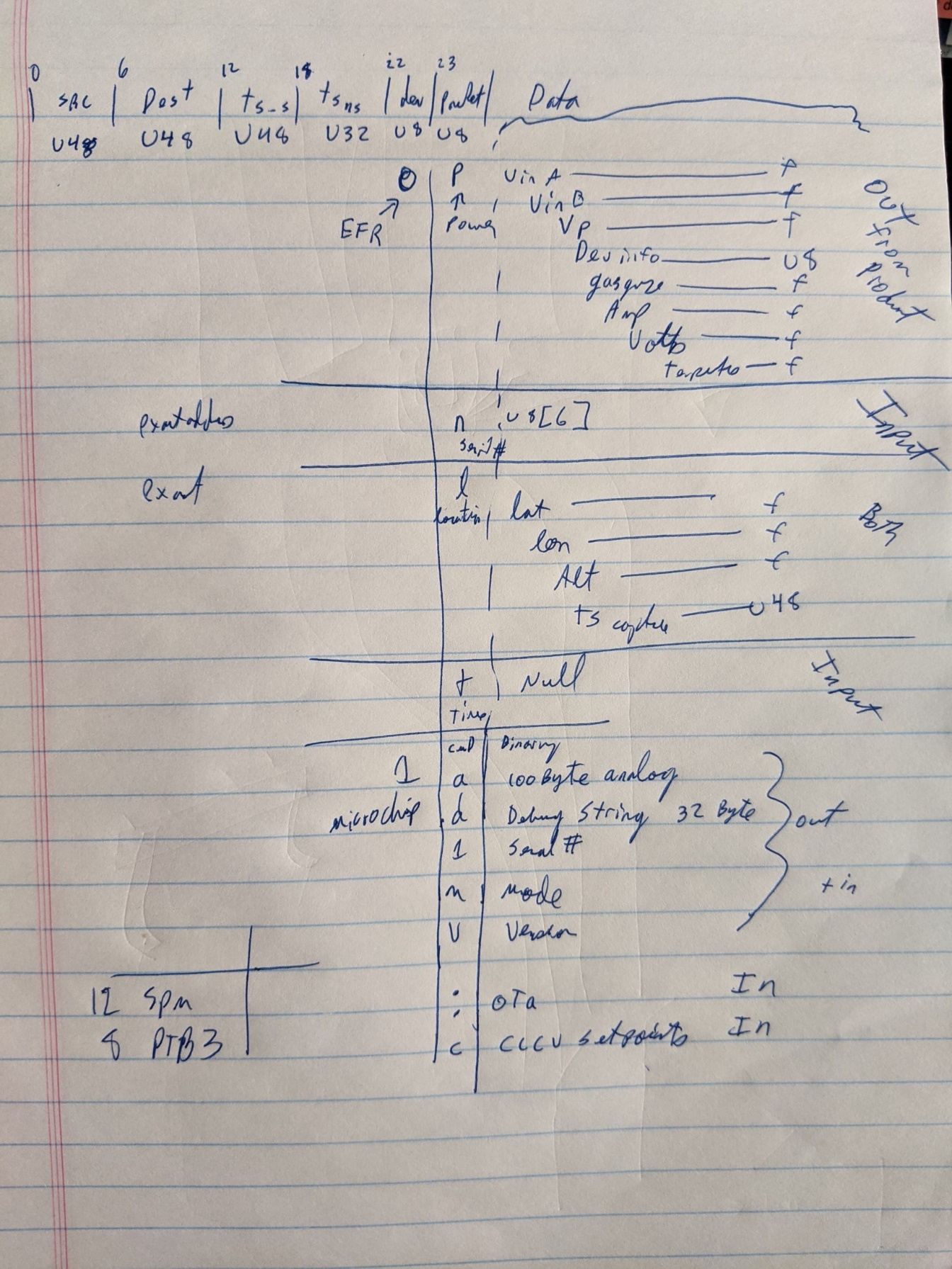
#define IO\_OverCurrent 20

#define VO\_OverVoltage 21

#define TO\_OverTemperature 22

#define TM\_OverTemperature 23

C



# US\_SMPC\_40W V0.2.x ICD

**Colorcode for messages:**

Blue = Down (to product)

Gold = Up (from product to phone)

**Brown = Both**

0 6 12. 22 23 24

|source | dest |. time | device| controlCharacter | [variable data]

U48 U48 u48+u32 U8 U8

**2 [config]|’i’ inquiry nvm read return code**

**‘c’ commit to config ram to nvm**

**returns debug of sz length of config saved and debug nvm return code**

**‘n’ pull config from nvm to ram**

**‘w’ wipe config – DO NOT USE, not stable as it does not init variables. Needs to be coupled with reset**

**‘s’ setbytes | ‘c’ config partition | uint16 offset | uint16 length | byte array of data**

**Returns packet to address 0 with the data written**

**‘r’ readbytes| ‘c’ config partition | uint16 offset | uint16 length**

**‘s’ status partition | uint16 offset | uint16 length**

**Returns packet to address 0 with the data read**

**‘r’ readbytes | -- config partition | uint16 offset | uint16 length | byte array of data**

When writing to uint64, note the LSB is first.

**3 [router]| ‘r’ reset request|**

**4 [canbus]| N/A**

**5 [ble] | loopback functionality**

**6 [thread]| N/A**

**7 [hk] | 3 fields: blob id, offset, data. Length of data is derived from the remaining length of the packet received. Blob id is uint8\_t and is currently ‘s’ and ‘c’ for status and config data. The offset of the segment of binary is uint16\_t and is the start address of the data transmitted. Currently all data is 100B or less to enabled routing through the wireless system.**

**8 [modectl]| TODO**

**9 [pwr ctl]| N/A**

**10 [time] |’t’, 116, 0x74 set time from sending packet**

**11 [dfu] |’@’ start dfu**

**‘^’ continue DFU after reset TODO does not work**

‘:’ firemware line| uint32 offset, LSB first | bytes of glb

For offset 10000, 0x10270000

‘%’ validate TODO add CRC

‘$’ finalize

0xff test control (loop back)

‘d’or 100 of 0x64 debug control char | followed by ascii string of debug message

**12 [gnss] | N/A**

**13 [sense]| TODO**

**14 [modecontroller]| ‘M’ modecontroller command**

**Data[0] = uint8\_t modestate** IE: **modecontrollerstatusstruct\_modestate\_IVCurve** from us\_modecontroller.h