CANDAC 16*16M

Revision 1.
Embedded software version 7.

1. Features

This device is designed for power supply control in control systems of accelerators as embedded intelligent controller. The device may be used as general purpose multi-channel digital-to-analog converter.

The device includes:

- 16 channels DAC;
- 8-channel output register with galvanically isolated outputs;
- 8-channel input register with galvanically isolated inputs;
- CANbus interface for interaction with control computer;
- micro-controller.

CANDAC16*16M

The device may be used whether sixteen independent digital-to-analog converters or a single multi-channel functional generator. User can store in on-board memory up to 8 files, which describes changing output voltages in time by using a linear interpolation method. A start of file procession may be initiated for all devices on line (or part of them) by broadcast message. All devices will process files synchronously. It is provided by internal circuitry with good stability.

Multiple channels in device are implemented by using a single DAC chip and 16 sample-and-holds. The device is intended to be incorporated in power supply rack. The device requires for proper operation the only power supply with voltage +5V (±5%).

2. Specifications:

1. Resolution – 16 bits.
2. Accuracy – 0.05%.
3. Output voltage- ±10V (unipolar range may be chosen by on-board jumper).
4. External load – 10 Kohm.
5. Time slice for table procession – 10 ms.
6. Files in on-board memory – 8.
7. Records in file – 30.
8. Accuracy of internal timer – 0.05%.
9. Channels of output register – 8.
10. Maximal working voltage for output register – 50V.
11. Maximal switched current- 32 mA.
12. Channels of input register- 8.
13. Voltage for input register- 2.5÷6.0V.
14. Input resistance for input register- 510 Ohm.
15. CANbus transceiver is galvanically isolated from network and it is in compliance with ISO 11898-24V (chip PCA82C251).
16. Voltage between transmission line and device- 1000V.
17. Hardware implementation allows using both standard and extended CANbus frames. Software implementation is based on standard frames (short identifier).
18. Baud rates- 1000, 500, 250, 125 Kbaud (may be chosen by jumpers).
19. Voltage of power supply- +5V, ±5%.
20. Power supply current- <1A (typical value- 0.7A).

3. External connection

The device is implemented as module in “WISHNYA” standard, width is 40 mm. A front panel of device contains a network connector (DB-9M), RESET button and two LEDs. One LED is blinking during processing CANbus message. The second LED is on during file procession. Connection with external channels of control and measurements carry out by DRB-37M connectors on back panel. Analog outputs of device are connected with pins of X1 connector. Outputs and inputs of registers are connected with X2 connector.

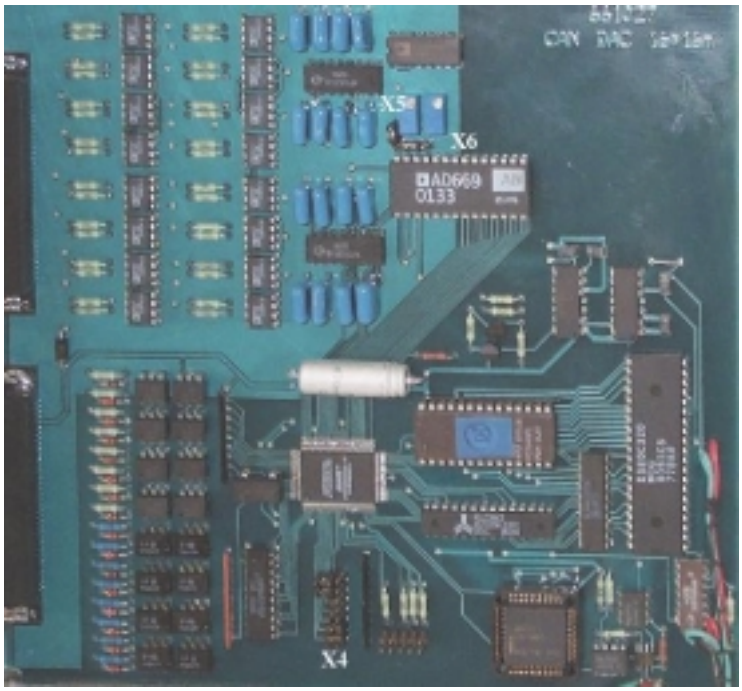
3.1. Jumpers

The device CANDAC 16*16 has the following set of jumpers:

X5 and X6 jumpers define output range of device and used reference source. **User should not change these jumpers.**

X4 includes 8 jumpers. 6 jumpers define number (address) of device in network (this number is used to compose identifier of messages) and 2 jumpers define baud rate.

Jumpers location is shown below on board photo.



Jumpers location on board

X5 defines used reference source. An external reference source is used in top position of jumper. An internal reference source is used in bottom position of this jumper.

X6 defines output range of DAC. A left position of jumper sets bipolar output range (from –10V to +10V). A right position of jumper sets unipolar output range (from –0V to +10V).

Destination of jumpers in X4 group.

Designation	Location	Destination
X4-7	Upper	N5- included in device number (most significant bit)
X4-6	…	N4- included in device number
X4-5	…	N3- included in device number
X4-4	…	N2- included in device number
X4-3	…	N1- included in device number
X4-2	…	N0- included in device number (least significant bit)
X4-1	…	BR1- defines baud rate
X4-0	Lower	BR0- defines baud rate

Jumpers N5…N0 defines logical number (address) of device which is used to compose message identifier for CANbus network (for more detail see PROTOCOL part of this description). An installed jumper should be interpreted as logical 0 and absence of jumper should be interpreted as logical 1.

Baud rate defining

BR1	BR0	Baud rate
Connected	Connected	1000 Kbit/sec
Connected	Disconnected	500 Kbit/sec
Disconnected	Connected	250 Kbit/sec
Disconnected	Disconnected	125 Kbit/sec

NOTES:

1. CANbus is bus with multiple access and incorrect baud rate setting may affect on transfer messages of other devices in addition to impossibility of access to this device.
2. In network may exist concurrently devices with identical numbers (addresses). Formally it is permissibly, but actually it do cause a lot of problem. Connecting to network devices with identical numbers is strictly not recommended.
3. It is forbidden to change X5 and X6 by user.

3.2 Front panel.

A front panel includes:
Line LED
Table LED
Reset button
CANbus connector

Line LED is blinking during processing CANbus messages by onboard processor.

Table LED is on during file procession procedure. It indicates process of autonomous changing output voltages.

After power-on the device blinks by both LEDs a few times.

Reset button is intended for hardware reset. It isn't intended for daily using.

CANbus connector (DB-9M) is intended for connection to media. Pin designations follows below in table.

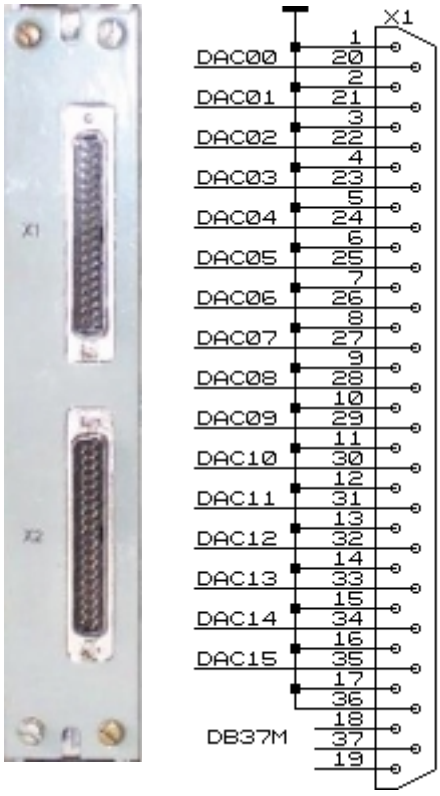
Shielded twisted pair is used as media. According to the ISO 11898-2 it should has a nominal impedance 120 Ohm. Line termination has to be provided through termination resistors of 120 Ohm located at both ends of the line.

2	CAN-L	One wire in pair
3	GND	Shield of cable
7	CAN-H	One wire in pair

3.3 Back panel.

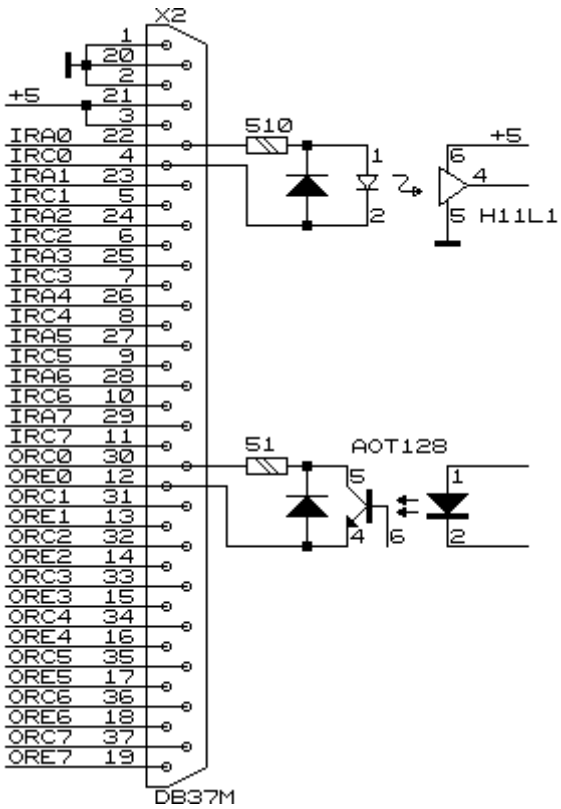
A two connectors (DRB-37M) are placed on the back panel. Connection external signals with device should be done through these connectors. X1 connector contains DAC outputs and X2 connector includes outputs and inputs of registers and power supply pins.

3.3.1 X1 connector.



X1 connector provides 16 pair of DAC outputs. These outputs are not galvanically isolated and they have common ground. A connection of DAC outputs with a destination should be done by twisted pair. An input resistance of load should not be less than 10 Kohm.

3.3.2 X2 connector.



X2 connector provides pins of input and output registers. Powering of the device should be done through pins of X2. The device requires the only external power supply with voltage +5V (±5%). There is shown a location of signals on the connectors pins. A fragment of circuitry is included to show an implementation of input and output registers. Both registers are designed with galvanic isolation which is provided by optocouplers.

Design of input register is based on a chip H11L1. It is intended for detection external voltage or current. Input voltage range is from 3V to 12V. Input current range is from 4mA to 20mA. Unconnected input (no current in LED) is interpreted as logical 0.

Output register design is based on transistor optocoupler and it is able to provide output current up to 32mA. Output voltage may be up to 30V.

4. Basics of operations for CANDAC 16*16M

The device includes multi-channel DAC, input register, output register and a micro-controller. The micro-controller integrates all parts together and provides a connection with a control computer by CANbus. Logically input and output registers are not connected with DAC and are controlled by specific messages. After power-up the micro-controller writes into output register zero.

An operation of DAC is quite complicated. After power-up the micro-controller writes into all channels of DAC codes corresponding half of scale. It means zero voltage for bipolar output range. All files in on-board memory are cleared by micro-controller because the device has no nonvolatile memory. After that micro-controller sends into line a message with his attributes.

A control computer can change each DAC channel independently from other. A message from line changes code in memory location corresponding required channel. After some time the micro-controller transfers this value into corresponding DAC channel. An uncertainty of this transfer is 10 msec.

In “File procession” mode all DAC channels in common way. A control computer writes in on-board memory a file which describes changing all output voltages in time. A micro-processor being started in “file procession” mode sequentially reads records from required file, calculates codes (voltages) for each DAC channel and for each time points and writes calculates codes into corresponding channels of DAC. A time quantum is 10 ms. It means that each 10 ms all channels of DAC change a voltage on their outputs (or keep on previous value). The device uses a linear approximation method. A control computer describes time points and micro-controller calculates intermediate values. This approach allows to reduce size of files.

The device can store up to 8 files in on-board memory. A file procession may be initiated by special message received from CANbus. The process may be started by address message and by broadcast message. A broadcast message can start all devices connected with line or part of them. To provide an opportunity of starting a group of devices files are labeled. A broadcast message also have a label. On receiving broadcast message “Start File” the device compares labels in command and in file. If labels are identical the will be taken for execution.

Each file consist of:

1. Table label
2. Table length (calculated by the device)
3. Records

A record describes a linear changing output voltages in time. Each record consist of an increment number (counter) which is the same for all channels, and 16 increments (values) for all channels. During record procession the micro-processor add an increment value to DAC code for each channel and subtract 1 from increment counter. After exhausting increment counter (when it reaches zero) micro-processor fetches next record from file and processes it. The process is completed by exhausting file.

Each DAC channel has appropriate accumulator in processor memory. An accumulator size is 4 bytes (32 bits). A two most significant bytes of accumulators are transferred by processor into DAC chip each 10 millisecond for converting to voltage. A two least significant bytes are used in file procession only to provide required accuracy. If an user don’t use this mode he can forget about these bytes. They are not significant.

A file procession may be paused any time. In this state (PAUSE state) an user has opportunity to correct an output voltages (by direct write to appropriate accumulators) and to correct the processed file (a following records) by address write commands. After these procedures the file procession may be resumed either from next time moment or from next record.

Notes:

1. DAC uses offset binary coding. A minimal binary code yields a minimal output voltage, a maximal binary code yields a maximal output voltage. A table below includes characteristic points.

Code (hexadecimal)	Voltage
FFFF	+9.9997 V
8000	0.0 mV
7FFF	-0.3 mV
0000	-10 V

2. When an user composes file and calculates increment values he should keep in mind that processor sums accumulator code and increment value as 32-bit unsigned integer numbers. To increase accumulator code on 2 an increment value should be 0002 (hexadecimal). To decrease accumulator code on 1 an increment value should be FFFFFFFF (hexadecimal). So, if increment value is zero then during processing this record the output voltage will not be changed.
3. An increment counter has 2 bytes size and it can contain numbers from 0 to 65535. A zero code (0) is interpreted by processor as 65536. So, each record can describe a behavior of output voltages no more than 11 minutes.
4. Each file has length 2Kb and contains no more than 30 records. It allows to describe processes not longer than 11 hours.

5. A command set for CANDAC 16*16M

Identifier bit distribution

Identifier bits	ID10…ID08	ID07...ID02	ID01…ID00
Bit field	Field 1	Field 2	Field 3
Destination	Priority	Address	Reserve

Comments to bit distribution:

Field 1 – priority field (type field):
Code 5 – a broadcast message (field 2 is ignored).
Code 6 – ordinary (address) message.
Code 7 – response (reply for type 6 message).
Code 0 is forbidden, other combination is not used (they are reserved for future extensions).

Field 2 – a physical address field. It defined address device (this address is defined by jumpers on-board).

Field 3 – extension field. User should set zero in this field. The device may set any combination of these bits.

Any device on receiving address message interprets an information by its content. If received message requires a reply, the device sends required information by message with code 6 (response type message). A broadcast messages should be received by all devices simultaneously and required actions should be done in all devices.

An interpretation of data fields:
On receiving message a device interprets data in following way: a first byte (byte 0) is descriptor of message, the other bytes are an additional information.

There is a list of message descriptors (codes are hexadecimal).
00 - 0F – write to accumulator of DAC channel 0 - 15
10 - 1F – request data from accumulator of DAC channel 0 - 15

F2 – address write to file
F3 – create of file
F4 – sequential write to file
F5 – close of file
F6 – request data from file
F7 – address start of file procession
F8 - request for data from input and output registers
F9 - write to output register
FE - device status request
FF - device attributes request

Detail description of messages for CANDAC16*16M
(all codes are hexadecimal)

Messages 00 - 0F – (write to accumulator of DAC channel 0-15), the following bytes are data bytes.

Example:

0A	12	80	00	00
10th channel	Byte 2	Byte 3	Byte 0	Byte 1

This message places into 10-th DAC channel value +18 (decimal).
3-th byte is the most significant byte, byte 0 – the least significant byte. If You don't use a file procession mode then values of two least bytes are not significant and may be any.

Messages 10 - 1F – (request code from accumulator of DAC channel 0-15). The following bytes will be ignored. In reply the device send a message with the same code but with data from accumulator (bytes 2, 3, 0, 1).

NOTE: Bytes 2 and 3 are transferred into DAC chip and bytes 0, 1 are used for calculations only in a file procession mode. If You don't use this mode then content of least bytes is not significant.

Message F2 – address write to file:

F2	Descriptor	Laddress	Haddress	Data0	Data1	Data2	Data3

Here
Descriptor- a file descriptor
Laddress, Haddress- least and most significant bytes of address in file for writing data.
Data…- up to 4 bytes of data.

Message F3 – create of file:
Byte 1 is file descriptor: the most 4 bites (used only 3 from 4) define a file number (physical number) and least 4 bites compose an identifier of file (for identification by broadcast messages).

Message F4 - sequential write to file up to 7 data bytes (look at file structure below).

Message F5 – close of file:
Byte 1 is file descriptor: the most 4 bites (used only 3 from 4) define a file number (physical number) and least 4 bites compose an identifier of file (in this message an identifier bit field may be ignored by device).

On receiving this message the device sends message with:
Byte 0 = F5 (the same byte as in request message), byte with file descriptor, least then most significant bytes of file length (physical). This message may be used to check if the file is loaded.

Message F6 – request data from file.
Here byte 1 is file descriptor, bytes 2 and 3 (least and most) compose requested data address in file. In reply the device sends message with data bytes (look at file structure below).

Message F7 – address start of file procession.
Byte 1 is file descriptor: the most 4 bites (used only 3 from 4) define a file number (physical number) and least 4 bites compose an identifier of file (for identification by broadcast messages).

Message F8 – request date from registers. This message has not additional information. In reply a device sends a message with output register byte and input register byte.

F8	Output Register Data	Input Register Data

Message F9 – write data to output register.
Byte 1 contains information to be written into output register. The device don't respond on this message.

F9	Output Register Data

Message FE – request device status. There is not additional information. In reply a device sends the following message:

FE	Status	Descriptor	Lpointer	Hpointer	Lsteps	Hsteps

Here:
Status- status of file procession. This byte is bit field. There are flags:
Bit 0 – RUN- file procession mode is running.
Bit 1 – FPRR- file procession request is received.
Bit 2 – PAUSE- file procession is paused by command and may be resumed.
Bit 3 – PCR- PAUSE command is received.
Bit 4 – RCR- RESUME command is received.
Bit 5 – GNCR- GO NEXT command is received.
Descriptor- file descriptor of file in process.
Lpointer, Hpointer- low and high bytes of pointer in processed file.
Lsteps, Hsteps- low and high bytes of increment counter in processed record.

In STATUS byte flags 3 and 4 are temporary. After receiving RESUME or GO_NEXT commands the device leaves PAUSE mode with delay 0-10 milliseconds. RCR and GNCR bits are intended to mark receiving appropriate commands.

The device sends this message after completion of file procession.

Message FF – device attribute request. There is not additional parameters. In reply a device sends the following message:

FF	Device Code	HW version	SW version	Reason

Device Code- device type (for CANDAC16*16 it is equal 1).
HW version- hardware version of device.

SW version- software version of device.
Reason- reason of sending this message:

0- After power-up.
1- After reset by button on front panel.
2- On request by address message with code FF.
3- On request by broadcast message (who is here?).
4- On restart by Watchdog timer.
5- On busoff recovery.

The device sends this message after power-up without any requests.

BROADCAST messages

For broadcast messages all devices analyze only field 1 in CANbus identifier. Valid combination is 5. A first byte of date present a broadcast command.. CANDAC16*16 uses the following broadcast commands:

1 – BREAK of file procession.
2 – START of file procession.
6 – PAUSE of file procession.
7 – RESUME (or GO_NEXT) file procession.
FF- request "Who is here". On this broadcast request all devices on-line must send into network message with their attributes (and identifier).

The BREAK command has no additional parameters. The START and PAUSE commands have an additional byte- a file identifier. In this byte the most 4 bites (used only 3 from 4) define a file number (physical number) and least 4 bites compose an identifier of file. The 7^{th} command has two additional bytes. The first byte is a file identifier. The second byte is a command modifier. If bit 0 of this byte is 0 then command is interpreted as RESUME command. If bit 1 of this byte is 0 then command is interpreted as GO_NEXT command. On receiving GO_NEXT command the device loads in working registers a next record of file before leaving PAUSE state.

FF- request "Who is here". On this broadcast request all devices on-line must send into network message with their attributes (and identifier).

FILE STRUCTURE and some comments

The device can store in on-board memory up to 8 files with no more than 30 records. An additional information (file labels and file lengths) are stored and transferred separately.

File structure

Address (byte)	Data
****	Records
0	Increment counter for record 0 (least significant byte)
1	Increment counter for record 0 (most significant byte)
2	Byte 0 of increment value for DAC 0
3	Byte 1 of increment value for DAC 0
4	Byte 2 of increment value for DAC 0
5	Byte 3 of increment value for DAC 0
6	Byte 0 of increment value for DAC 1

7	Byte 1 of increment value for DAC 1
8	Byte 2 of increment value for DAC 1
9	Byte 3 of increment value for DAC 1
…	…
62	Byte 0 of increment value for DAC 15
63	Byte 1 of increment value for DAC 15
64	Byte 2 of increment value for DAC 15
65	Byte 3 of increment value for DAC 15
66	Increment counter for record 1 (least significant byte)
67	Increment counter for record 1 (most significant byte)
****	And so on

After starting file procession the device fulfils the following actions:

1. Loading in a working area an increment counter and increment values.
2. Each time quantum (10 ms) the device adds increment values to DAC accumulators, loads results into DACs, decrements an increment counter.
3. The operations listed above are repeated up to exhausting increment counter.
4. If processed file is not completed the device loads the next record and repeats points 2 and 3.

File loading.

The files are files with sequential access during write operation and they are files with random access during read operation. The CREATE_FILE command erases previous records in this file and provides a sequential access for writing new data. On receiving data bytes the processor places them sequentially. If a maximal file size is exceeded then the processor ignores received data bytes. Closing file stops this process. All received data bytes will be ignored if there is not opened file. A CLOSE_FILE command may be used to check a presence of file and its length.

An ADDRESS_WRITE command don't require opening file.

The device can store 8 files with size up to 30 records.

6. Software versions for CANDAC 16*16M

There will be described modifications for software versions beyond 3rd.

Version 4.

1. After power up and on clicking RESET button the device blinks by all LEDs a few times.
2. A set of changes and additions concerning a file procession mode.
   a) An arithmetical calculation size (and file data) is increased to 32 bits and an increment counter is increased to 16 bit.
   b) There is added a special command (address write) to provide an opportunity to write corrections without reloading all file.
   c) There are added modes (and appropriate commands) PAUSE, RESUME, GO_NEXT.
   d) There is added an additional status information.
3. There was probability to read an incorrect accumulator value during a file procession mode. It is corrected.

Version 5.

1. Writing into inexisting accumulator might distort internal data. It is corrected.

Version 6.

1. Unknown address command might halt processor. It is corrected.

Version 7.

1. The device sends a status message (with code FE) after completion of file procession. This message isn't sent after breaking file procession by external command.
2. There is used Watchdog timer. If internal program is halted by any reason then Watchdog timer restart it from beginning. To differ this situation from other restarts the device sends status message with reason code 4.
3. A high error rate on the line may transfer a CANbus controller into busoff state. After detecting this state the device reinitializes the controller and sends status message with reason 5.