CAC208

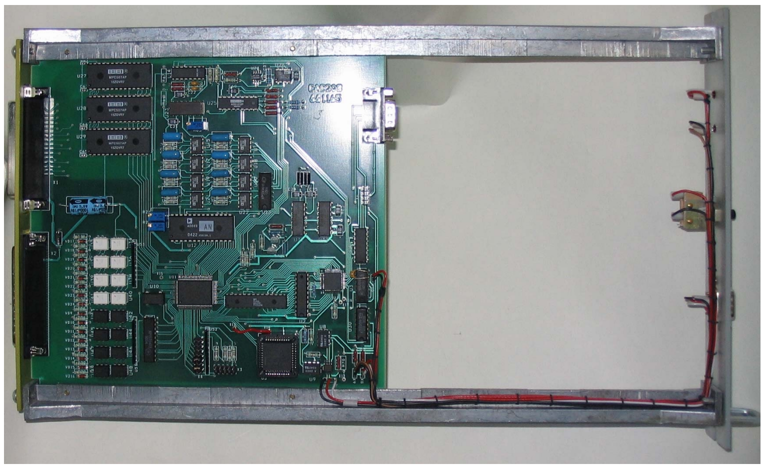
17-dec-2004.
`Embedded software version 1.`

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 a general purpose digital-to-analog and analog-to-digital converter.

The device includes:

- 16-bits 8-channels bipolar DAC;
- 20-channels ADC
- 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.



A photo of CAC208.

DACs of the device may be used whether eight 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 8 sample-and-holds.

As ADC the device may work in different modes. The base mode is a multi-channel mode. In this mode the device scans predefined channels, measures them, stores measured values in internal memory and sends data in CANbus (if it was required by mode). For investigation of powers supply behavior here may be used one-channel mode (digital oscilloscope mode). In this mode the device measures the only channel with defined time of measurements and sends these values to network. In order to record and analyze occasional spices of output current there may be used mode of continual recording. In this mode the device measures chosen channel and stores measured values in internal ring buffer with capacity 4096 values. A control computer can break the process any time, request a pointer of ring buffer, read date which was written before break and then analyze behavior of chosen channel. Actually, both latest modes are the one mode. The only difference is a value of flag in mode specification. This flag defines behavior of device- the information should be sent to network or it should be stored in ring buffer.

All ADCs in CANbus network or predefined group of ADC might be started in multi-channel mode simultaneously by broadcast message. This property is implemented by using labels. An user can assign label to device when it is started in multi-channel mode. Later, if device receives broadcast start message, it compares label in broadcast message with label assigned to multi-channel mode. If both label are identical the broadcast start accepts like address start command. The are stop command only address type or global type, group stops are not defined in protocol.

Hardware implementation of converter consist of delta-sigma ADC chip, and 20-channel bi-wire multiplexer. The device is intended to be embedded in power supply racks. The device requires for proper operation the only power supply with voltage +5V (±5%).

2. Specifications:

1. ADC resolution - 24 bits.
2. Effective resolution – from 15 bits (time of measurements is 1 ms) to 20 bits (time of measurements is 20 ms and more).
3. ADC offset error- 0.1 mV.
4. ADC accuracy- 0.003%.
5. ADC input ranges- ±10V (main range), ±1.0V, ±0.1V and ±0.01V (additional ranges).
6. ADC input current 1 nA.
7. ADC common-mode input range- 10.5V.
8. ADC common-mode rejection- 75 dB.
9. ADC time of measurements- from 1 ms to 160 ms.
10. DAC resolution - 16 bits.
11. DAC accuracy – 0.05%.
12. Output range for DAC - ±10V.
13. External load – 10 Kohm.
14. Time slice for table procession – 10 ms.
15. Files in on-board memory – 8.
16. Records in file – 30.
17. Accuracy of internal timer – 0.1%.
18. Channels of output register- 8.
19. Maximal voltage for output register- 50V.
20. Maximal current for output register- 16 ma.
21. Channels of output register- 8.
22. Voltage for input register- 2.5-6.0V.
23. Input resistance for input register- 510 Ohm.
24. Temperature sensitivity of on-board sensor (typical) – 1,9 mV/°C.
25. Output voltage of temperature sensor at +25 °C – 0.56 B ±10%.
26. CANbus transceiver is galvanically isolated from network and it is in compliance with ISO 11898-24V (chip PCA82C251).
27. Voltage between transmission line and device- 1000V.
28. Hardware implementation allows using both standard and extended CANbus frames. Software implementation is based on standard frames (short identifier).
29. Baud rates- 1000, 500, 250, 125 Kbaud (may be chosen by jumpers).
30. Voltage of power supply- +5V, ±5%.
31. Power supply current- <1.0A (typical value- 0.7A).

3. External connections

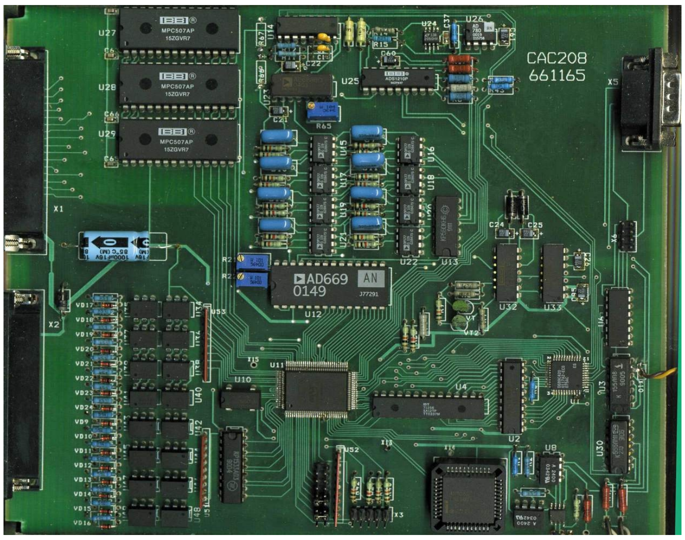
The device CAC208 is implemented as module in “WISHNYA” standard. A front panel of the device contains a network connector (DB-9M), RESET button and two LEDs. One LED is blinking during processing CANbus messages. The second LED is ON during file procession. Connection with external channels of control and measurements carry out by DRB connectors on back panel. Analog outputs and inputs of device are connected with pins of X1 connector. Outputs and inputs of registers are connected with X2 connector.

3.1. Jumpers

The device CAC208 has the following set of 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 on board photo.



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.

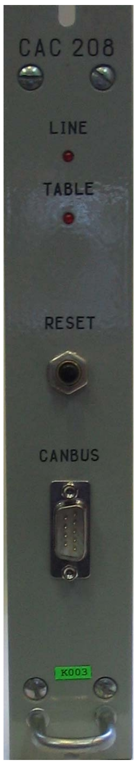
Задание скорости обмена с линией.

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.

3.2 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 all 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.

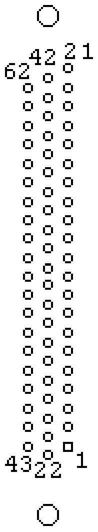
2	CAN-L	One wire in pair
3	GND	Shield of cable
7	CAN-H	One wire in pair

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

3.3 Back panel.

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

3.3.1 X1 connector.



62	INP1	42	INM0	21	INP0
61	INM2	41	INP2	20	INM1
60	INP4	40	INM3	19	INP3
59	INM5	39	INP5	18	INM4
58	INP7	38	INM6	17	INP6
57	INM8	37	INP8	16	INM7
56	INP10	36	INM9	15	INP9
55	INM11	35	INP11	14	INM10
54	INP13	34	INM12	13	INP12
53	INM14	33	INP14	12	INM13
52	INP16	32	INM15	11	INP15
51	INM17	31	INP17	10	INM16
50	INP19	30	INM18	9	INP18
49	DAC1	29	AGND	8	INM19
48	DAC3	28	AGND	7	DAC0
47	DAC5	27	AGND	6	DAC2
46	DAC7	26	AGND	5	DAC4
45	AGND	25	AGND	4	DAC6
44	PGND	24	PGND	3	AGND
43	E+5	23	PGND	2	PGND
		22	E+5V	1	E+5V

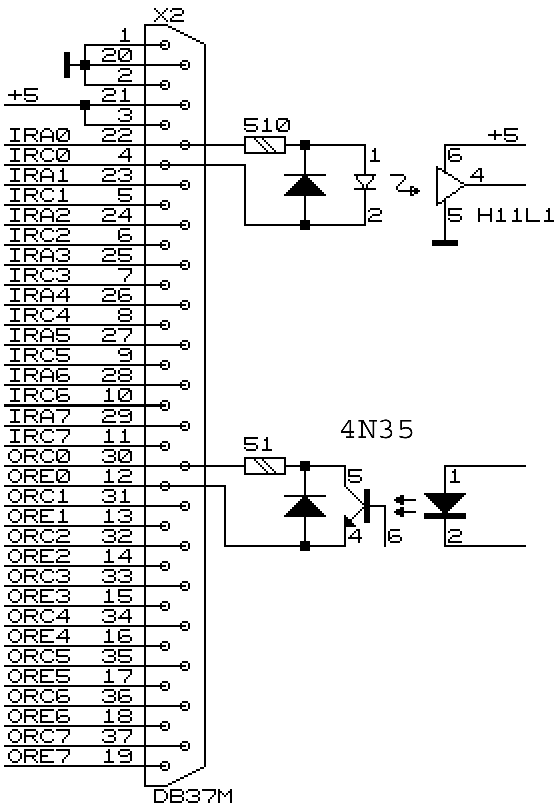
X1 (DHR-62F) connector provides 20 pair of ADC inputs, 8 pairs of DAC outputs and power supply pins. All analog outputs and inputs have common ground with a device ground.

Connections of ADC inputs and DAC outputs with a destination should be implemented by twisted pairs.

A mnemonics of designation is the following: INPx means "input positive x-number"; INMx means "input negative x-number".

Powering of device may be done with pins of X1 connector. In this case "ground" (negative) wire of power supply MUST be connected with PGND pins. AGND pins may be used only for signal connections.

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 that 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 CAC208

The device includes a multi-channel ADC, a multi-channels 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 ADC/DAC and are controlled by specific messages. After power-up the micro-controller writes to DAC a codes corresponding zero voltage on output pins, writes into output register zero and sends to host a power-up message.

In “File procession” mode a control computer writes in on-board memory a file which describes changing output voltage in time. A micro-processor being started in “file procession” mode sequentially reads records from required file, calculates DAC codes (voltages) for each time point and writes calculates codes into DAC chip. A time quantum is 10 ms. It means that each 10 ms DAC changes a voltage on its output (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 reducing 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 an output voltage in time. Each record consists of increment number (counter) and increment values. During record procession the micro-processor add an increment value to DAC code and subtract 1 from increment counter. After exhausting increment counter (when it reaches zero) micro-processor fetches next record from file and processes it. Exhausting file completes the process.

DAC has appropriate accumulator in processor memory. An accumulator size is 4 bytes (32 bits). A two most significant bytes of accumulator are transferred by processor into DAC chip each 10 millisecond for converting to voltage. 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 DAC code and increment value as 32-bit unsigned integer numbers. To increase accumulator code on 2 an increment value should be 00000002 (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 0,25 Kb and contains no more than 30 records. It allows describing processes not longer than 11 hours.

An operation of ADC is quite complicated also. A converter consists of an ADC chip, a reference source an analog multiplexer and a calibration reference source. There is used a delta-sigma ADC chip. A delta-sigma converter technology has some specific properties, which affects on operations of all devices. It is useful to observe these properties for good understanding of device operations.

Delta-sigma converters provide very high resolution with low noise level, but they have low stability. There is used a calibration procedure in order to compensate this instability. An on-board micro-controller performs the calibration procedure in hidden way from an user, but this procedure consumes an extra time and leads to delays in measurements.

Delta-sigma converters use very complicated digital signal processing and as a rule they cannot process any step change of input signal. If voltage is changing on significant value (or on unknown value) as it is in multi-channel measurements, first measured codes may not be authentic. To avoid errors due this effect the micro-controller discards unauthentic (or perhaps unauthentic) codes. This discarded codes are first 2-3 measured values in multi-channel mode.

The described peculiarities lead to two consequences. At first, if time of measurement is defined 20 ms, the micro-controller will send data into network (or write data into internal memory) with the same rate 20 ms in case of single-channel mode. In case of multi-channel measurements the micro-controller discards first three measured values after changing channel number. It means that actually data rate will be 80 ms. The second consequence is a result of calibration procedure. In digital oscilloscope mode processor performs the calibration procedure once, then it provides fast measurements for chosen channel. In multi-channel mode the calibration procedure is being performed each start of scanning (before processing first channel). This procedure leads to delay of measured data at 11-12 cycles (approximately 240ms for 20ms range).

An user should keep in mind that ADC effectively rejects an interference with period equal to time of measurements or more in integer times. It is good practice to use ranges 20ms or more. A high frequency interference are rejected by ADC and passive circuitry.

4.1. Base modes of CAC208

A converter of device can work in a few modes as it was mentioned above. The main mode is a multi-channel measurement mode. In this mode the device is involved by CANbus message with code 1. Bit fields of message specifies this mode in detail. They define a first and last channels in a scanning frame. There is a flag which points if it is required a single cycle of scanning or converter should scan channels up to STOP message. Another flag defines if a measured information should be sent in network or it should be stored in on-board memory. One byte contains a label of this mode to allow using a group start message.

In multi-channel mode the device performs a calibration procedure before any measurements. Then micro-controller connects an ADC with input channel defined as first, performs a few measurements, discards possible invalid values, stores correct value in on-board memory and, if it defined, sends it into network. A latest measured data can be requested from on-board memory. Each input channel has a personal location in the memory and on external request the micro-controller sends into network a contents of requested location. If requested channel was not measured at all the contents would have an arbitrary value. After processing first channel in frame the micro-processor connects ADC to next input channel and process it. When the last channel in frame was processed micro-controller or goes to idle state (for single frame case) or begins all actions from calibration procedure and so on. If multi-channel mode request contained label (not equal zero) a broadcast message can start measurements all devices with the same value of label simultaneously.

Don't forget that in multi-channel mode an output data rate is four times slower than it should be with defined time of measurements.

For investigation of powers supply behavior here may be used one-channel mode (digital oscilloscope mode). In this mode the device is involved by CANbus message with code 2. Bit fields of message specify this mode in detail. They chose a measured channel, a time of measurement. There is a flag that points if it is required a single measurement or converter should work up to STOP message. Another flag defines if a measured information should be sent in network or it should be stored in ring-buffer of on-board memory. After receiving an one-channel mode command the micro-controller performs a calibration procedure and then begin measure a chosen channel. A data rate corresponds to chosen time of measurements in this mode. Measurements may be stopped by message with code 0. If data are sending into network they aren't storing in on-board memory.

If "send to line" flag is zero then flag "continuous" is considered as 1. So, the device perform continuous measurements and stores data in on-board memory. These values can be requested by message with code 4. A current value of ring-buffer pointer can be requested by message "Status request". The counter point at measurement location but not at byte.

Actually, ADC of device has 24 measurement channels. A 20 from them are intended for an external connection and connected with connector's pins. Four inputs have on-board connections. 22nd channel is connected with output temperature sensor, 23rd channel is connected with power supply, 21st channel is connected with measurements "Ground" and 20th channel is connected with an output of calibration reference source (it voltage is +10V). 20th and 21st channels are used by calibration procedure. Logically all 24 channels are equal. An user can measure any combination of channels. On-board temperature sensor is not intended for accurate measurements. For relative temperature measurements user should read its voltage after power-up and use this value as reference.

Note:

An ADC data is coded as 24-bit signed integer value. A correspondence between codes and voltages is seen in a table below. An user should take in consideration that CANADC40*24 allows some over voltage without loosing accuracy. An user should keep it in mind to reach compatibility of software.

Code (hexadecimal)	Voltage
3FFFFF	+10 V
000000	+0.0 V
FFFFFF	-0.0 V
C00000	-10 V

5. A command set for CAC208

Identifier bits 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: User should set zero in this field. The device can send messages with different values in this field.

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 - break a measurements procedure
01 - defines and starts multi-channel measurements
02 - define and start one-channel measurements
03 - request of a multi-channel value from on-board memory (measured before request)
04 - request for value from a ring-buffer
80 – 87 – write code to DAC with channel number 0-7
90 – 97 – request code from DAC with channel number 0-7
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
FD - DAC status request
FE - device status request
FF - device attributes request

Detail description of messages for CAC208
(all codes are hexadecimal)

Message 00 – break a measurements procedure. There is not additional information. Addressed devices should not reply on this message.

Message 01 – configuration and start multi-channel measurements procedure. The message looks as:

01	ChBeg	ChEnd	Time	Mode	Label

ChBeg- first channel of multi-channel frame.
ChEnd- end channel of multi-channel frame. Channel numbers are from 0 to 7.
Time- time of measurements code. Valid codes are from 0 to 7.
Mode- bit flags to detail procedure.
Label- label for group start command. If label is 0 it means “no label”.
Mode includes the following flags:
Bits 0 and 1 defines gain for even (0, 2…) channels.
Bits 2 and 3 defines gain for odd (1, 3…) channels.
Bit 4: 0 means single scanning cycle; 1 means continuous measurements (up to STOP message or message 1 or 2).
Bit 5: 0 means that measured values should be stored in on-board memory and should be sent into network. If this bit is 0 then measured values should not be sent into network.

If a multi-channel measurements was started, a device sends measured data in following message (if bit 5 in Mode is set):

01	Attribute	Low byte	Middle byte	High byte

Byte Attribute consist of a measured channel number (least 6 bits) and a gain code (most 2 bits). Other 3 bytes contain measured value.

Message 02 – configuration and start one-channel measurements procedure. The message looks as:

02	Channel	Time	Mode

Channel- consist of channel number (least 6 bits) to be measured and gain code (most 2 bits). Channel numbers are from 0 to 39.
Time- time of measurements code. Valid codes are from 0 to 7.
Mode- bit flags to detail procedure.
Mode includes the following flags:
Bit 4: 0 means single measurement; 1 means continuous measurements (up to STOP message or message 1 or 2).
Bit 5: 1 means that measured values should be stored in on-board memory and should be sent into network. If this bit is 0 then measured values should not be sent into network and device stores them in ring-buffer.

Comment: if bit 5 in mode byte is 0 then bit 4 will be ignored (No sense to store a single measurement).

If a one-channel measurements was started, a device sends measured data in following message (if bit 5 in Mode is set):

02	Attribute	Low byte	Middle byte	High byte

Byte Attribute consist of a measured channel number (least 6 bits) and a gain code (most 2 bits). Other 3 bytes contain measured value.

Message 03 – request data from multi-channel buffer (request for previous measured and stored data). The message looks as:

03	Channel

Channel- a channel number. A request refer to previous measured data for channel specified in this command.

As a response a device sends measured data in following message:

03	Attribute	Low byte	Middle byte	High byte

Byte Attribute consist of a measured channel number (least 6 bits) and a gain code (most 2 bits) which was set at measurement moment. Other 3 bytes contain measured value.

Message 04 – request date from ring-buffer. The message looks as:

04	Low byte	High byte

Here low and high bytes compose pointer at current measurement in a ring-buffer. The ring-buffer can hold 4096 measurement values. A micro-controller begins to store measured data with pointer value 0. After write al last address (4095) a micro-controller continues storing data from first address (0) again. For correct interpretation data (oldest and youngest) user should read current value of ring-buffer pointer. It may be done by "request status" message (code FE).

As a response a device sends measured data in following message:

04	Attribute	Low byte	Middle byte	High byte

Byte Attribute consist of a measured channel number (least 6 bits) and a gain code (most 2 bits) which was set at measurement moment.. Other 3 bytes contain measured value.

80 - 87 – write code to DAC and accumulator with channel number 0-7. The message looks as:

8A	12	80	80	80
10th channel	Byte 3	Byte 2	Byte 1	Byte 0

This message contains data bytes to be written to 10th channel number of DAC and accumulator. Byte3 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. This message don't require reply.

90 - 97 – request code from DAC and accumulator with channel number 0-7. The message looks as:

9A

The device responds in reply by:

9A	Byte 3	Byte 2	Byte 1	Byte 0

Here 4 data bytes are code from accumulator.

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 4 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 4 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 FD – request DAC status. There is not additional information. In reply a device sends the following message:

FD	Status	Descriptor	Lpointer	Hpointer	Lsteps	Hsteps	CalLabel

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.
Bit 6 – reserved.

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 FE – request device status. It hasn't any parameters. In reply a device sends the following message:

FE	Dev. Mode	Label	Low pADC	High pADC	File ident.	Low pDAC	High pDAC

Here:

Device Mode- bit field. There are flags:

Bit 4 – SCAN- if this flag is set it means that a device process multi-channel measurements procedure.

Bit 3 – RUN- if this flag is set it means that a device process measurements procedure (multi-channel or one-channel).

Bit 2 – Calibration- this flag indicates that DAC calibration procedure is in process.

Bit 1 – FPRR- file procession request is received.

Bit 0 – FRUN- file procession mode is running.

Label- label value for ADC.

Low and high bytes of pADC compose a ring buffer pointer. It points at location that would be updated by next measurement. If ring-buffer was overwritten (after long time) the pointer points at oldest data.

File ident- file descriptor of file in process. It is valid only during file procession.

Low pDAC, High pDAC- low and high bytes of pointer in processed file.

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 CAC208 it is equal 4).

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.

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.. CDAC20 uses the following broadcast commands:

1 – BREAK of file procession.
2 – START of file procession.
3 - STOP- to stop measurements procedure.
4 - group START, group code (label) is placed in second data byte.
5 –

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 7th 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.

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.

A file looks as:

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
…	…
32	Byte 0 of increment value for DAC 7
33	Byte 1 of increment value for DAC 7
34	Byte 2 of increment value for DAC 7
35	Byte 3 of increment value for DAC 7
36	Increment counter for record 1 (least significant byte)
37	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.

Some additional information.

Channel gain is defined as shown in table:

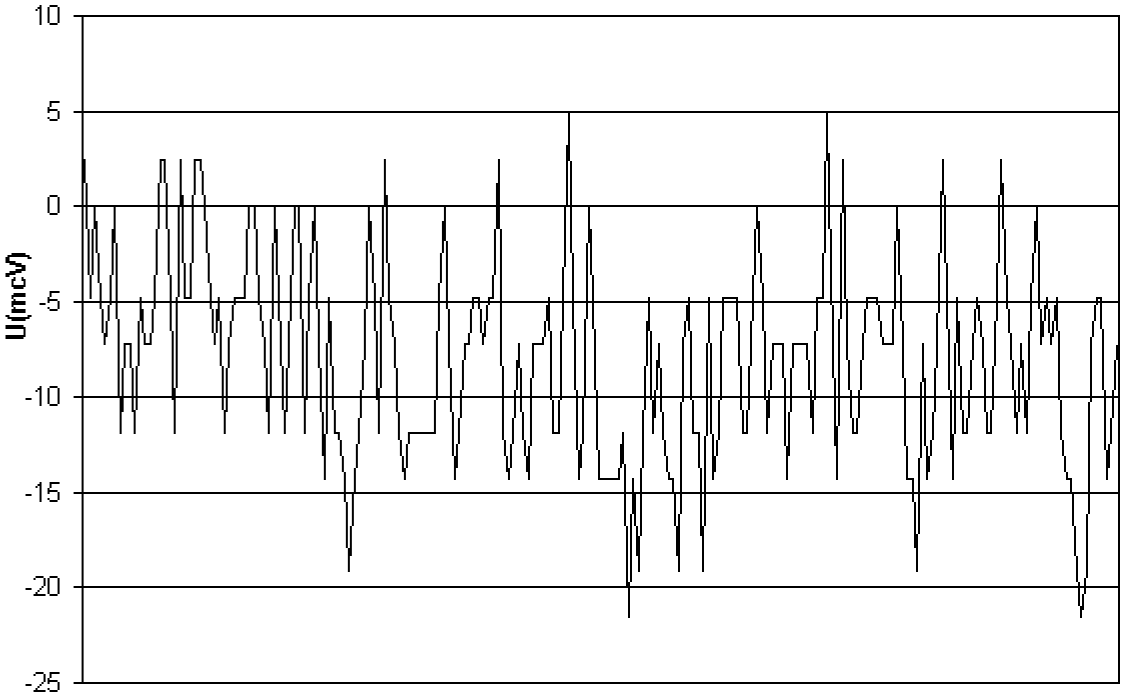
Binary code	Gain
00	1
01	10
10	100
11	1000

You should use gain 1 and 10. Using other ranges is not recommended.

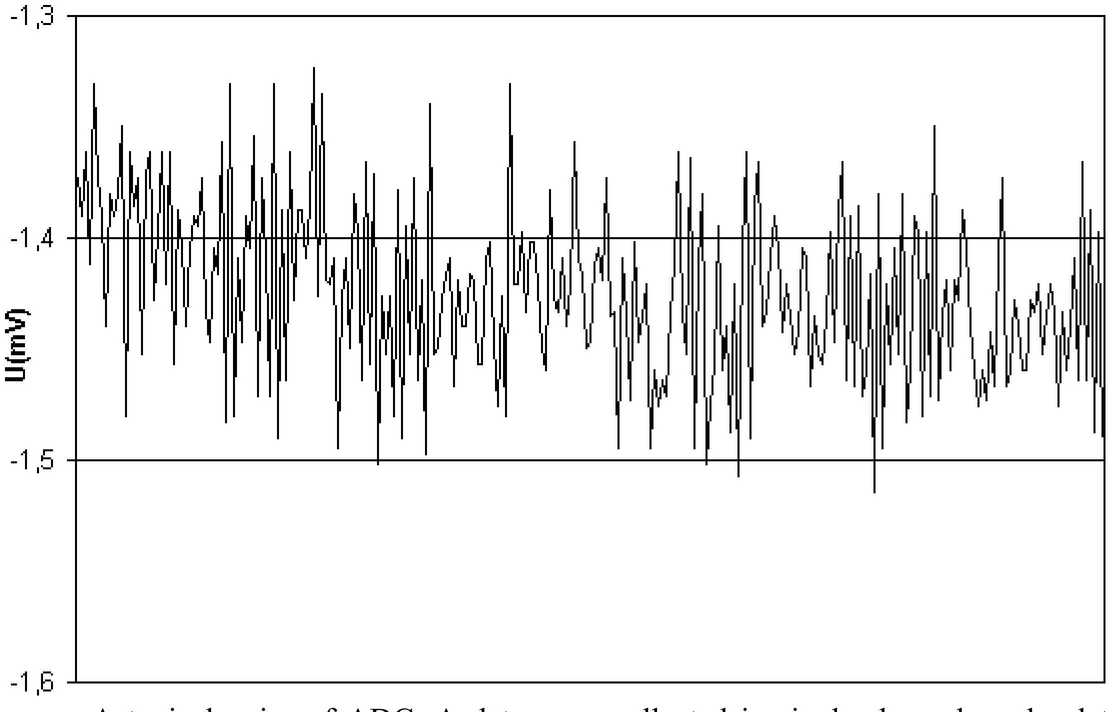
Measurement time

Code decimal	Measurement Time
0	1 ms
1	2 ms
2	5 ms
3	10 ms
4	20 ms
5	40 ms
6	80 ms
7	160 ms

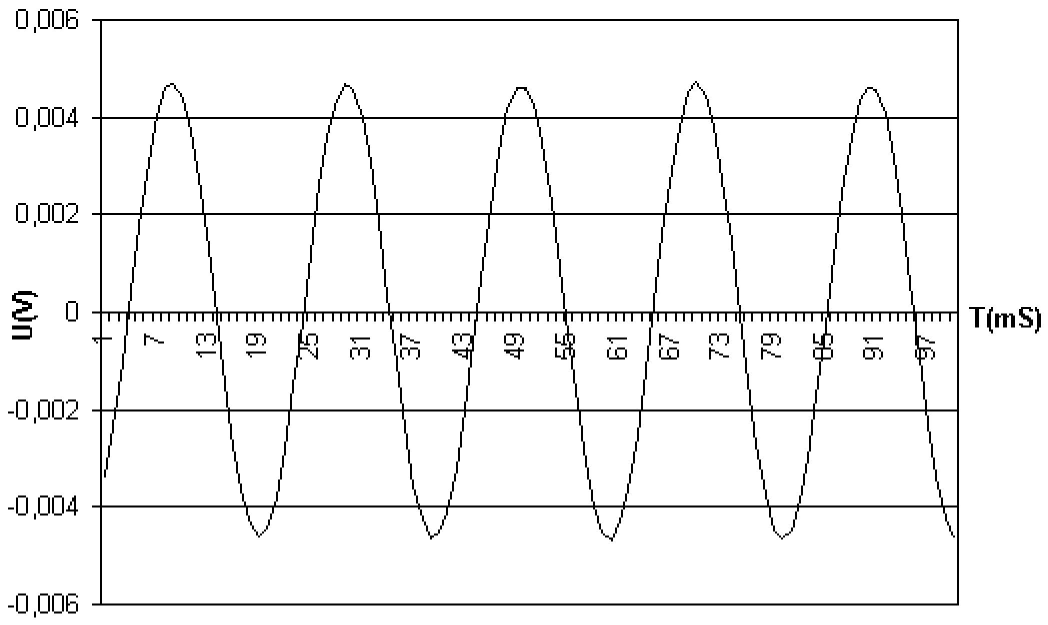
6. Some typical performance curves for CAC208



A typical noise of ADC. A data were collected in single-channel mode, data rate was 20 ms/measurement.



A typical noise of ADC. A data were collected in single-channel mode, data rate was 1 ms/measurement.



A fast measurements of low-level signal. A data were collected in single channel mode, data rate was 1 kHz (time of measurements was 1 ms/point).

7. Software versions for CAC208

There will be described modifications for software versions beyond 2nd.

Version 2.

1.