CEAC124

1-apr-2011.
`Embedded software version 4.`

Content

1. Main 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 4-channels bipolar DAC;
- 12-channels precise ADC
- 4-channel output register with galvanically isolated outputs;
- 4-channel input register with galvanically isolated inputs;
- CANBUS interface for interaction with control computer;
- On-board micro-controller.

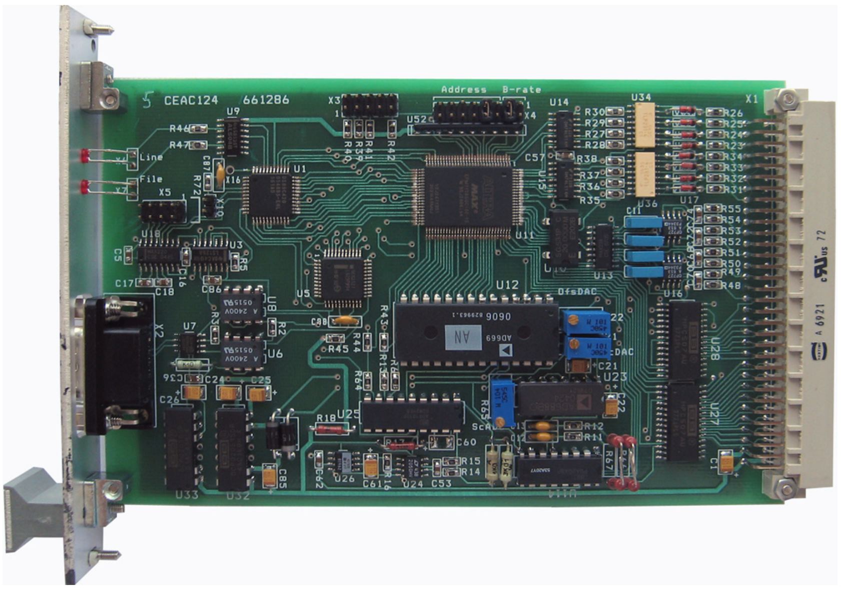


Photo of CEAC124

DACs of the device may be used whether four independent digital-to-analog converters or a single multi-channel functional generator. User can store in on-board memory a file, 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 (with uncertainty less than 10 msec). It is provided by internal circuitry with good stability. Multiple channels in device are implemented by using a single DAC chip and 4 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 128 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. A 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 labels are identical the broadcast start accepts like address start command. There 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 12-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 (main parameters):

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 - 50 mcV.
4. ADC accuracy (for integration time 20 mS and more) - 0.003%
5. ADC accuracy (for integration time 1 mS) - 1.0%
6. ADC input ranges- ±10V (main range), ±1.0V, ±0.1V and ±0.01V (additional ranges).
7. ADC input current 1 nA.
8. ADC common-mode input range- 10.5V.
9. ADC common-mode rejection- 75 dB.
10. ADC time of measurements- from 1 mS to 160 mS.
11. DAC resolution - 16 bits.
12. DAC output settling time – 0.1 Sec.
13. DAC accuracy – 0.05%.
14. Output range for DAC - ±10V.
15. External load – 10 KOhm.
16. Time slice for file procession - 10 mS.
17. Files in on-board memory -1.
18. Records in file e- 27.
19. Accuracy of internal timer 0.01%.
20. Uncertainty of starting file execution relative to receiving START command - 10 mS.
21. Channels of output register – 4.
22. Maximal voltage for output register – 50 V.
23. Maximal current for output register - 5÷10 mA.
24. Channels of input register – 4.
25. Voltage for input register- 2.5-6.0V.
26. Input resistance for input register- 510 Ohm.
27. Input-output isolation voltage for registers - 1500 B.
28. Temperature sensitivity of on-board sensor (typical) – 1,9 mV/°C.
29. Output voltage of temperature sensor at +25 °C – 0.56 B ±10%..
30. CANbus transceiver is galvanically isolated from network and it is in compliance with ISO 11898-24V (chip PCA82C251).
31. Voltage between transmission line and device- 1000V.
32. Baud rates- 1000, 500, 250, 125 Kbaud (may be chosen by jumpers).
33. Voltage of power supply- +5V, ±5%.
34. Power supply current- <0,8 A (typical value- 0.5A). Power supply must provide starting current > 1,2 A.
35. Size of device - 3U * 160 mm.

3. External connections

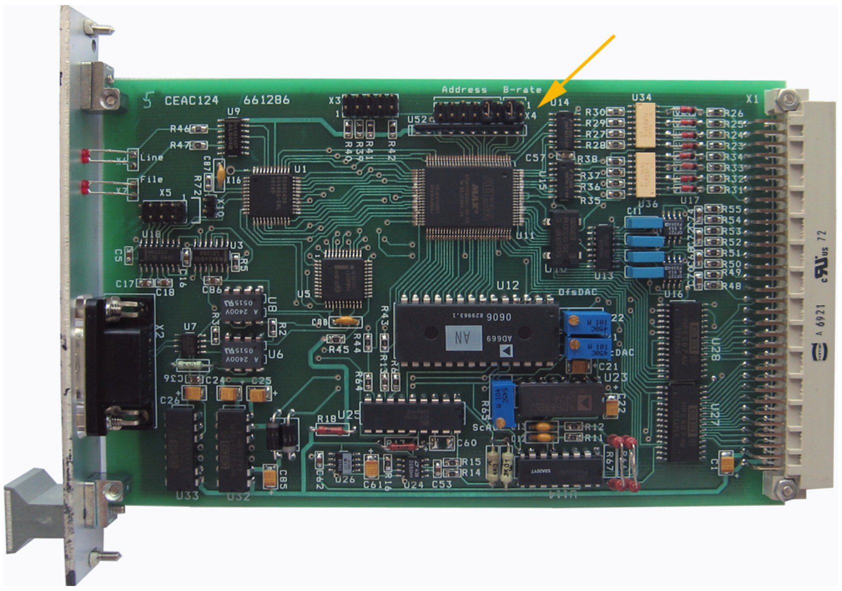
The device CEAC124 is implemented as eurocard 3U*160. A front panel of the device contains a network connector (DB-9M) and two LEDs. One LED is blinking during processing CANbus messages. The second LED is ON during file procession. Connection with external channels of control and measurements carry out by rear connector.

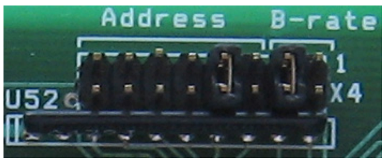
3.1. Jumpers

The device CEAC124 has the only set of jumpers- X4.

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.

Jumper 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.

Don't use addresses 34, 3C, 3D, 3E и 3F (hexadecimal).

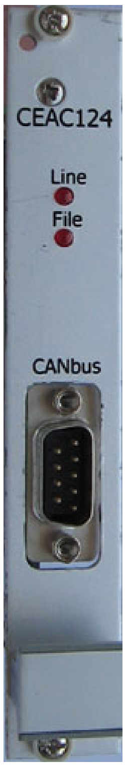
Baude rate coding.

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 accesses and incorrect baud rate setting may affects on transfer messages of other devices up to impossibility of access to the device.
2. In network may exist concurrently devices with identical numbers (addresses). Formally it is permissibly, but actually it does 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
File LED
CANbus connector

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

File 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. Both LEDs are blinking during “bus-off recovery” procedure also.

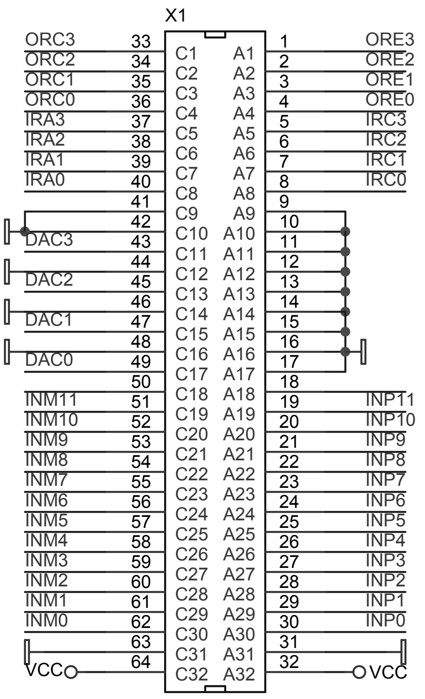
CANbus connector (DB-9M) is intended for connection to media. Pin designations follow below in table.

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 by termination resistors of 120 Ohm located at both ends of the line.

3.3 Main connector

There is DIN 41612 connector for connecting with input and output signals. Both analog and digital signals use the same connector.



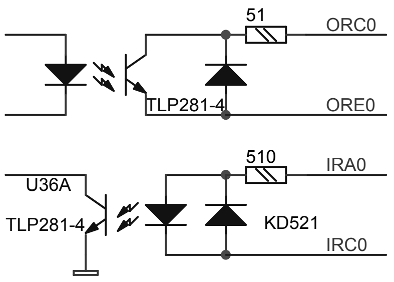
The main connector provides for user 12 pair of ADC inputs, 4 pair of DAC outputs, input and output registers and pins for powering device.

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”.

The device uses the only external power supply +5B (5%). For powering device one should use pins 31, 32, 63 and 64. Other “ground” pins are connected with analog “ground” and should not be used for powering device.

Below is shown a fragment of circuitry input/output registers. Both registers are designed with galvanic isolation that is provided by optocouplers. Both registers use TLP281 optocouplers.



Input register 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 is able to provide output current up to 10 mA. Output voltage may be up to 30V.

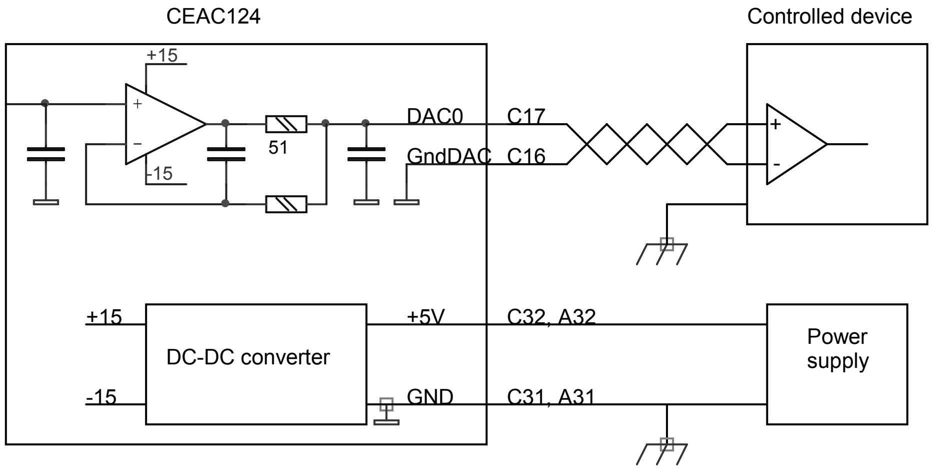
4. Basics of operations for CEAC124

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. Moreover, microcontroller initiates multichannel measurement mode of ADC with 20 mS integration time, scanning channels from 0 to 15 and without sending data into line. It allows avoid ADC initialization in simple applications.

There are described main features of analog-to-digital and digital-to-analog converters of device.

4.1. Digital-to-analog converter of CEAC124

In simple applications with low accuracy requirements user can connect DAC to external device quite careless. But in precise applications this connection must be performed correctly. Digital-to-analog converter has common "ground" with other resources of device. For correct transfer DAC output signal to end equipment user should use differential connection with DAC output and his "personal ground" pin. Below is shown "how to do" correct connection of DAC and controlled device. There is shown output circuitry for better understanding of connection.



The digital-to-analog converter may be used as "simple DAC" when code received from line immediately converted to output voltage. This mode is obvious and don't need any comments.

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 all DACs change a voltage on its 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 reducing size of files and traffic in line.

The file contains no real voltages but set of increments (decrements). It means if control program lose control context it should overwrite correctly DAC accumulator before starting file procession.

The device can store only single file 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 has a label. On receiving broadcast message "Start File" the device compares labels in command and in file. If labels are identical the command 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.

DACs have appropriate accumulators in processor memory. An accumulator size is 4 bytes (32 bits). A two most significant bytes of accumulators are converted to output voltages each 10 millisecond. Two least significant bytes are used in file procession only to provide required accuracy. If a user doesn't use this mode he can forget about these bytes. They are not significant in "simple DAC mode".

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

FILE STRUCTURE and some comments

The device can store in on-board memory the only file with no more than 27 records. Additional information (file labels and file lengths) are stored and transferred separately.

Собственно файл

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
…	…
14	Byte 0 of increment value for DAC 3
15	Byte 1 of increment value for DAC 3
16	Byte 2 of increment value for DAC 3
17	Byte 3 of increment value for DAC 3
18	Increment counter for record 1 (least significant byte)
19	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, convert results into output voltages, 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 file is file with sequential access during standard write/read operations. The CREATE_FILE command erases previous records in the 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 doesn't require opening file.

The device can store 1 file with size up to 27 records.

Notes:

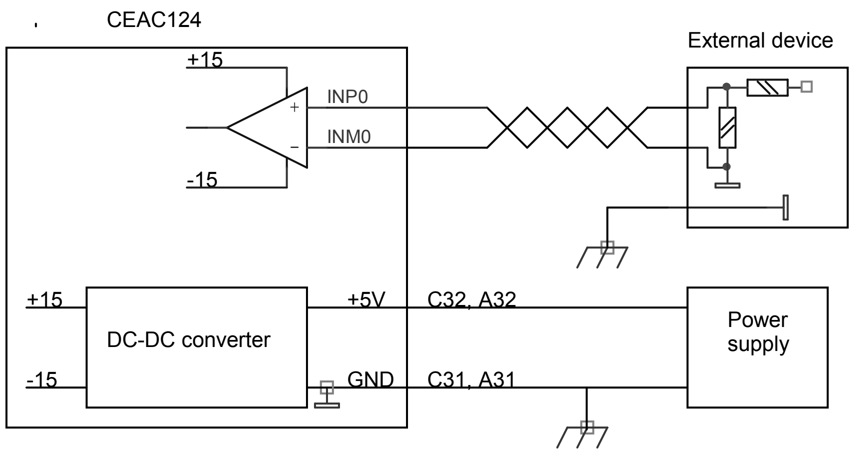
1. DAC uses offset binary coding. A minimal binary code yields a minimal output voltage, a maximal binary code yields a maximal output voltage. A table below shows characteristic points.

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

2. When a 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 by 2 an increment value should be 00000002 (hexadecimal). To decrease accumulator code by 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. The file has length 0,5 Kb and contains no more than 27 records.

4.2. Analog-to-digital converter of CEAC124

Analog-to-digital converter of CEAC124 is intended for differential measurements. The picture below may help to connect ADC with signal source more correctly.



Analog-to-digital 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 a 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. These discarded codes are first 4 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 four measured values after changing channel number. It means that actually data rate will be 100 ms. The second consequence is a result of calibration procedure. In digital oscilloscope mode processor performs the calibration procedure once, and 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).

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

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 specify this mode in detail. They define 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 measured information should be sent to 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 content 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 five 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 measured information should be sent in network or it should be stored in ring-buffer of on-board memory. After receiving a 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 performs 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 pointer points at measurement location but not at byte.

Notes:

Actually, ADC of device has 16 measurement channels. A 12(In0÷In11) from them are intended for an external connection and are connected with connector's pins. Four inputs have on-board connections. 12th channel is connected with output temperature sensor, 13th channel is connected with power supply, 15th channel is connected with measurements "Ground" and 14th channel is connected with an output of calibration reference source (it voltage is +10V). 14th and 15th channels are used by calibration procedure. Logically all 16 channels are equal. A 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.

An ADC data is coded as 24-bit signed integer value. A correspondence between codes and voltages is seen in a table below. A user should take in consideration that some devices allow some overvoltage without loosing accuracy. A 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

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

5. Protocol (interaction with CANbus) of CEAC124

Распределение битов идентификатора

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). Don't use addresses 34, 3C, 3D, 3E and 3F (hexadecimal).

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. The device doesn't check size of message except for F4 command (sequential write to file).

Size of transmitted messages is not defined and may be different in different versions of device.

Data interpretation.

Receiving CANbus message device interprets data the following way: the first byte (byte 0) is interpreted as descriptor of message (command) and the following bytes are parameters.

5.1. List of commands (hexadecimal codes).

00 - break a measurements procedure
01 - defines and starts multi-channel measurements
02 - defines and starts 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
E7 – RESUME (cancel PAUSE state) , *from software version 4*
EB – PAUSE (file procession should be paused) , *from software version 4*
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
FB – BREAK (file procession should be breaked) , *from software version 4*
FD - DAC status request
FE - device status request
FF - device attributes request

5.2. Detail description of messages (commands) (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 15.
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), which was used during measurement. Other 3 bytes contain measured value.

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

02	Channel	Time	Mode

Channel- consists of channel number (least 6 bits) to be measured and gain code (most 2 bits). Channel numbers are from 0 to 15.
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 single-channel measurements were 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.

Data in internal buffer are stored in the same format (4 bytes of data and attributes).

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 refers 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 data from ring-buffer. The message looks as:

04	Low byte	Middle byte

Here low and high bytes compose pointer at current measurement in a ring-buffer. The ring-buffer can hold 128 measurement values. A micro-controller begins to store measured data with pointer value 0. After write al last address (127) 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 consists 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 - 83 – write code to DAC and accumulator with channel number 0-3. The message looks as:

83	80	12	80	80
3-й канал	Байт 3	Байт 2	Байт 1	Байт 0

This message contains data bytes to be written to 3rd channel 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 doesn’t require reply.

90 - 93 – request code from DAC and accumulator with channel number 0-3. The message looks as:

92

The device responds in reply by:

92	Byte 3	Byte 2	Byte 1	Byte 0

Here 4 data bytes are code from accumulator from 2nd channel of DAC.

Note: if there was not used file procession two least bytes have random values.

Message E7 –RESUME (cancel PAUSE state), *from software version 4*
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).

E7	Descriptor

EB –PAUSE (suspend file procession), *from software version 4*
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).

EB	Descriptor

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.
Address write don't need opened file.

Message F3 – create (open) 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).

F3	Descriptor

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

F4	Data	Data	Data	Data	Data	Data	Data

Bytes to be written are defined by message length. If there isn't opened file the messages would be discarded.

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).

F5	Descriptor

On receiving this message the device sends the following message:

F5	Descriptor	Low byte	High byte

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.

F6	Descriptor	Laddress	Haddress	Data0	Data1	Data2	Data3

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).

Message F7 - START of file procession.

F7	Descriptor

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.

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

F9	Output Register Data

Message FB –BREAK (break file procession), *from software version 4*
This message has not additional information.

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

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 – reserved for CALIBRATION flag.

Bit 1 – TableR – file procession request is received.
Bit 0 – Table – 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 CEAC124 it is equal 20).
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.

5.3. GLOBAL messages (broadcasts)

For broadcast messages all devices analyze only field 1 in CANbus identifier. Valid combination is 5. A first byte of date presents a broadcast command. CEAC124 uses the following broadcast commands:

1 – BREAK of file procession in all devices. There is no additional parameters.
2 – START of file procession in labeled group.
There is 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.
3 - STOP- to stop ADC measurements procedure.
4 - group START of ADC measurements, group code (label) is placed in second data byte.
5 -
6 – PAUSE of file procession in labeled group, group code (label) is placed in second data byte.
7 – RESUME (or GO_NEXT) file procession in labeled group.
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.

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

FF	Device Code	HW version	SW version	Reason

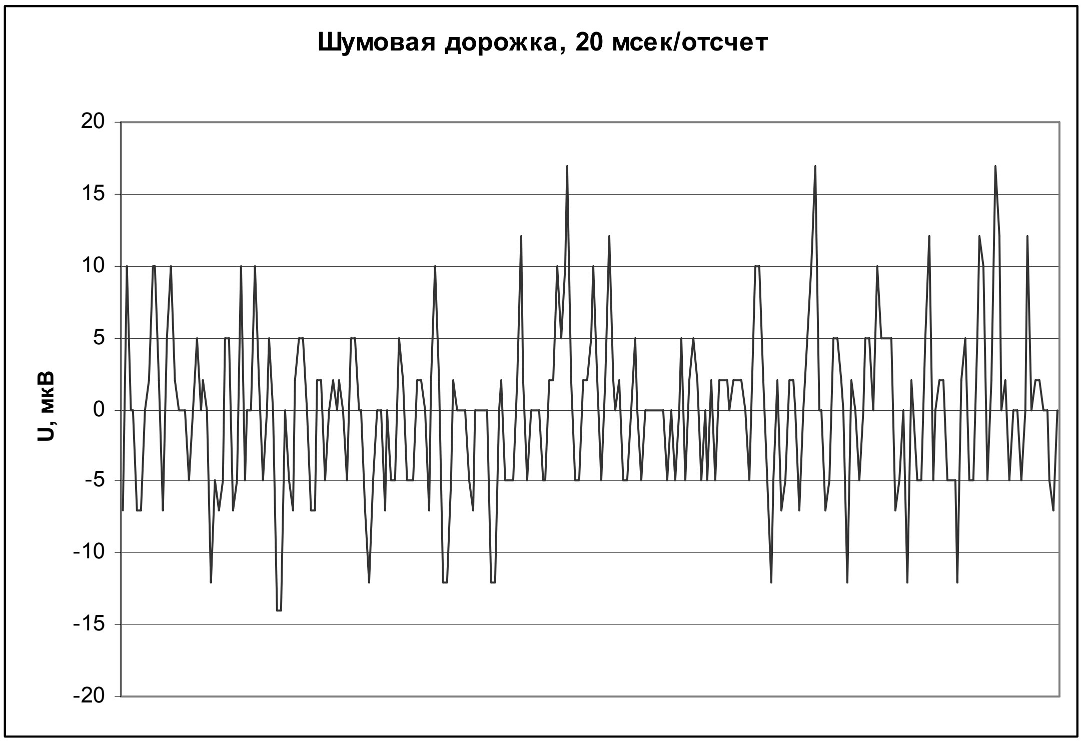
Device Code- device type (for CEAC124 it is equal 20).
HW version- hardware version of device.
SW version- software version of device.

Reason- reason of sending this message:

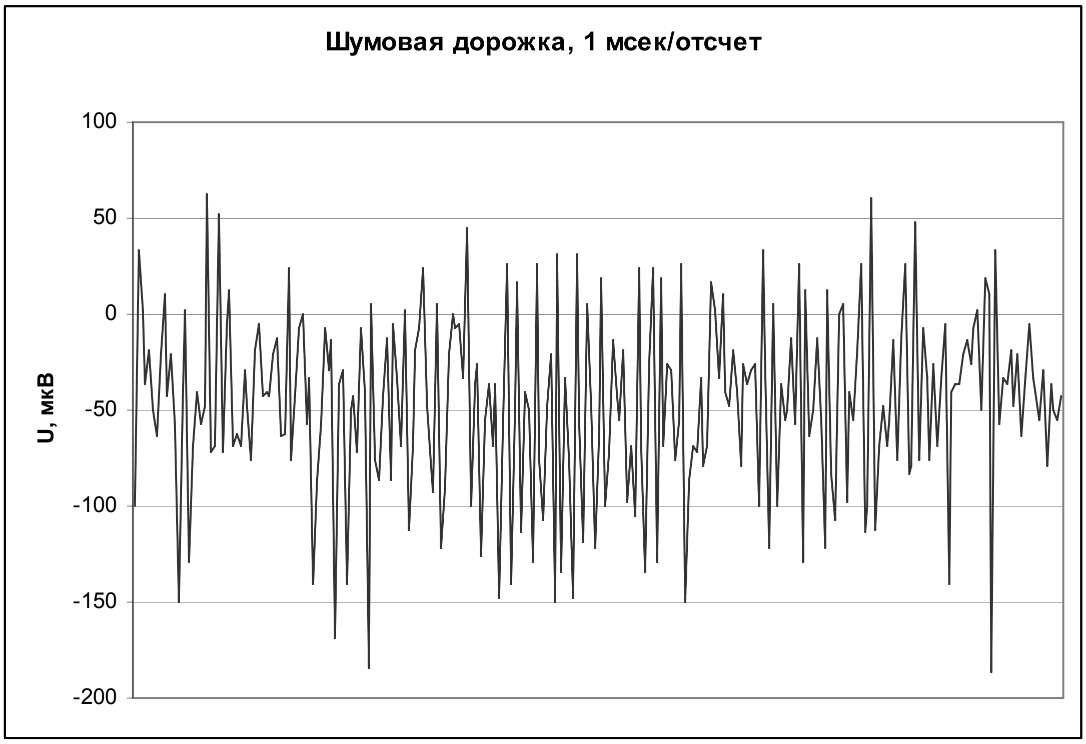
- 0 - After power-up.
- 1 - After reset by button on front panel.
- 2 - On request by address message with code FF.
- 3 - On request by broadcast message (who is here?).
- 4 - On restart by Watchdog timer.
- 5 - On busoff recovery.

6. Some typical pictures for CEAC124

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.



7. Software versions for CEAC124

Version 2.

Measurements channels above 15 corrupts DAC data (and output voltages). It is corrected.

Version 3.

File procession is debugged and corrected message FD on file processing completion.

Version 4.

New commands are added- E7, EB, FB.