NanoCom
AX2150

Manual

Manual for NanoCom AX2150

Release 3.0.1

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

Document Details

Title: NanoCom AX2150 Manual
Reference: 1028903
Revision: 3.0.1
Date: 05 January 2023

Confidentiality Notice

Table of Contents

1. Introduction

The AX2150 is a combined S-band radio and CSP router designed for nanosatellite missions. This manual describes the use of the AX2150. Please refer to *[ax2150-datasheet]* for electrical and mechanical characteristics.

1.1 Unpacking and handling

Warning: Please observe precautions for handling electrostatic sensitive devices

The AX2150 is an ESD sensitive device vulnerable especially on the following interfaces: RF-Connector all CPU I/O pins. Proper precautions must be observed during the handling of the device.

Please use an ESD mat and a wrist strap as a minimum. Please wear gloves to avoid fingerprints on the anodized aluminum shield, this part is particularly difficult to rinse off. If any cleaning of the parts are required prior to flight, use only ESD safe cleaning methods and a neutral, non-reactive, IPA solvent.

Fig. 1.1: ESD handling tools

1.2 Getting started

1.2.1 RF Load

Warning: Please ensure that the RF connector is connected to a proper 50 ohm RF load capable of handling minimum 2 Watt, or a properly matched 50 ohm antenna, at all times when keying up the transmitter. Failure to do so will result in damage or degradation of the transmitter.

The first thing to connect to the AX2150 is a dummy load, or antenna. The right angle MCX connector gives a solid click when connected.

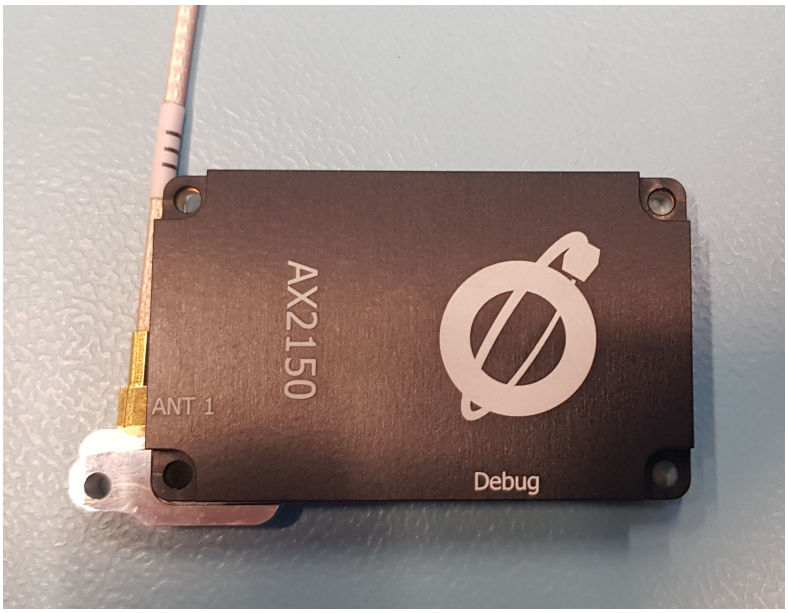


Fig. 1.2: Connecting the MCX connector

In this example a RG316 cable is used with MCX/SMA. The SMA end will be connected to a 2 W dummy load.

Note: Cable and dummy load is not included with the AX2150.

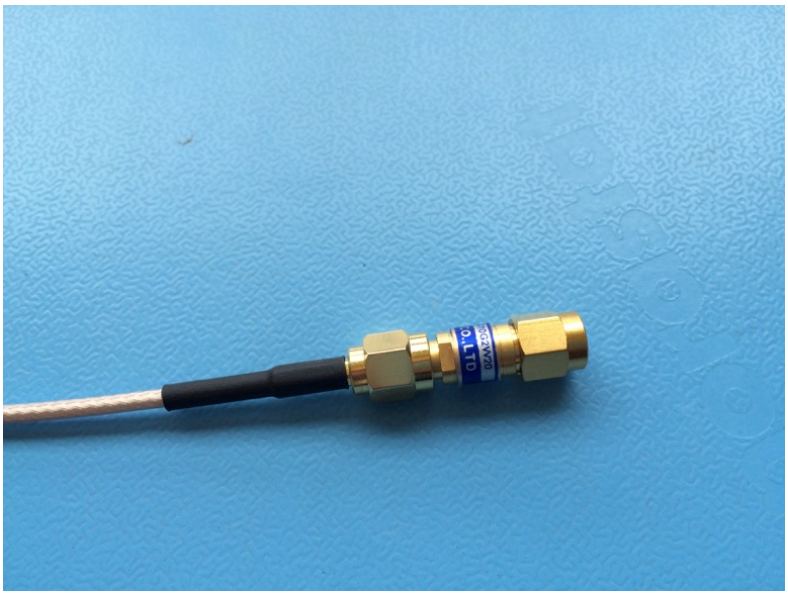


Fig. 1.3: 50 Ohm dummy load of 2 Watt

The dummy load is in this case a 20 dB attenuator. These SMA loads is not designed for continuous transmission and will get very hot if the transmitter is left operating for more than 2-3 minutes at a time. Please allow the load to cool if testing continuously, or get a bigger load with a heat sink.

1.2.2 Debug connector

The AX2150 is designed to fit on a PC/104 GomSpace motherboard with the FSI connector. Please refer to the motherboard documentation for information about mounting and powering the AX2150 module.

Fig. 1.4: AX2150 Mounted on motherboard

The easiest way to access the AX2150 is to use the accompanying 13-pin debugging connector. It has GND and UART RX/TX.

Warning: Please ensure that any PC/Laptop connected to the USB cable has a properly grounded AC-plug. The external power supply ground and PC/Laptop ground must be the same. Failure to do so will result in Common Mode noise on the GND lines of up to 100+ Volts.

The external power supply must be set to 5.0 volt and a current limiter of 1000 mA.

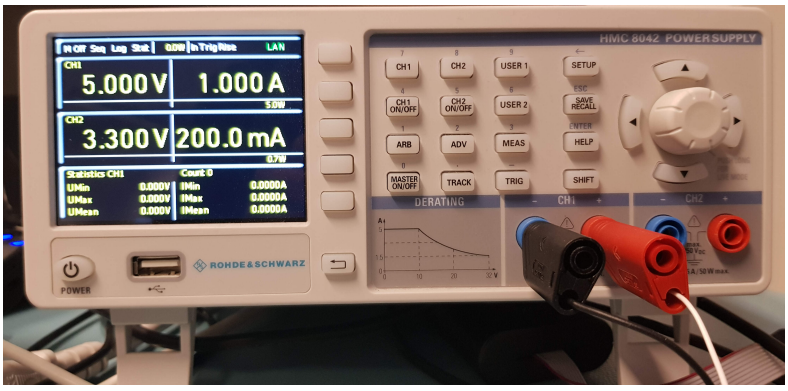


Fig. 1.5: External power supply

1.2.3 EMI Shield

The black anodized aluminum EMI shield and heat sink will come pre-installed on the AX2150 module. The module will be factory checked out both before and after mounting the shield. Pictures will be taken of the PCB before mounting the shield, so please do not try to remove the shield to check what is inside. Furthermore the screws of the shield will be tightened to factory specifications and secured with locktite™ fastener glue. Removing the shield will void the warranty on the product.

1.2.4 Access debug interface (GOSH)

GomSpace has developed a console-like interface called GomSpace Shell (GOSH), which provides a simple but extensive debug and configuration interface through UART. GOSH is a general feature present on several GomSpace products. The console provides a text-interface to a given input/output stream such as a serial port.

The console on the AX2150 is running at 500000 baud, 8n1, 3.3 V, TTL levels. For Linux users the program 'minicom' can be recommended, and for windows the program 'putty' is recommended.

In order to setup 'minicom' on a debian based distribution, or similar, please follow these steps:

- use 'apt-get install minicom'.
- open minicom as the root user and with the -s option: 'sudo minicom -s', which enters setup.
- Go to console settings and setup the baud rate to 500.000 baud, 8n1, and disable flow-control.
- Go back and select 'save as dfl' to store those settings as the default.

After installation and configuration, minicom can be opened, see example for opening command:

```
minicom -D /dev/ttyUSB0 -con
```

When connected correctly the prompt will update by the press of enter, looking like this:

```
ax2150 #
ax2150 #
ax2150 #
```

Now it is ready to receive commands. Typing "help" and pressing enter will list all available commands:

```
ax2150 # help
  bertest             ax2150: BER test
  cal_rssi            ax2150: Read RSSI
  config              ax2150: Config commands
  checkout            ax2150: Checkout commands
  pllrange            ax2150: Test PLL range
  regcmp              ax2150: Register dump
  up                  ax2150: CW key up
  dn                  ax2150: CW key dn
  ax2150              client: NanoCom AX2150
  param               Parameter System (local)
  ping                csp: Ping
  rps                 csp: Remote ps
  memfree             csp: Memory free
  buffree             csp: Buffer free
  reboot              csp: Reboot
  shutdown            csp: Shutdown
  uptime              csp: Uptime
  cmp                 csp: Management
  route               csp: Show routing table
  ifc                 csp: Show interfaces
  conn                csp: Show connection table
  raw                 csp: Send bytes to CSP node/port
  rdpopt              csp: Set RDP options
```

(continues on next page)

(continued from previous page)

```
  reset               Reset local system
  ps                  Task list
  peek                Read byte(s) from memory
  poke                Write byte to memory
  free                Show memory usage
  help                Show help
  sleep               Sleep mS
  watch               Run commands at intervals (abort on key)
  watch_check         Run commands at intervals (abort on key/failure)
  clock               Get/set system clock
  log                 Log system
  debug               Set log group mask: e|w|n|i|d|t|stand|all|off
```

Typing “param list telemetry” and pressing enter will show the telemetry parameter table:

```
ax2150 # param list 4
Table telemetry (4):
  0x0000 temp_brd        I16 297
  0x0002 temp_pa         I16 298
  0x0004 last_rssi       I16 -123
  0x0006 last_rferr      I16 39
  0x0008 tx_count        U32 0
  0x000C rx_count        U32 0
  0x0010 tx_bytes        U32 0
  0x0014 rx_bytes        U32 0
  0x0018 bootcount       U16 6
  0x001C bootcause       U32 8
  0x0020 last_contact    U32 0
  0x0024 bgnd_rssi       I16 -122
  0x0026 tx_duty         U8  0
  0x0028 tot_tx_count    U32 69
  0x002C tot_rx_count    U32 0
  0x0030 tot_tx_bytes    U32 14931
  0x0034 tot_rx_bytes    U32 0
  0x0038 cur_rf3v3_raw   I16 119
  0x003A cur_rf3v3       I16 60
  0x003C cur_5v_raw      I16 19
  0x003E cur_5v          I16 17
  0x0040 hw_det          I8  1
  0x0042 gnd_wdt_cnt     U16 0
  0x0044 gnd_wdt_left    U32 157383
```

The telemetry table includes all telemetry and is further explained in Section 8.2.5.

The console behaves like a normal UNIX shell, where entire commands (and arguments) are written as strings and executed when <enter> is pushed.

The console uses a traditional keyboard shortcut layout for navigating history, line editing and includes tab completion. The complete list of command shortcuts found in Table 1.1.

Table 1.1: console shortcut keys

Key	Description
<ctrl><a>	go to beginning of line
<ctrl><b> or <left>	go back a char
<ctrl><d>	delete char to the right of cursor
<ctrl><e>	go to end of line
<ctrl><f> or <right>	go forward a char
<ctrl><h> or <backspace>	backspace
<ctrl><k>	kill rest of line
<ctrl><l>	clear terminal
<ctrl><n> or <up>	next in history
<ctrl><p> or <dn>	prev in history
<ctrl><t>	transpose chars
<ctrl><u>	kill line from beginning
<enter>	execute command
<tab>	try and complete command

2. Configuration and Operation

The AX2150 is, as many other GomSpace products, configured and operated through the GomSpace parameter system. A description of the GOSH commands for the parameter system is found in Section 9.2, the parameters in the AX2150 is described in Section 8.

This chapter describes how to configure the AX2150 before flight, and how to operate the AX2150 in flight.

The AX2150 application handles the following:

- Routing of CSP packets between space and ground.
- Telemetry sampling
- Over temperature protection

2.1 Interfaces and protocols

This section describes the different interfaces and the protocols on top of these. For the pin out please refer to *[ax2150-datasheet]*.

The AX2150 has five physical communication interfaces:

- Debug UART
  - 500000 baud, 8n1, 3.3 V, TTL levels
  - Use for debug and configuration.
- KISS UART
  - 9600 to 500000 baud, 8n1, 3.3 V, TTL levels
  - Can be used for operation mode.
- CAN
  - ISO 11898-2 compliant
  - External termination resistors needed (120 Ohm between CAN_H and CAN_L at each end of the bus)
  - Use for configuration and operation
- I^2C
  - SDA and SCL at 3.3 V
  - Multimaster (for CSP)
  - Slave (for GSPP)
  - 7 bit addressing
  - Use for configuration and operation
- RF
  - Used for configuration and operation.

The Debug UART is the debug and configuration interface, and should only be used at such. It is advisably the first interface to make use of, refer to Section 1.2.4.

The four main interfaces are CAN, RF, KISS UART and I^2C. The protocol, Cube-Sat Space Protocol (CSP) is avaliable over these four interfaces. The communication stack is illustrated in Fig.2.1.

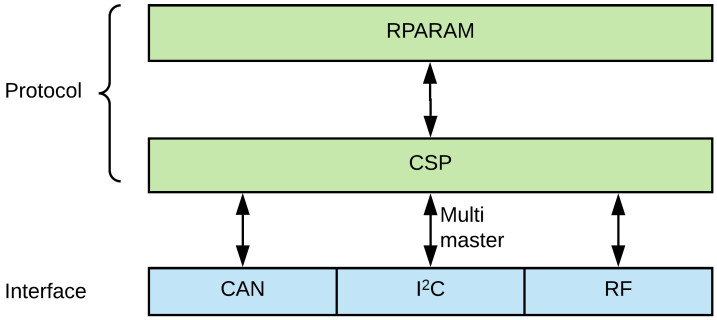


Fig. 2.1: Communication stack.

The communication interfaces are configured in the board table described in Section 8.2.1.

2.1.1 CSP and RPARAM

The protocol CSP is a small universal network layer delivery protocol similar to TCP/IP, just more suited for smaller networks without as much overhead. Most of the GomSpace subsystems utilizes this protocol. Refer to *[csp-client-man]* for further information.

On top of CSP there are one main protocols for controlling the AX2150: RPARAM.

The RPARAM protocol can be used to read, change and save parameters remotely, and is described in more detail in the *[csp-client-man]*.

For further information on implementation of RPARAM over CSP please refer to *[csp-client-man]*.

The AX2150 listens on the following port numbers on CSP:

Table 2.1: List of port numbers

Port	Name	Description
0	CSP_CMP	Control Port
1	CSP_PING	Returns a copy of the packet received
2	CSP_PS	Returns process list
3	CSP_MEMFREE	Returns memory free
4	CSP_REBOOT	Reboots subsystem
5	CSP_BUF_FREE	Returns number of free buffers
6	CSP_UPTIME	Returns subsystem uptime
7	GS_CSP_PORT_RPARAM	Controls AX2150 with parameter system (see below)
9	AX2150_PORT_GNDWDT_RESET	Resets the AX2150 ground WDT

The first ports 0-6 are default port numbers handled by the 'libcsp' network stack. They are common on all gomspace CSP systems. For a description on how to use the CSP ports, please see the libcsp repository (https://github.com/libcsp/libcsp).

The *RPARAM* port is used to service remote parameter get/set requests. For a full list of parameters of the AX2150 please see the next section of this manual. For a description on the parameter system and how to execute get/set commands, please refer to the libparam documentation.

For a description on the ground WDT please see *Ground WDT*.

2.2 Compatibility with third party ground stations

AX2150 in versions 3.0.0 and above are compatible with KSAT KSATLite ground stations in their out-of-the-box configuration. To stay within the compatible configuration, several parameters must not be changed from their default values. The default values can be found in Section 8.

Table 2.2: Parameters critical to KSAT KSATLite compatibility

Name	Table name (id)
guard	tx (5)
kup_delay	tx (5)
kup_mode	tx (5)
preamb	tx (5)
preamblen	tx (5)

Only baud rates 38400 and above are supported. As long as the parameters above are kept at their default values, KSATLite has support for the AX2150 at the Physical Layer (layer 1). Using the GomSpace NanoCom AX-Softmodem application on the mission server adds support for the data-link and network layers, for a full end-to-end system.

3. Safety functions

The AX2150 comes with several safety functions that try to take several cubesat fault scenarios into consideration.

3.1 Temperature protection

The simplest of the AX2150 safety features is the temperature protection. There is a system inside the MCU that will monitor the temperature of the power amplifier and automatically force a transmitter key-down in the event of an over temperature situation. The 'max_temp' parameter is used to set which value that will be at. If the radio goes into over temperature protection, but never comes out, for example if the environment temperature never goes below the set value, or the value was accidentally set too low, the receiver will still be running, and it should therefore be possible to set the 'max_temp' parameter to a higher value. The sample frequency on the temperature protection is 20 Hz with no hysteresis.

3.2 TX max time

The 'max_tx_time' parameter defines for how many seconds the transmitter can be keyed up, without a key-down. When the radio is transmitting a lot of data, the radio will automatically key down after this number of seconds, to listen for any uplink data, before keying up and continuing the transfer. The minimum delay between the key-down and the key-up is controlled by the TX 'guard' setting. For the default settings this is set to have a key-down period of 50 milliseconds for each 10 seconds, so the overhead of having this occasional stop and listen feature is very small. The benefit is that it should always be possible to get an uplink through to the satellite within a maximum delay of 10 seconds.

3.3 RX idle time

The RX idle timer will count receiver inactivity. If the receiver has been inactive, that means without receiving any data, for 'max_idle_time' seconds, the receiver configuration will be reinitialized. This feature is designed to prevent against receiver configuration SEU's (Single Event Upset). Initialization takes less than 100 milliseconds.

3.4 TX inhibit

The TX inhibit counter is special because it's the only configuration parameter that survives a reboot. It has been designed to count down each second until it reaches zero where it will stop. As long as the counter is not equal to zero, the transmitter will not be allowed to be keyed up. This is a safety feature used by many cubesats to ensure a certain period of radio silence after first power-up. It can also be used later to make the AX2150 be silent up for a long duration in case that the satellite is decommissioned.

Note: When the AX2150 is delivered, it will be running on the factory configuration (i.e. no boot-config or alternate-boot-config will be stored) – where the tx_inhibit value will be set to UINT32_MAX. This means that a system will not allow the user to transmit before the tx_inhibit value have been set to zero. This is a safety feature to ensure correct configuration, before transmitting.

3.5 Ground WDT

The most important safety feature of the AX2150 must be the ground WDT. This is a counter residing at a pre-set FRAM location, that independently from the parameter system will count down to zero. If it reaches zero, it will

revert to the default configuration stored in the backup stores for the parameter tables:

- board
- rx
- tx

This is a safety feature to protect against accidentally storing and setting an invalid configuration that would otherwise permanently disable the contact from the ground to the satellite.

Resetting the ground WDT has a separate CSP service on port 9 and can be cleared by sending an empty request to this port. Here is the example code for doing this:

```
csp_transaction(CSP_PRIO_HIGH, node_com, AX2150_PORT_GNDWDT_RESET, 1000,
NULL, 0, NULL, 0);
```

You can also reset the GNDWDT with the following gosh command:

```
ax2150 # ax2150 gndwdt_clear
```

The ground WDT timeout is 172800 seconds (48 hours). This means that the the ground WDT must be reset before the 48 hours elapses, otherwise the ground WDT will revert the AX2150 back to the default configuration and reboot the AX2150.

Please note that the ground WDT will be decremented by 1800 seconds everytime the AX2150 is rebooted. This is a safety feature to recover faster from an endless reboot situation caused by invalid configuration.

4. Layer 3: Network-layer (Cubesat Space Protocol)

The AX2150 works as a CSP router that is capable of receiving packets on one of its four interfaces: Radio-link, I2C-bus, CAN-bus or KISS/Serial interface, and route that message to one of the other interfaces. The basic RX/TX operation of the AX2150 radio does therefore not require any special commands in order to make the radio send and receive. Everything regarding the data-link layer, medium access control and physical setup is handled by the AX2150 automatically, but can be adjusted by the operator using the parameter system.

In order to transmit a packet from any subsystem on the satellite to a ground station node, a valid CSP packet must be generated and placed on the satellite bus. This could for example be the on-board computer (OBC) sending a beacon-message to the ground station. If the OBC has libcsp (a free open-source CSP library) built-in, all it needs is to call the function:

```
/* send out packet */
csp_sendto(CSP_PRIO_LOW, 10, 31, 0, CSP_O_NONE, beacon, 0);
```

In this example a low priority message will be generated with the destination address 10, destination port 31 and a source-port of zero (meaning you cannot reply to this message). This function will transmit the beacon over the satellite bus to the default router for the satellite. The AX2150 uses the destination field of the packet to route the message onto the radio-link interface to the ground station.

For more information on libcsp, go to http://libcsp.org

4.1 Understanding routing entries

The routing table of the AX2150 can be listed by using the ‘route’ command. The routing table is in the CIDR format. Each line in the table is read as ‘<addr>/<netmask bits> <interface> <nexthop>’

The CSP 1.0 address field is 5 bits. The netmask defines how many of these bits signify the network address, and implicitly how many signify the node address. Here are a few examples:

8/2 – Here the address is 8, and the netmask bits is 2. Translating the bits into a netmask it becomes a binary 11000b. This means that the hosts mask is 00111b which means that the host’s can range from 000b (0) to 111b (7) – So adding the host range to the network address, the routing entry 8/2 signify all routes from 8 to 8 + 7 = 15. Another way of seing this is if you have a packet destined for node 14, you take the destination address and apply the netmask 14 & 11000b = 8, so we have a match.

0/0 – This is the default route. Applying a mask of 00000b always gives the address of zero, therefor any packet destination will match this route.

The router gives priority to the routing entry with the highest netmask bits. So a /5 route takes precedence over a /2 and /0 route.

4.1.1 Example 1: Spacecraft routing table

Here is an example from an AX2150 in a satellite, with address 5.:

```
ax2150 # route
5/5 LOOP
0/0 RF
0/2 I2C
8/5 KISS
```

In this case traffic for 0-7 is routed to I2C, 8 to KISS, 5 to LOOP and every thing else to the AX2150 radio interface.

4.2 Altering the routing table (temporarily)

If you only wish to change the routing table present in RAM on a system, you can use the 'cmp route_set' command.:

```
ax2150 # cmp route_set
usage: route_set <node> <timeout> <addr> <mac> <ifstr>

ax2150 # cmp route_set 5 1000 6 255 RF
Sending route_set to node 5 timeout 1000
Dest_node: 6, next_hop_mac: 255, interface RF
Success
ax2150 # route
5/5 LOOP
0/0 RF
0/2 I2C
8/5 KISS
6/5 RF
```

This tells the router to send all traffic to node 6 over the AX2150 radio-link.

This routing table change will be reset to default after a reboot.

4.3 Altering the routing table (persistent)

In order to set the routing table, and remember it after a reboot, the table can be entered in a serialized string format using the 'csp_rtable' system parameter.

Here is an example of a routing table that is serialized:

4.3.1 Example 1: Serialized routing table

```
ax2150 # param get csp_rtable
  GET csp_rtable = "0/0 RF, 0/2 I2C, 6/5 CAN"
```

Here the table uses the same format as the route command outputs.

4.3.2 Example 2: Serialized routing table with MAC address

In some cases, the MAC address of the next hop is required. This is the case for the I2C interface. So in this example we have an OBC that needs to send its default route to the AX2150 radio. So it's default route looks a bit different:

```
ax2150 # param get csp_rtable
  GET csp_rtable = "0/0 I2C 5, 0/2 I2C"
```

This will send all traffic to 0/2 (nodes 0-7) to the I2C interface with no specified MAC address. This traffic will therefore have the same I2C address as the CSP address. However the default route is using another entry: '0/0 I2C 5' here the final 5 number is the I2C address for the next hop. So traffic using the default routing entry will have its MAC address set to address 5, and therefore be sent to the AX2150 radio from the OBC.

4.4 Interface statistics / Conn stats

Each interface has its own RX and TX counters which can be accessed with the following command:

```
ax2150 # ifc
LOOP    tx: 00002 rx: 00002 txe: 00000 rxe: 00000
                drop: 00000 autherr: 00000 frame: 00000
                txb: 38 (38.0B) rxb: 30 (30.0B)

RF      tx: 00000 rx: 00000 txe: 00000 rxe: 00000
                drop: 00007 autherr: 00000 frame: 00000
                txb: 0 (0.0B) rxb: 0 (0.0B)

I2C     tx: 00000 rx: 00000 txe: 00000 rxe: 00000
                drop: 00000 autherr: 00000 frame: 00000
                txb: 0 (0.0B) rxb: 0 (0.0B)
```

Not all counters are used by each interface, and it will be different what each interface is using their counters for.

CSP also has a list of active connection that can be displayed with the 'conn' command:

```
ax2150 # conn
[00 0x1c34] S:0, 0 -> 0, 0 -> 0, sock: 0x0
[01 0x1c58] S:1, 0 -> 0, 0 -> 0, sock: 0xf888
[02 0x1c7c] S:0, 5 -> 5, 17 -> 0, sock: 0x0
[03 0x1ca0] S:0, 5 -> 5, 0 -> 17, sock: 0x0
[04 0x1cc4] S:0, 0 -> 0, 0 -> 0, sock: 0x0
```

This output shows 4 connections in the idle state = 0, and one connection in the active = 1, state. The active connection has a socket pointer, indicating that this is a listening socket. The other closed connections still retain their last used source and destination addresses and ports so it is possible to trace the connection usage even when they are closed.

4.5 MTU

All the wired CSP interfaces (I2C, KISS and CAN) have an MTU of 255 bytes.

Interface	MTU [bytes]
I2C	255
KISS	255
CAN	255

The MTU calculation for the radio link (RF interface) is more complicated. The base MTU for a packet with no data-link layer features is 244 bytes. The MTU decreases when data-link layer features are added:

Feature	Cost (bytes)
Reed-Solomon	32
CRC-32	4
HMAC	4
Randomization	0

Below is a table of the final MTU for different data link layer options. The recommended configuration is shown in *italic* for downlink and **bold** for uplink.

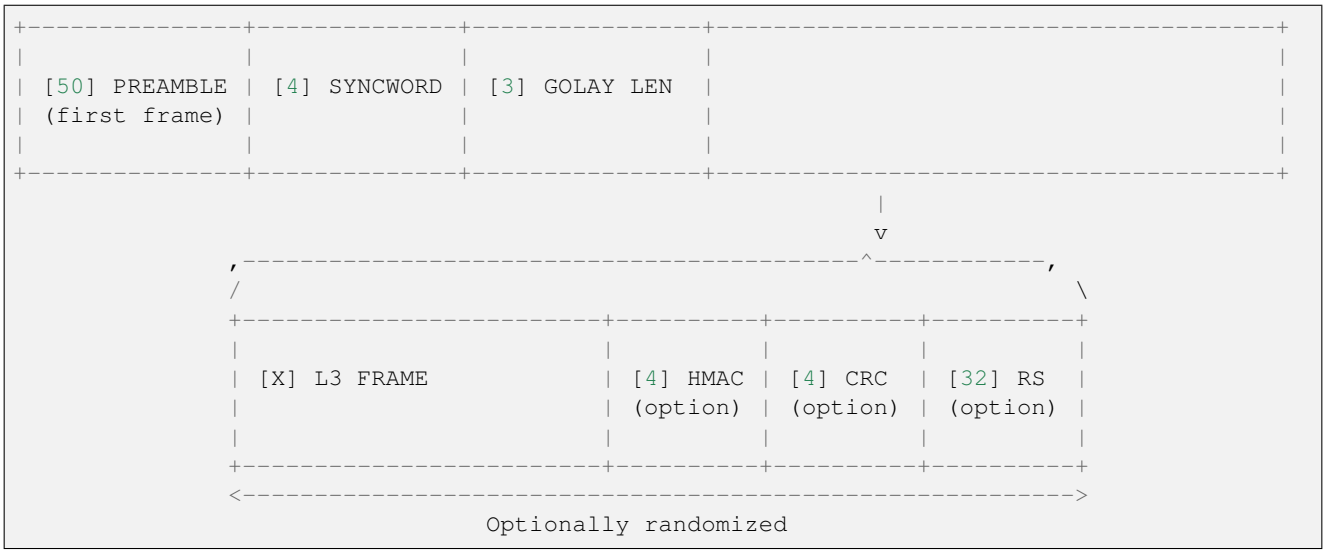
Interface	MTU [bytes]
RF (no ECC)	244
RF (CRC)	240
RF (HMAC)	240
RF (RS)	212
RF (RS+CRC)	208
RF (RS+HMAC)	208

5. Layer 2: Data-link layer

The data link layer of the AX2150 has three purposes:

1. To encapsulate the network-layer packet into a data-link-layer frame.
2. To provide frame-level error correction and control.
3. To control the medium access using a MAC protocol.

The full framing is shown here:



5.1 Data-link layer framing

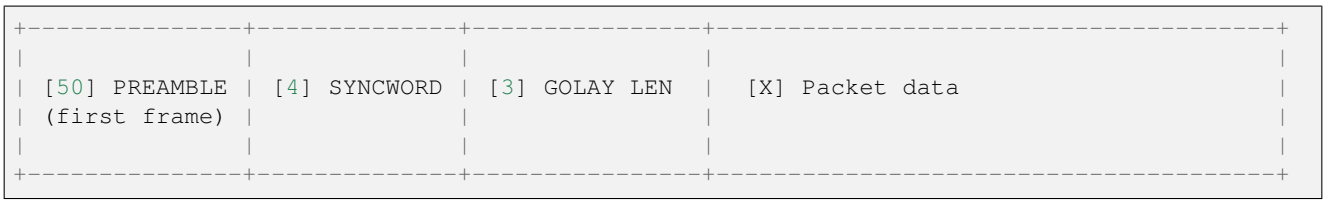
The data link frame has four parts: a preamble, a synchronization word, a length field and a data field.

The preamble is sent when transmission starts, to allow the receiever detect an incoming transmission. If multiple packets are sent in a sequence, the preamble is only added to the first packet.

The synchronization word is used to identify the start of a packet. The RF transceiver searches for a unique 32-bit ASM word in the bit-stream. It does so using a certain confidence, meaning that if more than 28 out of the 32 bits match, it is considered a match and sent to the MCU. Since searching for the sync word is performed in the RF transceiver module, the overall system power usage remains low.

The length field contains the 12-bit packet length, encoded with a G24 Golay code. This is used to correct up to three biterrors in the length field.

The 28/32-bit ASM sync, together with the robust Golay length field gives a very high certainty of a valid frame and length. In the rare case where a frame has too many bit errors in the length field, it is very likely that the packet data is also corrupt. This gives a very close to zero chance of receiving a corrupted frame.



Encoding: NRZ, No scrambling (CCSDS randomization recommended, see next section), Most significant bit first.

5.2 Error detection and correction

AX2150 Data-link layer supports the following additional modifications to the data field:

- *csp_crc*: 4 byte CRC32 checksum to the frames
- *csp_rs*: 32 byte Reed-Solomon (223, 255) block code
- *csp_rand*: CCSDS randomization of frame
- *csp_hmac*: Frame-Level HMAC

These features are applied for transmission in the following order:

```
HMAC -> CRC -> Reed Solomon FEC -> Randomization
```

It is recommended to always use randomization, Reed-Solomon and either HMAC or CRC. For uplink it is recommended to use HMAC for security. For downlink it is recommended to use CRC only, for simplicity.

5.2.1 CRC32 frame validation

When the CRC32 checksum is enabled, a 32-bit cyclic redundancy check is added to each outgoing frame. This is a simple check to detect any bit-flips in the frame.

The polynomial used is CRC-32C (Castagnoli). You can find the implementation on http://libcsp.org in `csp_crc32.c`.

5.2.2 Reed Solomon Coding

The other option for the FCS field is a 32-byte Reed-Solomon block code. This is known as a RS(223,255) coder and is capable of correcting up to 16 bytes per frame. It also provides error detection, so corrupted frames are discarded.

The RS coding increases the sensitivity of the receiver significantly. The disadvantage of RS coding is its rather large overhead, which remains constant even for smaller frames. That means that a simple 1 byte ping message would become 33 bytes with RS coding.

Generally, using the RS coding is recommended for most missions.

5.2.3 Randomization

In order to ensure low run-lengths of both ones and zeroes in the transmitted stream the data can be randomized by xor'ing with a pre-shared sequence. The pseudo-random sequence used is based on the CCSDS recommended polynomial:

$$h(x) = x8 + x7 + x5 + x3 + 1$$

5.2.4 Hash-based message authentication (HMAC)

If the HMAC is activated in the receiver only accepts packets from transmitters which have the matching `csp_hmac_key` parameter. This is useful for protecting a satellite from uplink packets sent from unauthorized ground stations.

5.3 Medium access control

The MAC is an important part of a half-duplex radio where functions like listen-before-talk and CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) are commonly used.

5.3.1 Carrier Sense (RSSI)

Whenever a frame is queued for transmission, the link is constantly monitored for any signal that another transmitter would be running. There are two indicators that are checked:

1. RSSI
2. Receiver de-framer

First of all if the RSSI level seen is above a certain threshold set by the TX 'rssibusy' parameter, the MAC layer assumes that the link is being taken by another transmitter. Secondly the receiver is asked whether or not it has detected a frame-start condition.

If any of these conditions are true, the MAC layer prevents the transmission and holds it in its outgoing buffer for later transmission.

The rssibusy level should be fine tuned after launch to optimize it to the real levels seen by the radio in space. This can be done by looking at the background RSSI over time and also look at the RSSI for the received frames. Then a good level will be below the RSSI for the received frames but also a few dB above the background RSSI to not prevent the radio from sending when the channel is free. Be sure to set rssibusy to a high value at launch to ensure that the radio will not be blocked from sending.

5.3.2 Collision avoidance

The MAC algorithm on the AX2150 takes advantage of the presumption that there are only two radios sharing the link. This gives a unique method for collision avoidance by inserting a guard period after each transmission has ended.

The TX guard is a guard period after each transmission has ended. It is very important to the synchronization between the ground station and the AX2150 when they are running at full link utilization. The idea is that whenever a transmission has just stopped, there is a guaranteed period at which there can be no collisions. This is known as Collision Avoidance.

5.3.3 Key up delay

When using the AX2150 with a ground station with a long lock-on time, a delay can be inserted as an extension of the preamble.

For example, when using the AX2150 with a receiver which specifies 1000 ms lock-on time, the 'kup_delay' parameter should be set to 1000 ms as a minimum. The key up delay is inserted in addition to the normal preamble, but does not replace it.

The key-up delay is implemented by transmitting bytes before the burst preamble. The number of bytes that are inserted is given by:

$$N = \left\lfloor \frac{T_{\text{kup}} R_{\text{TX}}}{8000} \right\rfloor$$

where

- N is the number of inserted bytes
- T_{kup} is the requested key-up delay in milliseconds
- R_{TX} is the transmission baud rate

This means that for low baud rates, the key-up delay resolution is lower than for higher baud rates. The parameter 'kup_mode' is be used to select which bytes are transmitted during the key-up delay:

- 'kup_mode' = 0: the preamble symbol (table 5, 'preamb')
- 'kup_mode' = 1: random bytes

6. Layer 1: Physical layer (RF)

The physical layer consists of a PLL, a modem and a low-pass channel filter. The configuration of each of these is described in the following sections.

6.1 Modulation

The physical layer of the AX2150 radio-link is a Gaussian shaped MSK modulation (GMSK).

6.2 Frequency

The frequency can be controlled using an on-board PLL. The AX2150 have two parameters: TX 'freq' and RX 'freq' which controls the RX and TX frequencies separately. Whenever the MAC state changes from RX to TX or vice versa, the PLL will be reconfigured.

The first time the PLL is setup at a certain frequency it will do and automatic ranging which takes a couple of milliseconds. The result of the ranging is written to the debug output during initialization. Here is an example:

```
AXSEM: PLLRANGE S 0x9 E 0x9 T 0us
```

This shows the PLL VCO range start value, 0x9 in this case, and the end value 0x9, which is similar, and finally the time used (0 microseconds).

The start value of the PLL autoranging is set to 0x9 as default, which gives an almost instantaneous PLL ranging at the desired IF frequency. When changing to other IF frequencies it might be that the VCO range result is different, and it could be an advantage to change the start value.

6.3 Bandwidth selection

The receiver bandwidth cut-off (*bw* parameter in table *rx*) should be at 1.5 times the bitrate. Setting this lower could cause attenuation of actually wanted signals, especially if there is a frequency offset between the transmitter and receiver. If the RX 'bw' parameter is set to zero, the bandwidth is automatically configured as 1.5 times the bitrate.

In case of a large frequency offset (compared to the baud rate) between the transmitter and receiver, increasing the bandwidth will increase the AFC pull-in range. However, increasing the bandwidth also decreases the receiver sensitivity, due to an increase in the noise floor. The bandwidth can be selected in steps of 1 Hz. An increase in the bandwidth by a factor two, decreases the receiver sensitivity by 3 dB.

7. Self-test Features

The AX2150 contains some advanced debugging functions available when using the GOSH debugging interface. These functions can be used for verification of the operation on the physical layer, data-link layer and network layer.

7.1 BER Test: RX Test routine

When testing the receiver, a key figure is the Bit Error Rate (BER). This can be measured by generating a modulated '01010101' sequence from a signal generator using the correct modulation parameters.

When the generator is setup, the AX2150 has a command to setup the BER test. This will disable all framing and start counting biterrors. Here is an example:

```
ax2150 # bertest begin
3120364499.004 default: BER: 0.498333, err55: 1505, errAA: 1495, count: 3000
3120364499.630 default: BER: 0.494167, err55: 3035, errAA: 2965, count: 6000
3120364500.254 default: BER: 0.499111, err55: 4508, errAA: 4492, count: 9000
3120364500.879 default: BER: 0.497667, err55: 5972, errAA: 6028, count: 12000
```

The system keeps running the bertest until the 'bertest pause' command is issued. In order to end the bertest run the 'bertest pause' command, or reset the system.

The BER is performed from an unsynchronized bit-stream by using the fact that the pattern '01010101' (0x55) can be either that, or the shifted by one version '10101010' (0xAA). It then counts the total number of bits and calculates two different error counters, one assuming correct sync on 0x55 and other based on the sync of 0xAA. It then selects the lowest of these numbers and divides that with the total number of bits to get the BER.

7.2 BER Test: TX pattern

The command `bertest txpattern` will key up the radio and send a continous '01010101' (0x55) test pattern. Remember to set the `max_tx_time` parameter to zero in order to avoid an automatic key-down of the transmitter during the tests. Also please observer the temperature of the system during any extended TX test.

7.3 Single Carrier

The commands `up` and `dn` will turn on and off an unmodulated carrier.

8. Parameter system

The parameter system on the ax2150 controls all of the functionality. It consists of five different tables, each containing a number of variables. The tables are stored as different copies in different stores.

8.1 Stores

Each table is stored in different dynamic versions in different stores with different levels of accessibility. It is shown in Table 8.1.

Table 8.1: Table store matrix

Table	FRAM (writable)	FRAM (write protected)	MCU Flash (write protected)
board	yes	yes	yes
rx	yes	yes	yes
calibration	yes	yes	yes
ax5043	no	no	no
telemetry	no	no	no
tx	yes	yes	yes

Note the actual naming used for the stores are shown in table Table 8.2.

Table 8.2: Table store naming

Medium	Name in parameter system
FRAM (w)	persistent
FRAM (wp)	protected
MCU Flash (wp)	flash

The parameter system will upon boot load each table from the leftmost available store (referring to Table 8.1). This is either the persistent or protected store. Each table is stored with a CRC value. If the CRC fails while loading, the parameter system will try to load the table from the next store. If all dynamic table versions get corrupted, then the table obtains non changeable default values, which will be the same for every AX2150.

For instance, if the writable version of the configuration table, stored in the external FRAM, gets corrupted it will fall back to the write protected version, stored in the external FRAM. If also this one fails, it falls back to the write protected version, stored in the MCU flash. If this is also corrupt, the last fall back is to the default version.

The write protected versions should be changed only on ground.

Refer to Section 9.2 for information regarding changing parameters and saving to different stores.

Warning: The values of the parameters can be different in each store. Make sure to save configuration in each available store before flight.

8.2 Parameters

The content of the four different tables are described in the following subsections.

8.2.1 Table 0: Board

The first parameter table contains all the system parameters. These parameters should be set once for your mission, and should not have to be modified during operations. Most of the parameters are boot time configuration that requires a reboot in order to take effect. It should be write protected before flight.

Table 8.3: Parameter Table 'board'

Name	Addr.	Type	
uid	0x00	string	Board id Default: `0123456789ABCD`
type	0x10	uint8	Board type Default: `1`
rev	0x11	uint8	Board revision Default: `0`
addr	0x12	uint8	CSP address Default: `5` Reboot Required: `Yes`
max_temp	0x14	int16	Maximum temperature in degrees of the PA allowed before automatic key down occurs. Default: `75` Unit: `degC`
bgndrssi_ema	0x18	float	Exponential moving average (alpha value) [0.01 to 1.00]. Default: `0.5`
i2c_en	0x1c	bool	Enables I2C. (0/1) Default: `True`
can_en	0x1d	bool	Enables CAN. Default: `True`
led_en	0x1e	uint8	Set to zero to disable the on-board leds. The LEDs should be disabled for flight in order to preserve power. Default: `1`
kiss_usart	0x1f	int8	Set which USART to use for KISS interface. Set to -1 to disable KISS interface. Refer to the datasheet for USART port numbers. Default: `-1`
gosh_usart	0x20	uint8	Set which USART to use for GOSH interface. Set to -1 to disable GOSH interface. Refer to the datasheet for USART port numbers. Default: `3`
i2c_brate	0x24	uint32	I2C bitrate in bps Default: `400000` Unit: `baud` Reboot Required: `Yes`
can_brate	0x28	uint32	CAN bitrate in bps Default: `1000000` Unit: `baud` Reboot Required: `Yes`
kiss_brate	0x2c	uint32	KISS baudrate in bps Default: `500000` Unit: `baud` Reboot Required: `Yes`
reboot_in	0x30	uint16	Number of seconds before automatic reboot will happen. This will automatically count down and reboot the AX2150 if set. Default: `0`
tx_inhibit	0x34	uint32	Number of seconds the transmitter will be shutdown. This will automatically count down and turn on the transmitter when reaching zero. *Note: this parameter is persistent* Default: `4294967295` Unit: `second` Auto Persist: `Yes`

Continued on next page

Table 8.3 – continued from previous page

Name	Addr.	Type	
log_store	0x38	uint8	Enable log-system FRAM storage backend: log hist gosh command (only use for debugging). Default: `0`
tx_pwr	0x39	uint8	TX power level Default: `2` Valid Values: **0:** 28.5 dBm (700 mW) **1:** 28 dBm (630 mW) **2:** 27 dBm (500 mW) **3:** 24 dBm (250 mW) **4:** 21.5 dBm (140 mW) **5:** 20 dBm (100 mW) **6:** 17 dBm (50 mW) **7:** 14 dBm (25 mW)
max_tx_time	0x3a	uint16	Maximum number of seconds to key up the transmitter. Default: `10`
max_idle_time	0x3c	uint16	Number of seconds the receiver can be idle, before the receiver is reinitialized. Default: `3600`
csp_rtable	0x3e	string	CSP routing table in the CIDR format: Example could be: 0/0 RF, 8/2 KISS Default: " "

8.2.2 Table 1: RX

The table holds configuration for both RX, and the mixer chip. RX parameters are all 'active' parameters which mean that they will take effect immediately after being changed. If you need to change multiple receiver parameters over the radiolink, use the rparam query system to build a packet with multiple parameters as described in *[csp-client-man]*.

Table 8.4: Parameter Table 'rx'

Name	Addr.	Type	
freq	0x00	uint32	Frequency in [Hz]. Default: `2067500000` Unit: `hertz`
if_freq	0x04	uint32	IF Frequency (change will also affects rx_freq). Default: `455000000` Unit: `hertz`
baud	0x08	uint32	Baudrate. Default: `38400` Unit: `baud`
guard	0x0c	uint16	RX guard in [ms] (guard period after receiving a frame). Default: `0` Unit: `millisecond`
pllrang	0x0e	uint8	Startup value of the PLLRANGE register. Default: `9`
csp_hmac	0x0f	bool	Enable HMAC (checksum and authentication). Default: `False`
csp_rs	0x10	bool	Enable Reed-Solomon. Default: `True`
csp_crc	0x11	bool	Enable CRC-32. Default: `True`
csp_rand	0x12	bool	Enable CCSDS randomization. Default: `True`

Continued on next page

Table 8.4 – continued from previous page

Name	Addr.	Type	
csp_hmac_key	0x13	data	HMAC key (needs to match transmitter). Default: `00000000000000000000000000000000`
mix_curset	0x24	uint16	RX Mixer current setting. Default: `2`
bw	0x28	uint32	Receiver bandwidth in Hz, must be at least 1.5 times bigger than the baudrate but can be set to zero in order to let the AX2150 auto calculate the best rx_bw for the desired bitrate. Default: `0` Unit: `hertz`
afcrange	0x2c	int32	Sets the AFC pull-in range in Hz. Set to zero to disable AFC. Set to a negative value to autocalculate based on the formula: afcrange = rx_bw/(-1 * rx_afcrange) – The automatic calculation is capped at 10 kHz. Default: `-4` Unit: `hertz`

8.2.3 Table 2: Calibration

The table holds board dependent calibration values set in production. This table gets write protected in production.

Table 8.5: Parameter Table 'calibration'

Name	Addr.	Type	
rssi_offset	0x00	int8	Sets the RSSI indicator offset, if you have external gain please adjust here for correct RSSI readings. Default: `-10`
tcxo_offset	0x02	int16	TCXO offset. Default: `20` Unit: `hertz`
tx_trx_output	0x04	uint16[8]	TX trx output indexed by power level Default: `[1500, 1500, 1500, 1500, 1500, 800, 500, 300]`
dac_value	0x14	uint16[8]	TX dac value indexed by power level Default: `[4095, 3700, 3150, 2500, 2075, 1900, 1900, 1900]`

8.2.4 Table 3: AX5043

The AX5043 table is a view of the registers of the AX5043 chip. The table can be used in debugging scenarios to read a and write the contents of the registers.

Table 8.6: Parameter Table 'ax5043'

Name	Addr.	Type	
revision	0x00	uint8	
scratch	0x01	uint8	
pwrmode	0x02	uint8	
powstat	0x03	uint8	
powststat	0x04	uint8	
powirqmask	0x05	uint8	
irqmask	0x06	uint16	
rdevenmask	0x08	uint16	
irqinversi	0x0a	uint16	
irqrequest	0x0c	uint16	
rdevenrequ	0x0e	uint16	
modulation	0x10	uint8	

Continued on next page

Table 8.6 – continued from previous page

Name	Addr.	Type	
encoding	0x11	uint8	
framing	0x12	uint8	
crcinit	0x14	uint32	
fec	0x18	uint8	
fecsync	0x19	uint8	
fecstatus	0x1a	uint8	
radiostate	0x1b	uint8	
xtalstatus	0x1c	uint8	
pinstate	0x1d	uint8	
pinfunsysc	0x1e	uint8	
pinfundclk	0x1f	uint8	
pinfundata	0x20	uint8	
pinfunirq	0x21	uint8	
pinfunants	0x22	uint8	
pinfunpwra	0x23	uint8	
pwramp	0x24	uint8	
fifostat	0x25	uint8	
fifodata	0x26	uint8	
fifocount	0x28	uint16	
fifofree	0x2a	uint16	
fifothresh	0x2c	uint16	
pllloop	0x2e	uint8	
pllcpi	0x2f	uint8	
pllvcodiv	0x30	uint8	
pllrnga	0x31	uint8	
freqa	0x34	uint32	
pllloopbst	0x38	uint8	
pllcpibst	0x39	uint8	
pllrngb	0x3a	uint8	
freqb	0x3c	uint32	
rssi	0x40	int8	Signal strength
bgndrssi	0x41	int8	Signal strength
diversity	0x42	uint8	Signal strength
agccounter	0x43	uint8	Signal strength
trkdatart2	0x44	uint8	Receiver tracking
trkdatart1	0x45	uint8	Receiver tracking
trkdatart0	0x46	uint8	Receiver tracking
trkampl	0x48	uint16	Receiver tracking
trkphase	0x4a	uint16	Receiver tracking
trkrffreq2	0x4c	uint8	Receiver tracking
trkrffreq1	0x4d	uint8	Receiver tracking
trkrffreq0	0x4e	uint8	Receiver tracking
trkfreq	0x50	uint16	Receiver tracking
trkfskdem	0x52	uint16	Receiver tracking
trkafskdem	0x54	uint16	Receiver tracking
timer2	0x56	uint8	Timer
timer1	0x57	uint8	Timer
timer0	0x58	uint8	Timer
wkuptimer	0x5a	uint16	Wakeup Timer
wkup	0x5c	uint16	Wakeup Timer
wkupfrq	0x5e	uint16	Wakeup Timer
wkupxoearl	0x60	uint8	Wakeup Timer
iffreq	0x62	uint16	Receiver Parameters

Continued on next page

Table 8.6 – continued from previous page

Name	Addr.	Type	
decimation	0x64	uint8	Receiver Parameters
rxdatrate2	0x65	uint8	Receiver Parameters
rxdatrate1	0x66	uint8	Receiver Parameters
rxdatrate0	0x67	uint8	Receiver Parameters
maxdroffs2	0x68	uint8	Receiver Parameters
maxdroffs1	0x69	uint8	Receiver Parameters
maxdroffs0	0x6a	uint8	Receiver Parameters
maxrfoffs2	0x6b	uint8	Receiver Parameters
maxrfoffs1	0x6c	uint8	Receiver Parameters
maxrfoffs0	0x6d	uint8	Receiver Parameters
fskdmax	0x6e	uint16	Receiver Parameters
fskdmin	0x70	uint16	Receiver Parameters
afskspace	0x72	uint16	Receiver Parameters
afskmark	0x74	uint16	Receiver Parameters
afskctrl	0x76	uint8	Receiver Parameters
amplfilter	0x77	uint8	Receiver Parameters
freqleak	0x78	uint8	Receiver Parameters
rxparamsets	0x79	uint8	Receiver Parameters
rxcurparamcur	0x7a	uint8	Receiver Parameters
agcgain0	0x7b	uint8	Receiver Parameter Set 0
agctarget0	0x7c	uint8	Receiver Parameter Set 0
agcahyst0	0x7d	uint8	Receiver Parameter Set 0
agcminmax0	0x7e	uint8	Receiver Parameter Set 0
timegain0	0x7f	uint8	Receiver Parameter Set 0
drgain0	0x80	uint8	Receiver Parameter Set 0
phasegain0	0x81	uint8	Receiver Parameter Set 0
freqgaina0	0x82	uint8	Receiver Parameter Set 0
freqgainb0	0x83	uint8	Receiver Parameter Set 0
freqgainc0	0x84	uint8	Receiver Parameter Set 0
freqgaind0	0x85	uint8	Receiver Parameter Set 0
amplgain0	0x86	uint8	Receiver Parameter Set 0
freqdev0	0x88	uint16	Receiver Parameter Set 0
fourfsk0	0x8a	uint8	Receiver Parameter Set 0
bboffsres0	0x8b	uint8	Receiver Parameter Set 0
agcgain1	0x8c	uint8	Receiver Parameter Set 1
agctarget1	0x8d	uint8	Receiver Parameter Set 1
agcahyst1	0x8e	uint8	Receiver Parameter Set 1
agcminmax1	0x8f	uint8	Receiver Parameter Set 1
timegain1	0x90	uint8	Receiver Parameter Set 1
drgain1	0x91	uint8	Receiver Parameter Set 1
phasegain1	0x92	uint8	Receiver Parameter Set 1
freqgaina1	0x93	uint8	Receiver Parameter Set 1
freqgainb1	0x94	uint8	Receiver Parameter Set 1
freqgainc1	0x95	uint8	Receiver Parameter Set 1
freqgaind1	0x96	uint8	Receiver Parameter Set 1
amplgain1	0x97	uint8	Receiver Parameter Set 1
freqdev1	0x98	uint16	Receiver Parameter Set 1
fourfsk1	0x9a	uint8	Receiver Parameter Set 1
bboffsres1	0x9b	uint8	Receiver Parameter Set 1
agcgain2	0x9c	uint8	Receiver Parameter Set 2
agctarget2	0x9d	uint8	Receiver Parameter Set 2
agcahyst2	0x9e	uint8	Receiver Parameter Set 2
agcminmax2	0x9f	uint8	Receiver Parameter Set 2

Continued on next page

Table 8.6 – continued from previous page

Name	Addr.	Type	
timegain2	0xa0	uint8	Receiver Parameter Set 2
drgain2	0xa1	uint8	Receiver Parameter Set 2
phasegain2	0xa2	uint8	Receiver Parameter Set 2
freqgaina2	0xa3	uint8	Receiver Parameter Set 2
freqgainb2	0xa4	uint8	Receiver Parameter Set 2
freqgainc2	0xa5	uint8	Receiver Parameter Set 2
freqgaind2	0xa6	uint8	Receiver Parameter Set 2
amplgain2	0xa7	uint8	Receiver Parameter Set 2
freqdev2	0xa8	uint16	Receiver Parameter Set 2
fourfsk2	0xaa	uint8	Receiver Parameter Set 2
bboffsres2	0xab	uint8	Receiver Parameter Set 2
agcgain3	0xac	uint8	Receiver Parameter Set 3
agctarget3	0xad	uint8	Receiver Parameter Set 3
agcahyst3	0xae	uint8	Receiver Parameter Set 3
agcminmax3	0xaf	uint8	Receiver Parameter Set 3
timegain3	0xb0	uint8	Receiver Parameter Set 3
drgain3	0xb1	uint8	Receiver Parameter Set 3
phasegain3	0xb2	uint8	Receiver Parameter Set 3
freqgaina3	0xb3	uint8	Receiver Parameter Set 3
freqgainb3	0xb4	uint8	Receiver Parameter Set 3
freqgainc3	0xb5	uint8	Receiver Parameter Set 3
freqgaind3	0xb6	uint8	Receiver Parameter Set 3
amplgain3	0xb7	uint8	Receiver Parameter Set 3
freqdev3	0xb8	uint16	Receiver Parameter Set 3
fourfsk3	0xba	uint8	Receiver Parameter Set 3
bboffsres3	0xbb	uint8	Receiver Parameter Set 3
modcfgf	0xbc	uint8	Transmitter parameters
fskdev2	0xbd	uint8	Transmitter parameters
fskdev1	0xbe	uint8	Transmitter parameters
fskdev0	0xbf	uint8	Transmitter parameters
modcfga	0xc0	uint8	Transmitter parameters
txrate2	0xc1	uint8	Transmitter parameters
txrate1	0xc2	uint8	Transmitter parameters
txrate0	0xc3	uint8	Transmitter parameters
txpwrcofa	0xc4	uint16	Transmitter parameters
txpwrcofb	0xc6	uint16	Transmitter parameters
txpwrcofc	0xc8	uint16	Transmitter parameters
txpwrcofd	0xca	uint16	Transmitter parameters
txpwrcofe	0xcc	uint16	Transmitter parameters
pllvcoi	0xce	uint8	PLL parameters
pllvcoir	0xcf	uint8	PLL parameters
pllockdet	0xd0	uint8	PLL parameters
pllrngclk	0xd1	uint8	PLL parameters
xtalcap	0xd2	uint8	Crystal oscillator
bbtune	0xd3	uint8	Baseband
bboffscap	0xd4	uint8	Baseband
pktaddrcfg	0xd5	uint8	Packet format
pktlencfg	0xd6	uint8	Packet format

Continued on next page

Table 8.6 – continued from previous page

Name	Addr.	Type	
pktlenoffst	0xd7	uint8	Packet format
pktmaxlen	0xd8	uint8	Packet format
pktaddr	0xdc	uint32	Packet format
pktaddrmsk	0xe0	uint32	Packet format
match0pat	0xe4	uint32	Pattern match
match0len	0xe8	uint8	Pattern match
match0min	0xe9	uint8	Pattern match
match0max	0xea	uint8	Pattern match
match1pat	0xec	uint16	Pattern match
match1len	0xee	uint8	Pattern match
match1min	0xef	uint8	Pattern match
match1max	0xf0	uint8	Pattern match
tmgtxboost	0xf1	uint8	Packet controller
tmgtxsettle	0xf2	uint8	Packet controller
tmgrxboost	0xf3	uint8	Packet controller
tmgrxsettle	0xf4	uint8	Packet controller
tmgrxoffsacq	0xf5	uint8	Packet controller
tmgrxcoaragc	0xf6	uint8	Packet controller
tmgrxagc	0xf7	uint8	Packet controller
tmgrxrssi	0xf8	uint8	Packet controller
tmgrxpreamb1	0xf9	uint8	Packet controller
tmgrxpreamb2	0xfa	uint8	Packet controller
tmgrxpreamb3	0xfb	uint8	Packet controller
rssireference	0xfc	uint8	Packet controller
rssiabsthr	0xfd	uint8	Packet controller
bgndrssigain	0xfe	uint8	Packet controller
bgndrssithr	0xff	uint8	Packet controller
pktchunksize	0x100	uint8	Packet controller
pktmiscflags	0x101	uint8	Packet controller
pktstoreflag	0x102	uint8	Packet controller
pktaccptflag	0x103	uint8	Packet controller
dacvalue	0x104	uint16	DAC
dacconfig	0x106	uint8	DAC
f1c	0x107	uint8	Performance tuning parameters

8.2.5 Table 4: Telemetry

The radio contains a separate memory area allocated for telemetry data such as the current temperature, data counters and the bootcounter. Telemetry is read only and updated every second for most parameters, others are updated on certain events. The memory map for the telemetry is as follows:

Table 8.7: Parameter Table 'telemetry'

Name	Addr.	Type	
temp_brd	0x00	int16	Board temperature. Default: `0` Unit: `decidegC`
temp_pa	0x02	int16	PA temperature. Default: `0` Unit: `decidegC`
last_rssi	0x04	int16	Last RSSI. Default: `0` Unit: `dBm`
last_rferr	0x06	int16	Last rferr. Default: `0` Unit: `hertz`
tx_count	0x08	uint32	TX count. Default: `0`
rx_count	0x0c	uint32	TX count. Default: `0`
tx_bytes	0x10	uint32	TX bytes. Default: `0` Unit: `byte`
rx_bytes	0x14	uint32	TX bytes. Default: `0` Unit: `byte`
bootcount	0x18	uint16	Boot counter. Default: `0` Auto Persist: `Yes`
bootcause	0x1c	uint32	Boot cause. Default: `0` Valid Values: **0:** Unknown **1:** Brownout **2:** Power on **3:** Watchdog **4:** Software **5:** External reset pin **6:** Wake from sleep **7:** CPU error (exception) **8:** JTAG
last_contact	0x20	uint32	Time stamp of Last contact. Default: `0` Auto Persist: `Yes`
pll_lock_tx	0x24	bool	PLL lock at last TX switch. Default: `False`
pll_lock_rx	0x25	bool	PLL lock at last RX switch. Default: `False`
bgnd_rssi	0x26	int16	bgnd rsssi. Default: `0` Unit: `dBm`
tx_duty	0x28	uint8	TX duty. Default: `0`
tot_tx_count	0x2c	uint32	Total TX count. Default: `0` Auto Persist: `Yes`
tot_rx_count	0x30	uint32	Total TX count. Default: `0` Auto Persist: `Yes`

Continued on next page

Table 8.7 – continued from previous page

Name	Addr.	Type	
tot_tx_bytes	0x34	uint32	Total TX bytes. Default: `0` Unit: `byte` Auto Persist: `Yes`
tot_rx_bytes	0x38	uint32	Total TX bytes. Default: `0` Unit: `byte` Auto Persist: `Yes`
cur_rf3v3_raw	0x3c	int16	Current 3v3 supply Raw ADC reading. Default: `0`
cur_rf3v3	0x3e	int16	Current 3v3 supply. Default: `0` Unit: `milliampere`
cur_5v_raw	0x40	int16	Current PA-LNA 5V Raw ADC reading. Default: `0`
cur_5v	0x42	int16	Current PA-LNA 5V. Default: `0` Unit: `milliampere`
hw_det	0x44	int8	Hardware detect. Default: `0`
rx_mode	0x45	uint8	RX mode (0: uninitialized, 1: RAW mode, 5: ready to receive) Default: `0`
gnd_wdt_cnt	0x46	uint16	Ground watchdog reboots Auto Persist: `Yes`
gnd_wdt_left	0x48	uint32	Ground watchdog value (remaining seconds before reboot) Unit: `second`

8.2.6 Table 5: TX

The table holds configuration for both TX, and the mixer chip. TX parameters are all 'active' parameters which mean that they will take effect immediately after being changed. If you need to change multiple receiver parameters over the radiolink, use the rparam query system to build a packet with multiple parameters.

Table 8.8: Parameter Table 'tx'

Name	Addr.	Type	
freq	0x00	uint32	Frequency (change will also affects tx_lo_freq). Default: `2245000000` Unit: `hertz`
if_freq	0x04	uint32	IF Frequency (change will also affects tx_freq). Default: `465000000` Unit: `hertz`
baud	0x08	uint32	Baudrate. Default: `38400` Unit: `baud`
guard	0x0c	uint16	RX guard in [ms] (guard period after receiving a frame). Default: `400` Unit: `millisecond`
pllrang	0x0e	uint8	Startup value of the PLLRANGE register. Default: `9`
csp_hmac	0x0f	bool	Enable HMAC (checksum and authentication). Default: `False`
csp_rs	0x10	bool	Enable Reed-Solomon. Default: `True`
csp_crc	0x11	bool	Enable CRC-32. Default: `True`

Continued on next page

Table 8.8 – continued from previous page

Name	Addr.	Type	
csp_rand	0x12	bool	Enable CCSDS randomization. Default: `True`
csp_hmac_key	0x13	data	HMAC key (needs to match transmitter). Default: `00000000000000000000000000000000`
mix_curset	0x24	uint16	TX Mixer current setting. Default: `4`
preamb	0x26	uint8	The byte to use as preamble. Default: `170`
preamblen	0x27	uint8	The length of the preamble in bytes. Default: `50`
preambflags	0x28	uint8	The flags to use for the preamble (do not modify). Default: `56`
intfrm	0x29	uint8	The byte to use between two frames. Default: `126`
intfrmlen	0x2a	uint8	The number of bytes to put between two frames. Default: `0`
intfrmflags	0x2b	uint8	The flags to use for the intfrm bytes (do not modify). Default: `56`
rssibusy	0x2c	int16	The transmitter will consider the channel busy when the RSSI is above this value. Default: `-95` Unit: `dBm`
kup_delay	0x2e	uint16	An additional delay of the first frame after the radio have been keyed up. Usefull for slow RX/TX relays. Default: `1000` Unit: `millisecond`
kup_mode	0x30	uint8	Type of key-up filler Default: `0` Valid Values: **0:** mode preamble symbol **1:** mode random
ber	0x34	float	Injects random bit-errors in transmitted frames with this probability. Default: `0.0`

Warning: If KSATLite compatibility is required the following parameters must remain at their default values. Please see Section 2.2 for more details.

- guard
- preamb
- preamblen
- kup_delay
- kup_mode

9. Commands

The AX2150 features the console-like interface GOSH, as described in Section 1.2.4. This chapter describes the AX2150 specific GOSH commands and the GOSH commands needed to operate the parameter system.

9.1 AX2150 commands

Every normal operation and configuration of the AX2150 is done through the parameter system. There is therefore only one command group, which is specific to the AX2150.This group is described in this section.

Table 9.1: AX2150 commands

Command	Description
bertest begin	Start biterror test
bertest pause	Pause biterror test
bertest restart	Restart biterror test
bertest tx_pattern <pattern>	Set the transmit pattern
cal_rssi	Read the RSSI level
config update_default <table_id>	Write parameter tables to all stores
config gnd_wdt [timeout]	Get or set the gnd_wdt reset value
up	Starts to transmit a CW signal
dn	Stop transmit
ax2150 node	Set CSP address of the ax2150 node
ax2150 hk	retrieve telemetry
ax2150 gndwdt_clear	Clear ground watchdog

9.2 Parameter system commands

Entering “param” shows the following sub commands:

```
ax2150 # param
Local Parameter System
  select              Select working table
  list                List all parameters
  tableinfo           Show table information
  export              Export parameters to stdout
  set                 Set parameter value
  get                 Get parameter value
  load                Load table
  save                Save table
  storeinfo           Show store information
  clear               Clear/invalidate store slot
  lock                Lock a store
  unlock              Unlock a store
```

Some of these are combined in some procedures in this section.

9.2.1 Setting the TX frequency

Enter “param select tx” to select the tx table to work on. Entering “param list” will now show the tx table:

```
ax2150 # param select tx
ax2150 # param list
Table tx (5):
  0x0000 freq            U32 2245000000
  0x0004 if_freq         U32 465000000
  0x0008 baud            U32 4800
  0x0010 guard           U16 50
  0x0012 pllrang         U8  9
  0x0014 csp_hmac        BL  false
  0x0015 csp_rs          BL  true
  0x0016 csp_crc         BL  true
  0x0017 csp_rand        BL  true
  0x0018 csp_hmac_key    STR "0000000000000000"
  0x002E mix_curset      U16 4
  0x0030 preamb          U8  0xaa
  0x0031 preamblen       U8  50
  0x0032 preambflags     U8  56
  0x0033 intfrm          U8  0xaa
  0x0034 intfrmlen       U8  0
  0x0035 intfrmflags     U8  0x38
  0x0036 rssibusy        I16 -95
  0x0038 kup_delay       U16 0
  0x003A kup_mode        U8  0
  0x003C ber             FLT 0.000000
```

The content is the volatile working copy of the table.

Enter “param set freq 2250000000” to change the frequency,

Entering “param get freq” shows the value of the parameter.

```
ax2150 # param get freq
  freq = 2250000000
```

9.2.2 Save configuration table in every storage

To write the volatile confiuration into all stores the following procedure should be followed:

- Unlock the MCU FLASH storage by entering “param unlock flash”.
- Unlock the protected FRAM storage by entering “param unlock protected” (note, that it automatically enables lock upon boot).
- Change the parameters in the working table.
- Save the changed working table to the different stores.
- Enter “config update_default all” to write the tables.
- Reset AX2150.

The above example is shown in the following console printout:

```
ax2150 # param unlock flash
ax2150 # param unlock protected
ax2150 # config update_default all
Updated settings in FRAM for table 0, saving to file persistent: OK
Updated settings in FRAM for table 0, saving to file protected: OK
Updated settings in FLASH for table 0, saving to file flash: OK
Updated settings in FRAM for table 1, saving to file persistent: OK
Updated settings in FRAM for table 1, saving to file protected: OK
Updated settings in FLASH for table 1, saving to file flash: OK
Updated settings in FRAM for table 2, saving to file persistent: OK
```

(continues on next page)

(continued from previous page)

```
Updated settings in FRAM for table 2, saving to file protected: OK
Updated settings in FLASH for table 2, saving to file flash: OK
Updated settings in FRAM for table 5, saving to file persistent: OK
Updated settings in FRAM for table 5, saving to file protected: OK
Updated settings in FLASH for table 5, saving to file flash: OK
ax2150 # reset
```

10. References

[csp-client-man] CSP-client manual - gs-man-nanosoft-product-interface-application-3.0.2.pdf

[ax2150-datasheet] NanoCom AX2150 datasheet - gs-ds-nanocom-ax2150.pdf

11. Disclaimer

Information contained in this document is up-to-date and correct as at the date of issue. As GomSpace A/S cannot control or anticipate the conditions under which this information may be used, each user should review the information in specific context of the planned use. To the maximum extent permitted by law, GomSpace A/S will not be responsible for damages of any nature resulting from the use or reliance upon the information contained in this document. No express or implied warranties are given other than those implied mandatory by law.