This document describes how to read and write values into the register of the ACM-180-20 drive amplifier used in the ATA antennas.

Supporting Documents:

The following documents are describing various aspects of the drive amplifier module. These documents are located at our Git Hub FrontPage repository.

- Module datasheet, 2021 datasheet and 2004 datasheet:
  https://github.com/SETIatHCRO/Front-Page/blob/master/Antenna/Drive%20Box/Accelnet%20Documents/ACM-datasheet.pdf
  https://github.com/SETIatHCRO/Front-Page/blob/master/Antenna/Drive%20Box/Accelnet%20Documents/ACM-datasheet-old.pdf

- Module Parameter Dictionary:
  https://github.com/SETIatHCRO/Front-Page/blob/master/Antenna/Drive%20Box/Accelnet%20Documents/Parameter-Dictionary-rev05.pdf

- Module ASCII Programmers Guide:
  https://github.com/SETIatHCRO/Front-Page/blob/master/Antenna/Drive%20Box/Accelnet%20Documents/ASCII_Programmers_Guide_Manual.pdf

Table of Contents

Dr. Alexander Pollak | SETI INSTITUTE | alexander.pollak.87@gmail.com

Communication:

There are two ways to communicate with a drive amplifier and one way to communicate with the drive box daughter board. The first one is to use the Accelnet Module Utility it provides a read and write function. This is used to copy entire configurations from one module to another. The second one is by directly connecting to the amplifier via a serial telenet connection.

Accelnet Module Utility:

The accelnet module utility is installed on all control boxes and is used to read and write all registers of a specified module.

- ssh ataant@control
- ssh 5g
- cd ata-boxes/utils
- python3 accelnet.py -h

An example to read all parameters from the module would be:

```
ataant@ant5g:~/ata-boxes/utils$ python3 accelnet.py read az
```

An example to write all parameters from a file to the module would be:

```
ataant@ant5g:~/ata-boxes/utils$ python3 accelnet.py write az az-golden.dat
```

We ran into the problem that sometimes not all parameters were written correctly into the module. This will be improved in a newer version of the accelnet module utility. However, it is recommended to read back the values after programming and restarting the module. Then compare the registers with the file to make sure they are correct.

Direct Serial Connection:

It is possible directly connect to the serial interface of the module through the drive box daughter board. This allows to directly interact with the module. Each module, AZ and EL has an individual port.

Azimuth amplifier port: 1513
Elevation amplifier port: 1514

To connect directly to the module, follow these commands:

- ssh ataant@control
- ssh 5g
- telnet drivebox 1513

There are three main commands that are used to read write and restart the module:

Code	Command	Description
s	Set	Set a value of a parameter in ram or flash.
g	Get	Read the value of a parameter in ram or flash.
r	Reset	Reset the drive.

An example to read a register from the module would be:

```
ataant@ant5g:~$ telnet drivebox 1513
Trying 192.168.242.2...
Connected to drivebox.
Escape character is '^]'.
g r0xab
```

Drive Box Daughter Board Communication:

One can also connect to the drive box daughter board and get information about the bakes and other parameters of the drive box. Note that not all functions are implemented.

To connect to the drive box daughter board, follow these commands:

- ssh ataant@control
- ssh 5g
- telnet drivebox

An example to show all functions by typing help in the command line is shown below:

```
ataant@ant5g:~/ata-boxes/utils$ telnet drivebox
Trying 192.168.242.2...
Connected to drivebox.
Escape character is '^]'.
help

SYNOPSIS
    help [command]
DESCRIPTION
    shows specific help information for daughterboard commands:

    stop                - immediately stop azimuth and elevation drives
    getpulsefreq        - get azimuth or elevation drive pulse frequency
    setpulsefreq        - set azimuth or elevation drive pulse frequency
    getazpulsemode      - get azimuth drive pulse mode
    setazpulsemode      - set azimuth drive pulse mode (normal or alt.)
    getenable           - get azimuth or elevation drive enable state
    setenable           - set azimuth or elevation drive enable state
   (getadc)             - get voltage or temp. at ADC input 1, 2, 3, or 4
    getstatus           - get status of brakes, azimuth limits, ...
    gettime             - get daughterboard date and time
    getonboardtemp      - read daughterboard temperature sensor
    connecttomodule     - connect to Accelnet module RS-232 (serial) port
    reset               - reset daughterboard and Accelnet modules
    getsoftwareversion  - get version of software stored in ip2022 flash
    getiposheapstatus   - show statistics for ipOS internal heap
    getiposnetpagestatus - show statistics for ipOS internal netpages
    help                - show help information for commands

    Note: commands in parentheses are not implemented

EXAMPLES
    show help information for setpulsefreq command:
      help setpulsefreq
OPTIONS
```

To read the status of the brakes use the following command.:

```
getstatus azbrake
on
```

Note that OFF means the brake is energized, allowing free motion, supplied with 24VDC. ON means that the brake is not energized and preventing free motion, not supplied with any voltage (0VDC).

Read drive amplifier module manufacturing year:

This function is used to read out the manufacturing year of all drive amplifier that are installed in the ATA. We needed this function to debug differences between the original installed and manufactured module in 2005 and the new replaced modules in 2019 and 2021.

To read the manufacturing year of all modules installed in the ATA, follow these commands:

- ssh obs@control
- ~obs/tkoumrian/dbinfo-year.sh

The script then outputs the antenna number and az, el module manufacturing year.
If it can't read a value it outputs the error code.

```
1a az: 05
1a el: 05
.
.
.
2m az: 05
2m el: error: java.lang.Exception: Read timed out
.
.
.
5h az: 21
5h el: 21
```

Read drive amplifier module output configuration:

This function is used to read out the module register 0x70 and 0x71 of the module, which defines the configuration of output 1 and output 2. The two outputs are used to indicate an amplifier Fault condition (not implemented) and to control the brake.

To read the output configuration register for all modules installed in the ATA, follow these commands:

- ssh obs@control
- ~obs/tkoumrian/dbinfo-out.sh

The script then outputs the value stored in the flash register for each antenna and az, el module. Both register values are displayed in the same line separated by "/". (0x70 / 0x71). If it can't read a value it outputs the error code.

```
1a az: 256 4456575 / 256 16384
1a el: 512 0 / 256 16384
.
.
.
5h az: 256 4456575 / 256 16384
5h el: 512 0 / 256 16384
```

For detailed information about the output register values and function look at section "Module output control".

Module output control:

The ACM-180-20 module has two digital outputs, one is used to control the brake and the second one is connected to an input on the daughter board to indicate a fault condition. However, the fault indication is not implemented in the daughter board firmware.
There are two data sheets the original one from 2004 and the new one downloaded in 2024, see links at the beginning of this document. When comparing the output pin description one can see that the designation of the outputs has changed.

2004 version:

Signal	J2 Pin		Signal
RS-232 TxD	2	1	RS-232 RxD
Signal Ground	4	3	Signal Ground
CANH	6	5	CANL
Signal Ground	8	7	Signal Ground
Fault [OUT1]	10	9	[OUT2] Brake
Signal Ground	12	11	[IN10] HSInput
HSInput [IN9]	14	13	[IN8] HSInput
HSInput [IN7]	16	15	[IN6] HSInput
GPInput [IN2]	18	17	[IN4] GPInput
GPInput [IN3]	20	19	[IN1] GPInput
Signal Ground	22	21	Signal Ground
GPInput [IN5]	24	23	Hall V
Hall W	26	25	Hall U
Encoder /X	28	27	Encoder X
Encoder /B	30	29	Encoder B
Encoder /A	32	31	Encoder A

2024 version:

Signal	J2 PIN		SIGNAL
RS-232 TxD	2	1	RS-232 RxD
Signal Ground	4	3	Signal Ground
CAN_H	6	5	CAN_L
±10V Ref(+)	8	7	±10V Ref(-)
Brake [DOUT2]	10	9	[DOUT1] Fault
Signal Ground	12	11	[IN10] CAN-Bit 3
CAN-Bit 2 [IN9]	14	13	[IN8] CAN-Bit 1
CAN-Bit 0 [IN7]	16	15	[IN6] Capture
Fwd Enable [IN2]	18	17	[IN4] Home
Rev Enable [IN3]	20	19	[IN1] Enable
Signal Ground	22	21	Signal Ground
Motemp [IN5]	24	23	Hall V
Hall W	26	25	Hall U
Encoder /X	28	27	Encoder X
Encoder /B	30	29	Encoder B
Encoder /A	32	31	Encoder A

We could confirm by measuring the path on the on the drive box main board and daughter board that the following pins are connected to the brake and fault indication:

- PIN 9 => Brake
- PIN10 => Fault

This means that on the 2004 version **OUT2** is connected to the brake and on the 2024 version **DOUT1** is connected to the brake. The fault connection on the 2004 data sheet shows **OUT1** and on the 2024 version **DOUT2**.

The Module Parameter Dictionary describes the following labeling / numbering for the output ports:

- 0xAB [Each bit represents an output number. Bit 0 = **digital Output 0** (**OUT1**), bit 1 = **digital Output 1** (**OUT2**), etc., up to output n (OUT(n+1)), number of digital outputs on drive.]

Based on this we could confirm that **DOUT1** and **OUT2**, which both are used to designate PIN 9 correspond to output register 0x71. Furthermore, we could confirm that output register 0x71 controls the brakes for both modules old and new ones.

Register 0x70 to 0x77 are used to program the function of each output. The register uses two numbers, the first one to define the output configuration and the second one to add a bitmap when monitoring event registers.

```
g f0x71 v 256 16384
```

[get function] – [register (f for flash r for ram)] – [value] – [number one] – [number two]

The Module Parameter Dictionary describes the output configuration as following:

- 0x70 [First word is bit-mapped configuration value. Remaining words give additional parameter data used by output pin. Typically, second and third words are used as 32-bit bitmask to identify which bit(s) in Event Status Register (0xA0) output should follow. If any selected bits in Event Status Register (0xA0) are set, then output will go active. If no selected bits in Event Status Register (0xA0) are set, then output will be inactive.]

Note that this number is a combination of the value (decimal number) for bit 0-4 which means numbers between 0 and 21 and a bit set for bit 8 or 9. In our case we use bit 8 to invert the output, which means to set bit 8 we add a decimal value of 256 to the first number.
For the output configuration above (256) this means that the value written in the register is a combination of bits 0-4 value = 0 plus bit 8 is set and therefore the output is configured to:

- [bits 0-4 value = 0] Track bits in Event Status Register (0xA0)
- [bit 8 = 1] If set, inverts normal active state of output. E.g., outputs that are normally active low become active high.

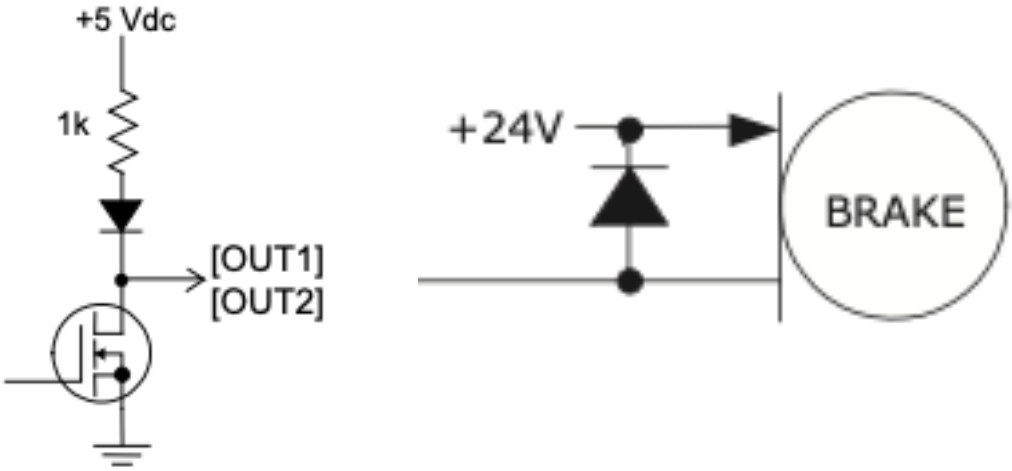
This means the output **DOUT1** tracks the bits in the status register 0xA0. The second number 16384 defines which bits are tracked by this output. In this case 16384 is the 14th bit in the register. And by looking into the description for register 0xA0 we can get the event which this output tracks now.

- 0xA0 bit 14 [Motor brake activated]

This means that the register setting above configures **DOUT1** to be inverted and track the motor brake activated register.

Module output brake wiring:

The brake for the elevation and azimuth drive is wired to PIN9 in the respective drive amplifier module. The brake is supplied with +24VDC and the output uses a MOSFET to either have a +5V pull up output or pull down the output pin to ground. The output can sink up to 1Amp.



The register 0xAB tracks the status of the output, it can be used to monitor if the bake is active or inactive. Using the 256 value in the output configuration the brake behaves as following:

- 0xAB bit1 = 0 [Pull down to GND, brake released, allows movement, OFF, +24V]
- 0xAB bit1 = 1 [Not pull down to GND, brake engaged, restricts movement, ON, 0V]

This means that if event register 0xA0 bit14 is 1 (motor brake activated) the output tracks this register and the bit in 0xAB will be set to 1, which means that the brake is ON.

Manual output control:

The drive amplifier module can also be configured to allow the user to control the outputs directly. This is done by writing the following parameter into the output configuration register 0x71:

```
s f0x71 258 0
```

This sets bit8 to 1 and inverts the output as well as sets the value 2 in the configuration bits (0-4). Setting the value 2 into the configuration leads to the following function:

- Value 2 [Track bits in Manual Output Control Register. See Output States and Program Control (0xAB)]

After restarting the module we now can control the output that we configured to be manual controllable, in this instance output **DOUT1** (0x71). To enable the brakes (ON) write the following command:

```
s r0xAB 2
```

This sets bit1 to high which corresponds to **DOUT1** or the output register 0x7. If the output bit is set to one the output is not pulled down and the brake restricts movement. Note that the value written into the register is a decimal number U32 which the corresponding bits of the number is used to control the outputs.

To read back the value either in manual control or event register control one can query the register by:

```
g r0xAB
```