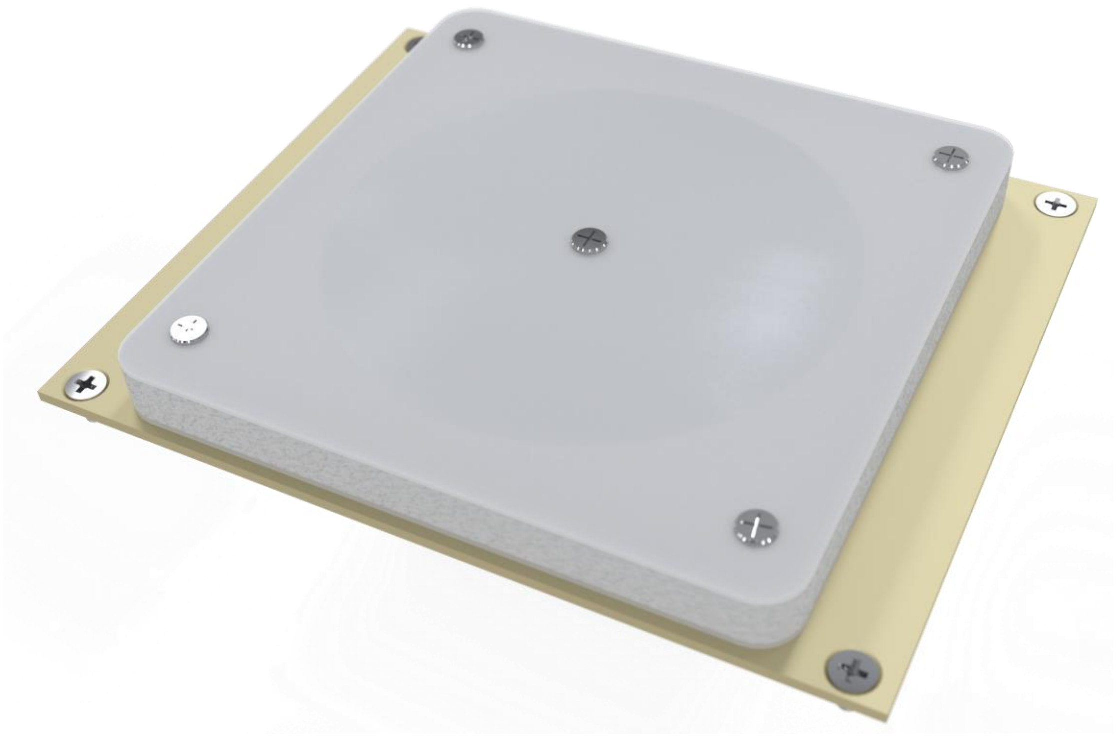


NanoCom
ANT2150 DUP
ANT2090 DUP
ANT2150 ISL

Manual

S-band active antenna

GomSpace A/S | Langagervej 6 | DK-9220 Aalborg East | Denmark
gomspace.com | sales@gomspace.com | T: +45 71 741 741

1 Table of Contents

2 Changelog

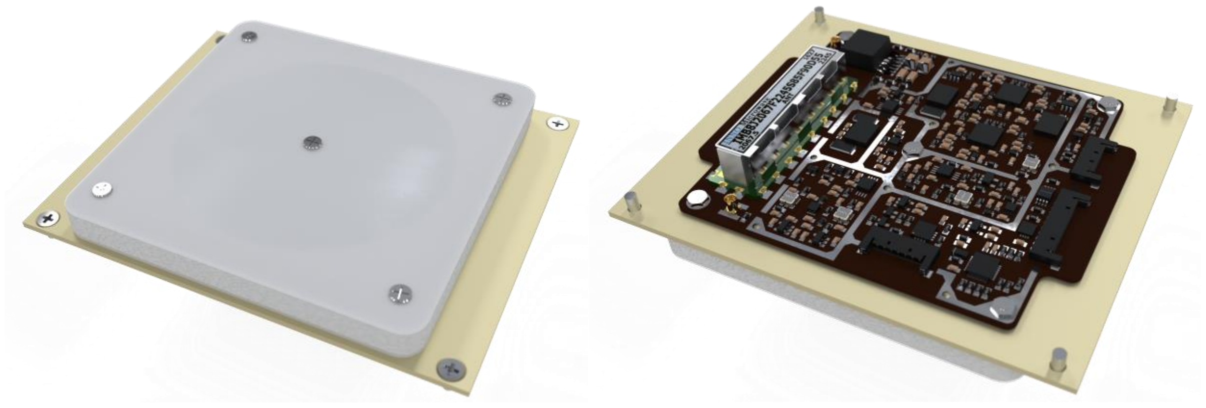
Date	Revision	Author	Description
12-10-2017	0.1	JESM/KLK	Draft
03-01-2017	1.0	JESM/PWU	First release
16-03-2018	1.0.1	JESM	Clarification of some parameters
03-05-2018	1.0.2	JESM	Parameter tables updated
28-05-2018	1.3	JESM/PWU	Added configuration and operation chapter + GOSH commands

3 Introduction

The S-band active antennas are specifically designed for interfacing with GomSpace SDR transceivers and S-band radios.

The active antenna is built as a sandwich around a shield/mounting plate. The RF signal connection between the antenna and electronics PCBs are made with RF compression connectors. This construction allows flexible mounting on several different satellite structures - just by changing the shield/mounting plate.

Below is shown a CAD of the top and bottom of the antenna:



The antenna comes in three versions:

Inter Satellite Link version
ISL-2150 Time Division Duplex in 2200-2290 MHz frequency band

Earth-Space Communication version
DUP-2150 for full duplex with RX in 2025-2110 MHz and TX in 2200-2290 MHz

Satellite Mobile Services version
DUP-2090 for full duplex with RX in 1980-2010 MHx and TX in 2170-2200 MHz

Please refer to the data sheet for technical specifications: gs-ds-nanocom-ant2000-<version>.pdf.

3.1 Unpacking and Handling Precautions

Warnings:

The antenna system employs components based on FETs and therefore requires anti-static handling precautions to be observed. Do not touch or handle the product without proper grounding.

4 Getting Started

This section contains a brief description of how to establish a connection with the antenna.

4.1.1 Connecting via J402 (debug)

By connecting to the debug interface (J402) one gain access to GOSH and all the parameter tables of the antenna – see chapter 9. GOSH is further described in the GOSH manual.

Note: *The different connectors are described in the ANT2000 datasheet.*

The following is a small example of how to use GOSH:

1) Connect to the J402 connector (UART) via serial cable
2) Your serial port communication program (minicom/tio) should be configured as follows:

   - baudrate: 500.000 baud, 8n1
   - disable flow-control

3) Once connected, one can list all commands by typing *help*:

```
ant2150-ISL # help
```

4) Use the root command *param* to list all parameter tables:

```
ant2150-ISL # param tableinfo
 id name

 0 board
 1 configuration
 2 calibration
 3 control
 4 telemetry
```

5) A specific table can be shown by running:

```
ant2150-ISL # param select configuration
  Parameter list configuration (1):

  0x0000 auto_duplex          BL  false
  0x0001 tx_pwr               U8  0
  0x0002 tx_pwr_levels        U16 1023 511 1
  0x0008 temp_warning          I16 800
  0x000A temp_dis_tx           I16 900
  0x000C temp_dis_all          I16 950
  0x000E pa_mode               U8  3
```

6) To get the value of the *pa_mode* parameter:

```
ant2150-ISL # param get pa_mode
```

7) To change the value of *pa_mode*, one must point to the correct address in memory. This is done with the *param mem* command:

```
ant2150-ISL #  param select configuration
```

8) Parameters in table ‘configuration’ can now be changed with the *param set* command:

```
ant2150-ISL #  param set pa_mode 1
```

9) Use *param list configuration* to verify, that the parameter value has changed.

10) To save the new configuration to the primary- and backup storage run:

```
ant2150-ISL #  param save configuration
ant2150-ISL #  param save configuration persistent_backup
```

The parameters are loaded from the backup storage, in case the primary storage gets corrupted.

11) To get information about storage run:

```
ant2150-ISL #  param storeinfo
name        info

persistent                  vmem: param-persistent
persistent_backup           vmem: param-persistent_backup
```

4.1.2 Connecting via CAN

By connecting a PC to the antenna via CAN, one can access the same parameter tables, that are accessible via the debug interface.

Below is given an example of how to configure a CAN adapter (PEAK PCAN-USB adapter):

1) Connect the adapter to the host PC

2) Run *lsusb* to list all connected usb devices

```
user@HostPC:~$ lsusb
.
.
.
Bus 001 Device 009: ID 17ef:100f Lenovo
Bus 001 Device 020: ID 0c72:000c PEAK System PCAN-USB
Bus 001 Device 007: ID 0bda:5411 Realtek Semiconductor Corp.
.
.
.
```

Look for your CAN device, in this case it is the PEAK System PCAN-USB.

3) Configure CAN

CAN should be configured with a speed of 1Mbps and auto restart in case of error:

```
user@HostPC:~$ sudo modprobe peak_usb
user@HostPC:~$ sudo ip link set dev can0 down
user@HostPC:~$ sudo ip link set can0 type can restart-ms 1000
user@HostPC:~$ sudo ip link set dev can0 up type can bitrate 1000000
```

***Note**: The configuration of CAN to USB adapter might vary depending on which dongle is used.*

The host PC should have csp-term installed. The GOSH manual contains a detailed description of how to configure csp-term for CAN communication.

5 Mounting

All versions are designed to be mounted on a structure using M2.5 countersunk screws.

5.1 Standard Tightening Torque for Screws

Below is shown a table for GomSpace suggested standard tightening torque for screws.

Screw Diameter [mm]	Torque [Nm]
2.5	0.65

6 Configuration

6.1 Cable Compensation and Input Level

Cable loss compensation is done at room temperature 15-35°C. It allows the SDR system to set a nominal level that will produce -10dBm at the input connector of the active antenna. All subsequent level setup must be relative to this value.

Setup is done by selecting TX mode (without *TX_EN*) and providing a carrier signal (near the center frequency) with a level of approximately -10dBm. The SDR then reads *vdeti* values and adjust level until *vdeti* equals *nom_deti_25c*.

***Note**: If modulation is applied instead of a carrier signal it is advised to average vdeti readings.*

This operation can be done once at integration, and can be repeated in operation whenever the temperature requirements are met.

In general, the input level (P_{in}) can be estimated based on *vdeti* as:

$$P_{in} = vdeti * gain\_deti\_25c + ofss\_deti\_25c$$

When this is done the output power is determined by the SDR input power (linear amplifier). Calibration values shows the nominal input level (*tx_plnom*) and frequency compensation values (*tx_fcomp*).

***Note**: GomSpace has developed a Python script, that can be used to create a plot that will show which vdeti value that will produce the nominal output power from the antenna – see Appendix 10.*

***Note**: The value of vdeti, vdeta and vdetb is a ‘raw’ measurement within a certain range. vdeta and vdetb are not calibrated detectors. They can only be used for verifying, that the two PAs have output power.*

6.2 Configuration Parameters

The configuration parameters are listed and described in Chapter 9.1.3.

The backup storage is used as a fall back, in case the primary storage gets corrupted” eller
“The parameters are loaded from the backup storage, in case the primary storage gets corrupted

***Note**: For the parameter pa_mode it should be noted, that the modes ‘A Only’ and ‘B Only’ are used mostly for debugging. In normal operation, it should be set to ‘A and B’. Switching off either A or B path lowers the output power by >6dB, in which case it is more efficient just to reduce the input power by 6 dB.*

7 Operation

This chapter describes how the antenna application operates. The antenna can be operated using the *control* parameter in the control table and/or using external control signals. When the module boots, both transmitter and receiver is turned off.

The external control signals (*TXON*, *RXON* and *TXEN*) are 3.3V CMOS logic level. They have an internal 10 kOhm pull down.

The receiver and transmitter operation are described separately.

***Note**: The control signals pins are not protected other than the protection inherent in the Atmel MCU.*

***Note**: The control parameter is a bitmask, where bit 0: RX_ON, bit 1: TX_ON and bit 2: TX_EN.*

7.1 Power On

The antenna is powered through the J400 power connector. The power connector is described in the antenna datasheet. It can be powered on by connecting a 3.3V or 5V power supply directly to the AABON pin – see Fig. 7-1.

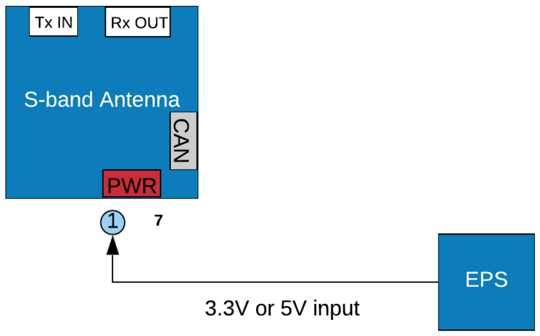


Fig. 7-1 Schematic overview of how to connect power supply

***Note**: When powering on/off the antenna via the AABON pin, it will boot with the default configuration. If the antenna is left in ‘idle’ mode custom settings will be preserved. In ‘idle’ mode, there is a small current consumption.*

***Note**: The power supply can also be controlled via the SDR. This is described in the ANT2000 Application Note. If the Application Note has not been delivered with the product, please contact GomSpace support.*

7.2 Receiver Operation

The control bit *RX_ON* is used to turn the receiver on (bitmask xx1). This can be done using the interfaces described in chapter 8.

The temperature is monitored while the receiver is operating. The receiver is shut off if the temperature gets above the threshold set in *temp_dis_all*.

7.3 Transmitter Operation

The operation of the transmitter is a 3-step approach:

1) Select transmit level
2) Turn transmitter on
3) Set transmitter active whenever the SDR wants to transmit

7.3.1 Selecting Transmit Level

To optimize the efficiency of the transmitter, the desired power level can be set by the SDR as low, medium, high, by setting the configuration parameter *tx_pwr*.

7.3.2 Setting Transmitter

The control bit *TX_ON* is used to turn the transmitter on (bitmask x1x). This can be done using the interfaces described in chapter 8.

Note: To turn on both the receiver and transmitter use 'x11'.

The temperature is monitored while the transmitter is operating. The transmitter is shut off if the temperature gets above the threshold set in configuration parameter *temp_dis_tx*.

7.3.3 Transmitter Active

There are 2 ways to activate the transmitter; auto or manual. The mode to use is based on the *auto_duplex* setting in the configuration table – see chapter 9.1.3.

<u>Manual mode</u>
In manual mode, the SDR must set the *TX_EN* bit to power on the PA. So, in order to power up the transmit chain, both the *TX_ON* and *TX_EN* bit need to be set (11x).

<u>Auto mode</u>
If *auto_duplex* is set to true, it will enable automatic antenna duplex detection. The ISL-2150 will automatically switch between RX and TX based on the input power level. DUP-2150 and DUP-2090 will automatically enable TX, but leave RX on.

7.4 Status Parameters

The status of the system can be evaluated by looking at the status parameter – see Chapter 9.1.5. The following is an example of how to read a status parameter value of 14:

[{"rx_on" } {"tx_on"} {"paon_a"} {"paon_b"} {"tx_en"} {"paen_a"} {"paen_b"} {"temp_warn"} {"temp_tx"} {"temp_all"}]

So, bit 0 is ‘RX_ON’, bit 1 is ‘TX_ON’ and so on.

Given a status of 14 (bitmask [1110]) means:

rx_on = 0 (false, is off)
tx_on = 1 (true, is on)
paon_a = 1 (true, is on)
paon_b = 1 (true, is on)

Everything else is off.

8 Interfaces

The antenna supports CAN and I^{2}C plus a debug interface. All interfaces give access to the parameters listed in chapter 9.

The antennas support two interfaces: CAN and I^{2}C. Which interface is used, is set as a hardware option when buying the product. The two interfaces are described in the following chapters.

In addition, GomSpace has developed a console-like interface called GomSpace Shell (GOSH), which provides a simple but extensive debug and configuration interface. GOSH is a general feature present on several of GomSpace's products.

It should be noted, that this manual only describes the interfaces. The actual use of the antenna in relation to other components (e.g. SR2000) is described in the accompanying application note.

Application Note

- gs-apn-nanocom-ant2000-<version>.pdf

GOSH manual

- gs-man-gosh-<version>.pdf

If these documents have not been provided, please contact GomSpace support.

8.1.1 CAN Interface

It's possible to communicate with the antenna via CAN using the CubeSat Space Protocol (CSP).

***Note**: When using the antenna with SR2000 a different CAN interface is used via the parameter 'addr_monitor'. This has been described in the ANT2000 Application Note: gs-apn-nanocom-ant2000-<version>.pdf.*

CSP is a routed network protocol that can be used to transmit data packets between individual subsystems on the satellite bus and between the satellite and ground station. For more information about CSP please read the documentation on libcsp.org and on Wikipedia: http://en.wikipedia.org/wiki/Cubesat_Space_Protocol.

The CAN interface is accessible through either:

1) GomSpace Shell (GOSH) using *rparam*
2) Remote Parameter Client API

AD.1
GOSH is running on most of GomSpace's products including csp-term. GOSH gives access to a root command called '*rparam*', which utilizes the remote Parameter Client API. The use of GOSH, *rparam* and csp-term is described in detail in the GOSH manual.

GOSH can be used for configuration and getting telemetry data.

AD.2
The remote Parameter Client API can be used for in-flight control of the antenna.

__Note__: The antenna ships with the remote Parameter Client API. It is included in the csp-client. If csp-client has not been provided, please contact GomSpace support.

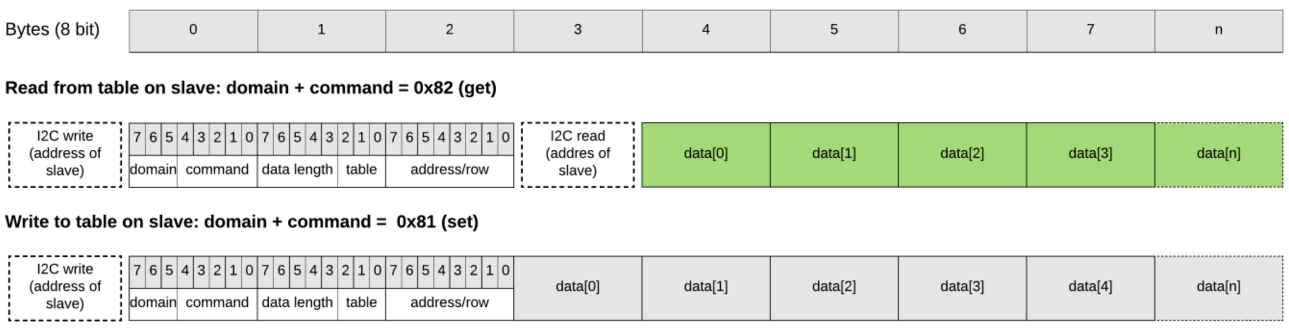
In case, the antenna is to be controlled by a third party OBC using CAN, we recommend using GomSpace's remote Parameter Client API.

Example code of how to use *rparam* can be found in *csp-client*.

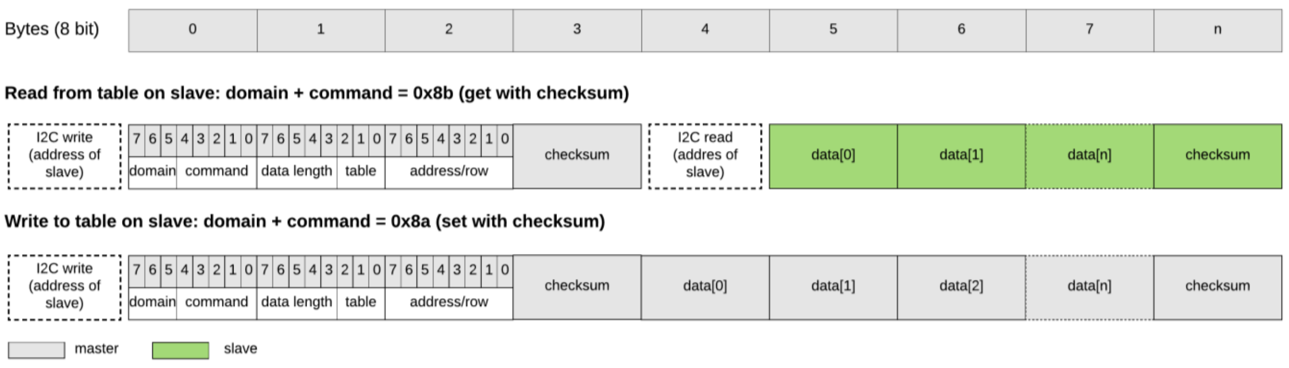
8.1.2 I^{2}C Interface

The antenna uses a generic I^{2}C interface common to other GomSpace products. The following bytes are used by the I^{2}C master and I^{2}C slave, respectively:

Without checksum



With checksum



<u>Example of how to get a parameter value</u>

The table below shows the bytes needed to get the *pa_mode* parameter value of table 1:

address of slave	domain + command	data length + table	address
09	0x82	0x11	0x000e

Example of how to set a parameter value

The table below shows the bytes needed to change the *pa_mode* parameter value to 1:

address of slave	domain + command	data length + table	address	data[0]
09	0x81	0x11	0x000e	0x0001

Since the *pa_mode* parameter is of type uint8 the data length is 1 byte – see table in chapter 9.1.3. The following tables show how the byte containing *domain + command* and the byte containing *data length + table* is split up into 5 and 3 bits:

Hex	**Binary**	**Domain** (3 bits)	**Command** (5 bits)
0x81	10000001	100 (4)	00001 (1)

Hex	**Binary**	**Data length** (5 bits)	**Table** (3 bits)
0x11	00001001	00001 (1)	001 (1)

Data is transferred in little endian.

8.1.3 GOSH

GOSH provides a text-interface to a given input/output stream such as a serial port and is used for debug and configuration.

The antenna is equipped with a debug interface (J402) that can be accessed through a USART serial cable. The debug connector is described in the datasheet. By connecting to the debug interface one gains access to GOSH.

GOSH can also be used to remotely access parameters of the antenna via *rparam* – either by connecting to the debug interface of another GomSpace product or via csp-term.

GOSH and the use of *param, rparam* and csp-term is described in the GOSH manual. If this manual has not been provided upon purchase please contact GomSpace support.

8.1.3.1 GOSH Commands

The following GOSH commands are available:

1) ant status
2) ant rx_on
3) ant tx_on
4) ant tx_en
5) ant readall
6) ant version

AD.1 (ant status)
Run *ant status* to get a status overview of all inputs and outputs of the AFE module. ADC and DAC readings are in raw values.

```
ant2150-ISL # ant status
```

AD.2 (ant rx_on)
Run *ant rx_on* to turn the receiver (Rx) on or off. Here is an example of setting *RX_ON* on (high) and off (low):

```
ant2150-ISL # ant rx_on 1
ant2150-ISL # ant rx_on 0
```

AD.3 (ant tx_on)
Run *ant tx_on* to turn the transmitter (Tx) on or off (not PA). In other words, it controls the state of the *TX_ON* output pin.

```
ant2150-ISL # ant tx_on 1
ant2150-ISL # ant tx_on 0
```

AD.4 (ant tx_en)
Run *ant tx_en* to enable the transmitter (turn PA on and off).

```
ant2150-ISL # ant tx_en 1
ant2150-ISL # ant tx_en 0
```

AD.5 (readall)
Run ant readall to read and display all values.

```
ant2150-ISL # ant readall
```

AD.6 (version)
Run *ant version* to show version.

```
ant2150-ISL # ant version
```

9 Parameters

The table below lists the parameter tables available for the antenna:

Table Number	**Name**	**Description**
0	Board	See Table 9-1: Board
1	Configuration	See Table 9-2: Configuration
2	Calibration	See Table 9-3: Calibration
3	Controller	See Table 9-4: Control
4	Telemetry	See Table 9-5: Telemetry

***Note**: Some of the parameters (auto_duplex, status and control) are further described in the context of the SR2000: gs-apn-nanocom-ant2000-<version>.pdf.*

9.1.1 Parameter Modifiers

Writing to a parameter can have different consequences, determined by how the parameter is accessed by the firmware. This is expressed using parameter modifiers found next to the type.

(A) Active parameter
The types marked with an A are active parameters; this means they are checked by the firmware during execution. In practice, this means that for example, the '*auto_duplex*' parameter is read for each time a packet is transmitted, and therefore the value will take effect on the next transmitted frame.

(B) Boot-up parameter
The types marked with an B can only be applied during system power-on. That means that after setting the parameter, nothing will happen. In order to apply the change, the table configuration must be stored and the system rebooted.

(R) Read-only parameter
The parameters marked with an R are read-only. That means that the firmware of the system will automatically update the value of the parameter, and that it can be used to readout a system state or value. Parameters marked with an R can be written to, but the value will most likely be overwritten by the firmware at a later time. An exception to this is counter values, which is only incremented by the firmware. So, in order to reset the counters, it is possible to write a zero to the counter parameters.

(P) Persistent parameter
The parameters marked with a P means that they are persistent. A persistent parameter will be stored each time they are changed in a separate memory location, allocated specifically for that single parameter. The advantage of this is, that they are not overwritten when saving and loading parameters to/from a parameter table.

9.1.2 Board Table: Id 0

Table 9-1: Board

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	uid	string (B)			Unique board identifier
0x0010	type	uint8 (B)			Board type 0: Prototype 1: ANT2150-ISL, 2: ANT2150-DUP 3: ANT2019-DUP
0x0011	rev	uint8 (B)			Board revision
0x0012	en_ext_input	bool (A)			Enable external inputs for controlling Rx-On, Tx-On and TX-Enable
0x0013	en_monitor	bool (B)			Enable Param monitor interface over CAN (uses standard CAN message id)
0x0014	brate_can	uint32 (B)	1000000	bps	CAN speed
0x0018	addr_monitor	uint8 (B)	100		Param monitor address. Min: 1 Max: 254
0x0019	en_csp_can	bool (B)			Enable CSP over CAN interface (uses extended CAN message id)
0x001a	addr_csp	uint8 (B)	10		Boards address on the CAN interface Min: 1 Max: 254
0x001b	en_i2c	bool	true		Enable I2C

9.1.3 Configuration Table: Id 1

Table 9-2: Configuration

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	auto_duplex	bool (A)	False		Automatically enable TX, when a certain TX power level is detected
0x0001	tx_pwr	uint8 (A)	0		Default TX power level. Min: 0, Max: 2
0x0002	tx_power_levels	uint16 (A)	[1023, 511, 1]		Maps tx_pwr(0..2) to a DAC value - DAC=0 max, DAC=1023 min power
0x0008	temp_warning	int16 (A)	800	0,1°C	Temperature warning level
0x000a	temp_dis_tx	int16 (A)	900	0,1°C	Temperature at which TX will be turned off
0x000c	temp_dis_all	int16 (A)	950	0,1°C	Temperature at which TX and RX will be turned off
0x000e	pa_mode	uint8 (A)	3		Power Amplifier mode, enable A and/or B. 0: None 1: A only 2: B only 3: A and B

9.1.4 Calibration Table: Id 2

Table 9-3: Calibration

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	gain_curr	float (A)			Gain factor for calculating the boards current consumption
0x0004	offs_curr	float (A)			Offset factor for calculating the boards current consumption
0x0008	nom_deti_25c	uint16			TX, nominal DETI (from telemetry table) value at 25°C. Not used by board logic
0x000c	gain_deti_25c	float		dBm	TX, DETI gain at 25°C. Not used by board logic
0x0010	ofss_deti_25c	float		dBm	TX, DETI offset at 25\u00b0C. Not used by board logic
0x0014	tx_plnom	float		dBm	TX, power nominal level. Not used by board logic
0x0020	tx_fcomp	int16		dB * 100	TX, frequency compensation. Not used by board logic
0x002e	tx_tcl	int16		dB/°C*100	TX, temperature compensation below 25°C. Not used by board logic
0x003c	tx_tch	int16		dB/°C*100	TX, temperature compensation above 25°C. Not used by board logic
0x004a	rx_fcomp	int16		dB * 100	RX, frequency compensation. Not used by board logic
0x0058	rx_nom_gain	float			RX Nominal Gain (center frequency and temperature). Not used by board logic
0x005c	rx_temp_gain	float	0.0443	dB*100/C	RX, temperature gain. Not used by board logic

9.1.5 Control Table: Id 3

Table 9-4: Control

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	status	uint32 (R)			State of the board (bits) 0 - RX ON 1 - TX ON 2 - PAON_A 3 - PAON_B 4 - TX enable 5 - PAEN_A 6 - PAEN_B 16 - temperature high, warning only 17 - temperature high, TX disabled 18 - temperature high, ALL disabled
0x0004	input	uint16 (R)			State of external input bits 0 - RX ON 1 - TX ON 2 - TX enable 3 - PIN OK
0x0006	control	uint8 (A)			Bit mask controlling the board 0 - RX_ON 1 - TX_ON 2 - TX_EN

9.1.6 Telemetry Table: Id 4

Table 9-5: Telemetry

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	uptime	uint32 (R)		seconds	How long the unit has been running, since last reset/boot
0x0004	bootcount	uint16 (R)			Number of times the unit has booted - NOT stored/persistent
0x0006	temp_ext	int16 (R)	INT16_MIN	0,1C	Temperature on board, between the PAs
0x0008	temp_int	int16 (R)	INT16_MIN	0,1C	Temperature inside MCU
0x000a	vcurr	uint16 (R)	UINT16_MAX		Voltage representing the boards current consumption as an ADC value
0x000c	i_board	int16 (R)	INT16_MAX	mA	Boards current consumption. Calculated as ((vcurr * gain_curr) + offs_curr)
0x000e	vdeti	uint16 (R)	UINT16_MAX		Raw ADC reading of the detected PA input power
0x0010	vdeta	uint16 (R)	UINT16_MAX		Raw ADC reading of the detected output power from PAA
0x0012	vdetb	uint16 (R)	UINT16_MAX		Raw ADC reading of the detected output power from PAB
0x0014	bootcause	uint16 (R)			Boot cause, extracted from the MCU

10 Appendix

10.1 Script to Generate Plots

```
#!/usr/bin/env python
from matplotlib.pylab import *

ant_cal_data = {"5426F2435DA93":
{"nom_deti": 2520, "gain_deti": 0.02494,
"offs_deti": -73.16, "tx_plnom": [-8.5, -12, -15],
"tx_fcomp": [400, 350, 200, 0, 150, 150, 450],
"tx_tcl": [4, 3, 2, 2, 2, 2, 2],
"tx_tch": [1, 1, 1, 1, 1, 3, 4]}}

ant_mapping = {"A": {"HSL": "5426F2435DA93"}}
index_to_freq = [2200, 2215, 2230, 2245, 2260, 2275, 2290]

def input_comp(fcomp, temp_c, tcl, hcl):
    if temp_c < 25.0:
        comp = fcomp + (temp_c - 25) * tcl
    else:
        comp = fcomp + (temp_c - 25) * hcl

    return comp

def txlvl_to_deti(txlvl, gain, offs):
    return txlvl / gain - (offs / gain)

def deti_to_txlvl(deti, gain, offs):
    return deti * gain + offs


ant = ant_cal_data[ant_mapping["A"]["HSL"]]
for f in range(0,7):
    comp = []
    for t in range(-25,80):
        res = input_comp(ant["tx_fcomp"][f], t, ant["tx_tcl"][f], ant["tx_tch"][f])
        comp.append(res/100.)

    plot(range(-25, 80), comp, label=str(index_to_freq[f]))

legend()

for level in [0, 1, 2]:
    figure(2 + level)
    title("vdeti for power level "+str(level)+" over frequency and temperature")
    for t in [-25, 0, 25, 50]:
        comp = []
        for f in range(0,7):
            res = input_comp(ant["tx_fcomp"][f], t, ant["tx_tcl"][f], ant["tx_tch"][f])
            comp.append(txlvl_to_deti(res/100 + ant["tx_plnom"][2-level], ant["gain_deti"], ant["offs_deti"]))
            if index_to_freq[f] == 2185 and t == 25:
                print "Vdeti@{0}MHz, {1}C, tx_pwr={2}: {3}".format(int(index_to_freq[f]), t, level, int(comp[-1]))

        plot(index_to_freq, comp, label=str(t))

    xlabel("frequency [MHz]")
    ylabel("vdeti [ADC code]")

    grid()
    legend(loc=0)
show()
```

11 Disclaimer