Harvy Manual

Variants

	HarvyLR-36	HarvyLR-360
Shunt Value	1 Ohm (+-1%)	0,09091 Ohm (+-2%) - 1 Ohm and 0.1 Ohm in parallel
Compatible CT	W15 - W80	W150 - W500
Max. Input	36 mA	360 mA

Configuration

Parameter	Default Value	Description
ShuntVal	1000	Shunt resistance in [mOhm] When set to 100 actually 91 is used!
TxMinInterval	900	Min. LoRaWAN upload interval in [s]
LoRaLevel	3100	Min. voltage level for regular LoRaWAN uploads in [mV]
ColdStartEndLevel	2700	Min. voltage level for end of initial cold-start in [mV]
ExpFilterExponent	4	Exponential filter exponent (2^x), Default: 4 -> 16
IsecBrownout	500	Isec value in [µA] that is considered a brownout and sent via confirmable uplink.
FastSampleLevel	3500	System Volatage in [mV], when above sample every second.
FilterAbs	180	Absolute Vamp difference in [mV] to trigger uplink.
FilterRel	85	Required relative change of Isys in [%] to trigger instant upload. Value [1..100] e.g. 85 => +-15%.
LogUart	false	Set to true to print more detailed logs on UART (needs more power). Even with "false" log some is generated.

Operation

Cold Start

After reset before reaching "Vsys" of "ColdStartEndLevel" the sensor does nothing but charging. Once the "ColdStartEndLevel" is reached, it is operating even blow that voltage.

Measurement interval

The measurement interval depends on "Isec (avg-long)"

Isec (avg-long)	Measurement interval
<= 30 mA **OR** Vsys > FastSampleLevel	1 Sec
>= 20 mA	3 Sec
>= 10 mA	5 Sec
>= 5 mA	10 Sec
>= 2 mA	20 Sec
else	30 Sec

Upload interval

The upload interval is defined by "TxMinInterval" and the event filter. By default every 900 seconds (15 minutes) one uplink is executed.

Each time a value is measured it is passed to the Event Filter. When the event Filter (based on "FilterAbs" and "FilterRel") decides the value is a reasonable change, it is uploaded.

For each regular uplink the Vsys must be above "LoRaLevel". An exception is a power loss. When "Isec (current)" drops to 0 an confirmable uplink is sent and repeated until it is confirmed, regardless of "LoRaLevel". The uplink has the Power Loss flag set to 1.

Payload Description

Messgrößen

Messgröße	Beschreibung	Werteberei
Vsys (current)	Aktuelle Systemspannung bzw. Ladung des 0.47F Supercaps	<= 3600mV
Vsys (min)	Minimum der Systemspannung seit letztem LoRaWAN Uplink Paket	"
Vsys (max)	Maximum der Systemspannung seit letztem LoRaWAN Uplink Paket	"
Vamp (rms)	Aktuelle RMS Ausgangsspannung des Current Sense Amplifiers (x100 Gain)	"
Isec (current)	Aktueller RMS sekundär Strom (Errechnet aus Shunt Wert und Vamp)	0...36mA (1 Ohm) 0...360mA (0 Ohm)
Isec (avg-long)	Moving Average Filter seit Power Up über Isec (current) Berechnung (exponent=4): Filtered Value = (15 * oldValue + newValue) / 16 `newValue = (((float) ((1 << exponent) - 1)) * oldValue + newValue) / ((float) (1 << exponent));`	"
Isec (avg-short)	Mittelwert Filter mit Reset bei letztem LoRaWAN Upload (max. 600 Werte)	"
Isec (min)	Minimum des sekundär Stroms seit letztem LoRaWAN Uplink Paket	"
Isec (max)	Maximum des sekundär Stroms seit letztem LoRaWAN Uplink Paket	"

Meas. Counter	Zähler der durchgeführten Messungen seit letztem LoRaWAN Upload	x (uint32)
Last upload	Sekunden seit letztem Upload / Zeitdauer der Mittelwertbildung	x Sekunden
Temperature	nRF52832 temp sensor +-5°C Genauigkeit	

LoRaWAN Payload Codierung (Port 1)

Multibyte Daten werden **Big Endian** kodiert.

name	type	bytes	description	example	Kommentar
Version	uint8[3]	0-2	Version of the firmware running on the device	1, 5, 1 v1.5.1	Device Property
Shunt Value	uint8	3	Value of shunt resistance on hw 1/10 Ohm	10 1 Ohm | 1 0.1 Ohm	Device Property
Amplifier Gain	uint8	4	Gain, typical 100x		Device Property
Vsys (current)	uint16	5-6	Systemspannung in 1/10 mV	29470 2947mV 2.947V	
Vsys (min)	uint16	7-8	Systemspannung in 1/10 mV		
Vsys (max)	uint16	9-10	Systemspannung in 1/10 mV		
Vamp (rms)	uint16	11-12	Output Current Sense Amplifier in 1/10 mV	35991 3.5991V	
Isec (current)	uint16	13-14	Sekundär Strom CT in 1/100 mA	3101 31.01mA | 12034 120.34mA	
Isec (avg-long)	uint16	15-16	Sekundär Strom CT in 1/100 mA		
Isec (avg-short)	uint16	17-18	Sekundär Strom CT in 1/100 mA		
Isec (min)	uint16	19-20	Sekundär Strom CT in 1/100 mA		
Isec (max)	uint16	21-22	Sekundär Strom CT in 1/100 mA		
Meas. Counter	uint16	23-24	Anzahl Messungen seit letztem Uplink	0x000f 15 Messungen seit letztem Uplink	
Last upload	uint16	25-26	Zeit in Sekunden seit letztem Uplink		
Temp	int16	27-28	Temperatur in 1/10 °C		
Power Loss?	uin8	29	1= Power Loss detected 0 = no Power loss detected		Wenn Isec < 0.75mA

Example Payload:

Payload Base-64: **AAEACmRnRmYbagUCKQA3ADgAOAA3ADgAAgCEAPUB**

Payload hex: **0001000a646746661b6a0502290037003800380037003800020084 00f501**

Result:

```
"data": {
    "version": "v0.1.0",
    "powerlost": 1,
    "isecMax_mA": 0.56,
    "isecMin_mA": 0.55,
    "vampRms_mV": 55.3,
    "vsysMax_mV": 2714.1,
    "vsysMin_mV": 2613.9,
    "temperature": 24.5,
    "amplifierGain": 100,
    "isecAvgLong_mA": 0.56,
    "isecCurrent_mA": 0.55,
    "shuntValue_Ohm": 1,
    "vsysCurrent_mV": 2643.8,
    "isecAvgShort_mA": 0.56,
    "lastUploadSec_sec": 132,
    "measurmentsCounter": 2
  }
```

Parser

```
function uint16_BE(bytes, idx) {
    bytes = bytes.slice(idx || 0);
    return (bytes[0] << 8 | bytes[1] << 0);
```

```
}

function readVersion(bytes, idx) {
    bytes = bytes.slice(idx || 0);
    return "v" + bytes[0] + "." + bytes[1] + "." + bytes[2];
}

function signed(val, bits) {
    if ((val & 1 << (bits - 1)) > 0) { // value is negative (16bit 2's complement)
        var mask = Math.pow(2, bits) - 1;
        val = (~val & mask) + 1; // invert all bits & add 1 => now positive value
        val = val * -1;
    }
    return val;
}

function int16_BE(bytes, idx) {
    bytes = bytes.slice(idx || 0);
    return signed(bytes[0] << 8 | bytes[1] << 0, 16);
}


// Decoder function for TTN
function Decoder(bytes, port) {
    // Decode an uplink message from a buffer
    // (array) of bytes to an object of fields.

    var decoded = {
        "version": readVersion(bytes, 0),
        "shuntValue_Ohm": bytes[3] / 10,
        "amplifierGain": bytes[4],
        "vsysCurrent_mV": uint16_BE(bytes, 5) / 10,
        "vsysMin_mV": uint16_BE(bytes, 7) / 10,
        "vsysMax_mV": uint16_BE(bytes, 9) / 10,
        "vampRms_mV": uint16_BE(bytes, 11) / 10,
        "isecCurrent_mA": uint16_BE(bytes, 13) / 100,
        "isecAvgLong_mA": uint16_BE(bytes, 15) / 100,
        "isecAvgShort_mA": uint16_BE(bytes, 17) / 100,
        "isecMin_mA": uint16_BE(bytes, 19) / 100,
        "isecMax_mA": uint16_BE(bytes, 21) / 100,
        "measurmentsCounter": uint16_BE(bytes, 23),
        "lastUploadSec_sec": uint16_BE(bytes, 25),
        "temperature": int16_BE(bytes,27) / 10,
        "powerlost": bytes[29]
    };

    Device.setProperty("version", decoded.version);
    Device.setProperty("shuntValue", decoded.shuntValue);
    Device.setProperty("amplifierGain", decoded.amplifierGain);


    return decoded;
}

// Wrapper for Lobaro Platform
function Parse(input) {
    // Decode an incoming message to an object of fields.
    var b = bytes(atob(input.data));
    var decoded = Decoder(b, input.fPort);

    return decoded;
}

// Wrapper for Loraserver / ChirpStack
function Decode(fPort, bytes) {
    return Decoder(bytes, fPort);
}

// Wrapper for Digimondo niota.io
// module is not defined in most environments
try {
    module.exports = function (payload, meta) {
        var port = meta.lora.fport;
        var buf = Buffer.from(payload, 'hex');
        return Decoder(buf, port);
```

```
    }
} catch(e) {}
```