Communication protocol specification

Protocol description

Controller can be controlled from the PC using serial connection (COM-port). COM-port parameters are fixed controller-side:

Speed: 115200 baud;
Frame size: 8 bits;
Stop-bits: 2 bits;
Parity: none;
Flow control: none;
Byte receive timeout: 400 ms;
Bit order: little endian;
Byte order: little endian.

Command execution

All data transfers are initiated by the PC, meaning that the controller waits for incoming commands and replies accordingly. Each command is followed by the controller response, with rare exceptions of some service commands. One should not send another command without waiting for the previous command answer.
Commands are split into service, general control and general information types.
Commands are executed immediately. Parameters which are set by Sxxx commands are applied no later than 1ms after acknowledgement.
Command processing does not affect real-time engine control (PWM, encoder readout, etc).
Both controller and PC have an IO buffer. Received commands and command data are processed once and then removed from buffer. Each command consists of 4-byte identifier and optionally a data section followed by its 2-byte CRC. Data can be transmitted in both directions, from PC to the controller and vice versa. Command is scheduled for execution if it is a legitimate command and (in case of data) if its CRC matches. After processing a correct command controller replies with 4 bytes - the name of processed command, followed by data and its 2-byte CRC, if the command is supposed to return data.

Controller-side error processing

Wrong command or data

If the controller receives a command that cannot be interpreted as a legitimate command, then controller ignores this command, replies with an "errc" string and sets "command error" flag in the current status data structure. If the unrecognized command contained additional data, then it can be interpreted as new command(s). In this case resynchronization is required.
If the controller receives a valid command with data and its CRC doesn't match the CRC computed by the controller, then controller ignores this command, replies with an "errd" string and sets "data error" flag in the current status data structure. In this case synchronization is not needed.

CRC calculation

CRC is calculated for data only, 4-byte command identifier is not included. CRC algorithm in C is as follows:

```
unsigned short CRC16(INT8U *pbuf, unsigned short n)
{
  unsigned short crc, i, j, carry_flag, a;
  crc = Oxffff;
  for(i = O; i < n; i++)
  {
    crc = crc ^ pbuf[i];
    for(j = O; j < 8; j++)
    {
      a = crc;
      carry_flag = a & OxOOO1;
      crc = crc >> 1;
      if ( carry_flag == 1 ) crc = crc ^ OxaOO1;
    }
  }
  return crc;
}
```

This function receives a pointer to the data array, pbuf, and data length in bytes, n. It returns a two byte CRC code.

Transmission errors

Most probable transmission errors are missing, extra or altered byte. In usual settings transmission errors happen rarely, if at all. Frequent errors are possible when using low-quality or broken USB-cable or board interconnection cable. Protocol is not designed for use in noisy environments and in rare cases an error may match a valid command code and get executed.

Missing byte, controller side
A missing byte on the controller side leads to a timeout on the PC side. Command is considered to be sent unsuccessfully by the PC. Synchronization is momentarily disrupted and restored after a timeout.

Missing byte, PC side
A missing byte on the PC side leads to a timeout on PC side. Synchronization is maintained.

Extra byte, controller side
An extra byte received by the controller leads to one or several "errc" or "errd" responses. Command is considered to be sent unsuccessfully by the PC. Receive buffer may also contain one or several "errc" or "errd" responses. Synchronization is disrupted.

Extra byte, PC side
An extra byte received by the PC leads to an incorrectly interpreted command or CRC and an extra byte in the receive buffer. Synchronization is disrupted.

Altered byte, controller side
An altered byte received by the controller leads to one or several "errc" or "errd" responses. Command is considered to be sent unsuccessfully by the PC. Receive buffer may also contain one or several "errc" or "errd" responses. Synchronization can rarely be disrupted, but is generally maintained.

Altered byte, PC side
An altered byte received by the PC leads to an incorrectly interpreted command or CRC. Synchronization is maintained.

Timeout resynchronization

If during packet reception next byte wait time exceeds timeout value, then partially received command is ignored and receive buffer is cleared. Controller timeout should be less than PC timeout, taking into account time it takes to transmit the data.

Zero byte resynchronization

There are no command codes that start with a zero byte ('\0'). This allows for a following synchronization procedure: controller always answers with a zero byte if the first command byte is zero, PC ignores first response byte if it is a zero byte. Then, if synchronization is disrupted on either side the following algorithm is used:
In case PC receives "errc", "errd" or a wrong command answer code, then PC sends 4 to 250 zeroes to the controller (250 byte limit is caused by input buffer length and usage of I2C protocol, less than 4 zeroes do not guarantee successful resynchronization). During this time PC continuously reads incoming bytes from the controller until the first zero is received and stops sending and receiving right after that.
Received zero byte is likely not a part of a response to a previous command because on error PC receives "errc"/"errd" response. It is possible in rare cases, then synchronization procedure will start again. Therefore first zero byte received by the PC means that controller input buffer is already empty and will remain so until any command is sent. Right after receiving first zero byte from the controller PC is ready to transmit next command code. The rest of zero bytes in transit will be ignored because they will be received before controller response.
This completes the zero byte synchronization procedure.

Zero byte synchronization procedure

Synchronization is performed by means of sending zero ('\0') bytes and reading bytes until a zero byte is received. Optionally one may clear port buffer at the end of synchronization procedure. Initially 64 zero bytes are sent. If there were no zero bytes received during the timeout, then a string of 64 bytes is sent 3 more times. After 4 unsuccessful attempts and no zero bytes received device is considered lost. In this case library should return rexult_nodevice error code. In case of successful syncronization library returns rexult_error.

Controller settings setup

Functions for adjusting engine read/write almost all controller settings.

Command SFBS

```
rexult_t xet_feedback_xettingx (device_t id, conxt feedback_xettingx_t* feedback_xettingx)
```

Command code (CMD): "sfbs" or 0x73626673.

Request: (18 bytes)

INT32U	CMD	Command
INT16U	IPS	The number of encoder counts per shaft revolution. Range: 1..655535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.
INT8U	FeedbackType	Type of feedback
		0x01 - FEEDBACK_ENCODER (Feedback by encoder.)
		0x03 - FEEDBACK_ENCODERHALL (Feedback by Hall detector.)
		0x04 - FEEDBACK_EMF (Feedback by EMF.)
		0x05 - FEEDBACK_NONE (Feedback is absent.)
INT8U	FeedbackFlags	Flags
		0x01 - FEEDBACK_ENC_REVERSE (Reverse count of encoder.)
		0x02 - FEEDBACK_HALL_REVERSE (Reverse count position on the Hall sensor.)
		0xC0 - FEEDBACK_ENC_TYPE_BITS (Bits of the encoder type.)
		0x00 - FEEDBACK_ENC_TYPE_AUTO (Auto detect encoder type.)
		0x40 - FEEDBACK_ENC_TYPE_SINGLE_ENDED (Single ended encoder.)
		0x80 - FEEDBACK_ENC_TYPE_DIFFERENTIAL (Differential encoder.)
INT16U	HallSPR	The number of hall steps per revolution.
INT8S	HallShift	Phase shift between output signal on BLDC engine and hall sensor input(0 - when only active the Hall sensor, the output state is a positive voltage on the winding A and a negative voltage on the winding B).
INT8U	Reserved [5]	Reserved (5 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description: Feedback settings.

Command GFBS

```
rexult_t get_feedback_xettingx (device_t id, feedback_xettingx_t* feedback_xettingx)
```

Command code (CMD): "gfbs" or 0x73626667.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT16U	IPS	The number of encoder counts per shaft revolution. Range: 1..655535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.
INT8U	FeedbackType	Type of feedback
		0x01 - FEEDBACK_ENCODER (Feedback by encoder.)
		0x03 - FEEDBACK_ENCODERHALL (Feedback by Hall detector.)
		0x04 - FEEDBACK_EMF (Feedback by EMF.)
		0x05 - FEEDBACK_NONE (Feedback is absent.)
INT8U	FeedbackFlags	Flags
		0x01 - FEEDBACK_ENC_REVERSE (Reverse count of encoder.)
		0x02 - FEEDBACK_HALL_REVERSE (Reverce count position on the Hall sensor.)
		0xC0 - FEEDBACK_ENC_TYPE_BITS (Bits of the encoder type.)
		0x00 - FEEDBACK_ENC_TYPE_AUTO (Auto detect encoder type.)
		0x40 - FEEDBACK_ENC_TYPE_SINGLE_ENDED (Single ended encoder.)
		0x80 - FEEDBACK_ENC_TYPE_DIFFERENTIAL (Differential encoder.)
INT16U	HallSPR	The number of hall steps per revolution.
INT8S	HallShift	Phase shift between output signal on BLDC engine and hall sensor input(0 - when only active the Hall sensor, the output state is a positive voltage on the winding A and a negative voltage on the winding B).
INT8U	Reserved [5]	Reserved (5 bytes)
INT16U	CRC	Checksum

Description:
Feedback settings.

Command SHOM

```
rexult_t xet_home_xettingx (device_t id, conxt home_xettingx_t* home_xettingx)
```

Command code (CMD): "shom" or 0x6D6F6873.

Request: (33 bytes)

INT32U	CMD	Command
INT32U	FastHome	Speed used for first motion. Range: 0..100000.
INT8U	uFastHome	Part of the speed for first motion, microsteps.
INT32U	SlowHome	Speed used for second motion. Range: 0..100000.
INT8U	uSlowHome	Part of the speed for second motion, microsteps.
INT32S	HomeDelta	Distance from break point.
INT16S	uHomeDelta	Part of the delta distance, microsteps. Range: -255..255.
INT16U	HomeFlags	Set of flags specify direction and stopping conditions.
		0x001 - HOME_DIR_FIRST (Flag defines direction of 1st motion after execution of home command. Direction is right, if set; otherwise left.)
		0x002 - HOME_DIR_SECOND (Flag defines direction of 2nd motion. Direction is right, if set; otherwise left.)
		0x004 - HOME_MV_SEC_EN (Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.)
		0x008 - HOME_HALF_MV (If the flag is set, the stop signals are ignored in start of second movement the first half-turn.)
		0x030 - HOME_STOP_FIRST_BITS (Bits of the first stop selector.)
		0x010 - HOME_STOP_FIRST_REV (First motion stops by revolution sensor.)
		0x020 - HOME_STOP_FIRST_SYN (First motion stops by synchronization input.)
		0x030 - HOME_STOP_FIRST_LIM (First motion stops by limit switch.)
		0x0C0 - HOME_STOP_SECOND_BITS (Bits of the second stop selector.)
		0x040 - HOME_STOP_SECOND_REV (Second motion stops by revolution sensor.)
		0x080 - HOME_STOP_SECOND_SYN (Second motion stops by synchronization input.)
		0x0C0 - HOME_STOP_SECOND_LIM (Second motion stops by limit switch.)
		0x100 - HOME_USE_FAST (Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.)
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set home settings. This function send structure with calibrating position settings to controller's memory.

Command GHOM

```
rexult_t get_home_xettingx (device_t id, home_xettingx_t* home_xettingx)
```

Command code (CMD): "ghom" or 0x6D6F6867.

Request: (4bytes)

INT32U	CMD	Command

Answer: (33 bytes)

INT32U	CMD	Command (answer)
INT32U	FastHome	Speed used for first motion. Range: 0..100000.
INT8U	uFastHome	Part of the speed for first motion, microsteps.
INT32U	SlowHome	Speed used for second motion. Range: 0..100000.
INT8U	uSlowHome	Part of the speed for second motion, microsteps.
INT32S	HomeDelta	Distance from break point.
INT16S	uHomeDelta	Part of the delta distance, microsteps. Range: -255..255.
INT16U	HomeFlags	Set of flags specify direction and stopping conditions.
		0x001 - HOME_DIR_FIRST (Flag defines direction of 1st motion after execution of home command. Direction is right, if set; otherwise left.)
		0x002 - HOME_DIR_SECOND (Flag defines direction of 2nd motion. Direction is right, if set; otherwise left.)
		0x004 - HOME_MV_SEC_EN (Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.)
		0x008 - HOME_HALF_MV (If the flag is set, the stop signals are ignored in start of second movement the first half-turn.)
		0x030 - HOME_STOP_FIRST_BITS (Bits of the first stop selector.)
		0x010 - HOME_STOP_FIRST_REV (First motion stops by revolution sensor.)
		0x020 - HOME_STOP_FIRST_SYN (First motion stops by synchronization input.)
		0x030 - HOME_STOP_FIRST_LIM (First motion stops by limit switch.)
		0x0C0 - HOME_STOP_SECOND_BITS (Bits of the second stop selector.)
		0x040 - HOME_STOP_SECOND_REV (Second motion stops by revolution sensor.)
		0x080 - HOME_STOP_SECOND_SYN (Second motion stops by synchronization input.)
		0x0C0 - HOME_STOP_SECOND_LIM (Second motion stops by limit switch.)
		0x100 - HOME_USE_FAST (Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.)
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Description:
Read home settings. This function fill structure with settings of calibrating position.

Command SMOV

```
rexult_t xet_move_xettingx (device_t id, conxt move_xettingx_t* move_xettingx)
```

Command code (CMD): "smov" or 0x766F6D73.

Request: (30 bytes)

INT32U	CMD	Command
INT32U	Speed	Target speed (for stepper motor: steps/s, for DC: rpm). Range: 0..100000.
INT8U	uSpeed	Target speed in microstep fractions/s. Using with stepper motor only.
INT16U	Accel	Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 1..65535.
INT16U	Decel	Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 1..65535.

INT32U	AntiplaySpeed	Speed in antiplay mode, full steps/s(stepper motor) or RPM. Range: 0..100000.
INT8U	uAntiplaySpeed	Speed in antiplay mode, 1/256 microsteps/s. Used with stepper motor only.
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set command setup movement (speed, acceleration, threshold and etc).

Command GMOV

```
rexult_t get_move_xettingx (device_t id, move_xettingx_t* move_xettingx)
```

Command code (CMD): "gmov" or 0x766F6D67.

Request: (4bytes)

INT32U	CMD	Command

Answer: (30bytes)

INT32U	CMD	Command (answer)
INT32U	Speed	Target speed (for stepper motor: steps/s, for DC: rpm). Range: 0..100000.
INT8U	uSpeed	Target speed in microstep fractions/s. Using with stepper motor only.
INT16U	Accel	Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 1..65535.
INT16U	Decel	Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 1..65535.
INT32U	AntiplaySpeed	Speed in antiplay mode, full steps/s(stepper motor) or RPM. Range: 0..100000.
INT8U	uAntiplaySpeed	Speed in antiplay mode, 1/256 microsteps/s. Used with stepper motor only.
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Description:
Read command setup movement (speed, acceleration, threshold and etc).

Command SENG

```
rexult_t xet_engine_xettingx (device_t id, conxt engine_xettingx_t* engine_xettingx)
```

Command code (CMD): "seng" or 0x676E6573.

Request: (34 bytes)

INT32U	CMD	Command
INT16U	NomVoltage	Rated voltage in tens of mV. Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set (used with DC only).
INT16U	NomCurrent	Rated current. Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 15..8000
INT32U	NomSpeed	Nominal speed (in whole steps/s or rpm for DC and stepper motor as a master encoder). Controller will keep motor shaft RPM below this value if ENGINE_LIMIT_RPM flag is set. Range: 1..100000.
INT8U	uNomSpeed	The fractional part of a nominal speed in microsteps (is only used with stepper motor).
INT16U	EngineFlags	Set of flags specify motor shaft movement algorithm and list of limitations
		0x01 - ENGINE_REVERSE (Reverse flag. It determines motor shaft rotation direction that corresponds to feedback counts increasing. If not set (default), motor shaft rotation direction under positive voltage corresponds to feedback counts increasing and vice versa. Change it if you see that positive directions on motor and feedback are opposite.)
		0x02 - ENGINE_CURRENT_AS_RMS (Engine current meaning flag. If the flag is set, then engine current value is interpreted as root mean square current value. If the flag is unset, then engine current value is interpreted as maximum amplitude value.)
		0x04 - ENGINE_MAX_SPEED (Max speed flag. If it is set, engine uses maximum speed achievable with the present engine settings as nominal speed.)
		0x08 - ENGINE_ANTIPLAY (Play compensation flag. If it set, engine makes backlash (play) compensation procedure and reach the predetermined position accurately on low speed.)
		0x10 - ENGINE_ACCEL_ON (Acceleration enable flag. If it set, motion begins with acceleration and ends with deceleration.)
		0x20 - ENGINE_LIMIT_VOLT (Maximum motor voltage limit enable flag(is only used with DC motor).)
		0x40 - ENGINE_LIMIT_CURR (Maximum motor current limit enable flag(is only used with DC motor).)
		0x80 - ENGINE_LIMIT_RPM (Maximum motor speed limit enable flag.)
INT16S	Antiplay	Number of pulses or steps for backlash (play) compensation procedure. Used if ENGINE_ANTIPLAY flag is set.
INT8U	MicrostepMode	Settings of microstep mode(Used with stepper motor only).
		0x01 - MICROSTEP_MODE_FULL (Full step mode.)
		0x02 - MICROSTEP_MODE_FRAC_2 (1/2 step mode.)
		0x03 - MICROSTEP_MODE_FRAC_4 (1/4 step mode.)
		0x04 - MICROSTEP_MODE_FRAC_8 (1/8 step mode.)
		0x05 - MICROSTEP_MODE_FRAC_16 (1/16 step mode.)
		0x06 - MICROSTEP_MODE_FRAC_32 (1/32 step mode.)
		0x07 - MICROSTEP_MODE_FRAC_64 (1/64 step mode.)
		0x08 - MICROSTEP_MODE_FRAC_128 (1/128 step mode.)
		0x09 - MICROSTEP_MODE_FRAC_256 (1/256 step mode.)
INT16U	StepsPerRev	Number of full steps per revolution(Used with stepper motor only). Range: 1..65535.
INT8U	Reserved [12]	Reserved (12 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set engine settings. This function send structure with set of engine settings to controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics. Use it when you change motor, encoder, positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

Command GENG

```
rexult_t get_engine_xettingx (device_t id, engine_xettingx_t* engine_xettingx)
```

Command code (CMD): "geng" or 0x676E6567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (34 bytes)

INT32U	CMD	Command (answer)
INT16U	NomVoltage	Rated voltage in tens of mV. Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set (used with DC only).
INT16U	NomCurrent	Rated current. Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 15..8000
INT32U	NomSpeed	Nominal speed (in whole steps/s or rpm for DC and stepper motor as a master encoder). Controller will keep motor shaft RPM below this value if ENGINE_LIMIT_RPM flag is set. Range: 1..100000.
INT8U	uNomSpeed	The fractional part of a nominal speed in microsteps (is only used with stepper motor).
INT16U	EngineFlags	Set of flags specify motor shaft movement algorithm and list of limitations
		0x01 - ENGINE_REVERSE (Reverse flag. It determines motor shaft rotation direction that corresponds to feedback counts increasing. If not set (default), motor shaft rotation direction under positive voltage corresponds to feedback counts increasing and vice versa. Change it if you see that positive directions on motor and feedback are opposite.)
		0x02 - ENGINE_CURRENT_AS_RMS (Engine current meaning flag. If the flag is set, then engine current value is interpreted as root mean square current value. If the flag is unset, then engine current value is interpreted as maximum amplitude value.)
		0x04 - ENGINE_MAX_SPEED (Max speed flag. If it is set, engine uses maximum speed achievable with the present engine settings as nominal speed.)
		0x08 - ENGINE_ANTIPLAY (Play compensation flag. If it set, engine makes backlash (play) compensation procedure and reach the predetermined position accurately on low speed.)
		0x10 - ENGINE_ACCEL_ON (Acceleration enable flag. If it set, motion begins with acceleration and ends with deceleration.)
		0x20 - ENGINE_LIMIT_VOLT (Maximum motor voltage limit enable flag(is only used with DC motor).)
		0x40 - ENGINE_LIMIT_CURR (Maximum motor current limit enable flag(is only used with DC motor).)
		0x80 - ENGINE_LIMIT_RPM (Maximum motor speed limit enable flag.)
INT16S	Antiplay	Number of pulses or steps for backlash (play) compensation procedure. Used if ENGINE_ANTIPLAY flag is set.
INT8U	MicrostepMode	Settings of microstep mode(Used with stepper motor only).
		0x01 - MICROSTEP_MODE_FULL (Full step mode.)
		0x02 - MICROSTEP_MODE_FRAC_2 (1/2 step mode.)

		0x03 - MICROSTEP_MODE_FRAC_4 (1/4 step mode.)
		0x04 - MICROSTEP_MODE_FRAC_8 (1/8 step mode.)
		0x05 - MICROSTEP_MODE_FRAC_16 (1/16 step mode.)
		0x06 - MICROSTEP_MODE_FRAC_32 (1/32 step mode.)
		0x07 - MICROSTEP_MODE_FRAC_64 (1/64 step mode.)
		0x08 - MICROSTEP_MODE_FRAC_128 (1/128 step mode.)
		0x09 - MICROSTEP_MODE_FRAC_256 (1/256 step mode.)
INT16U	StepsPerRev	Number of full steps per revolution(Used with stepper motor only). Range: 1..65535.
INT8U	Reserved [12]	Reserved (12 bytes)
INT16U	CRC	Checksum

Description:
Read engine settings. This function fill structure with set of useful motor settings stored in controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics.

Command SENT

```
rexult_t xet_entype_xettingx (device_t id, conxt entype_xettingx_t* entype_xettingx)
```

Command code (CMD): "sent" or 0x746E6573.

Request: (14 bytes)

INT32U	CMD	Command
INT8U	EngineType	Engine type
		0x00 - ENGINE_TYPE_NONE (A value that shouldn't be used.)
		0x01 - ENGINE_TYPE_DC (DC motor.)
		0x02 - ENGINE_TYPE_2DC (2 DC motors.)
		0x03 - ENGINE_TYPE_STEP (Step motor.)
		0x04 - ENGINE_TYPE_TEST (Duty cycle are fixed. Used only manufacturer.)
		0x05 - ENGINE_TYPE_BRUSHLESS (Brushless motor.)
INT8U	DriverType	Driver type
		0x01 - DRIVER_TYPE_DISCRETE_FET (Driver with discrete FET keys. Default option.)
		0x02 - DRIVER_TYPE_INTEGRATE (Driver with integrated IC.)
		0x03 - DRIVER_TYPE_EXTERNAL (External driver.)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set engine type and driver type.

Command GENT

```
rexult_t get_entype_xettingx (device_t id, entype_xettingx_t* entype_xettingx)
```

Command code (CMD): "gent" or 0x746E6567.

Request: (4bytes)

INT32U	CMD	Command

Answer: (14 bytes)

INT32U	CMD	Command (answer)
INT8U	EngineType	Engine type
		0x00 - ENGINE_TYPE_NONE (A value that shouldn't be used.)
		0x01 - ENGINE_TYPE_DC (DC motor.)
		0x02 - ENGINE_TYPE_2DC (2 DC motors.)
		0x03 - ENGINE_TYPE_STEP (Step motor.)
		0x04 - ENGINE_TYPE_TEST (Duty cycle are fixed. Used only manufacturer.)
		0x05 - ENGINE_TYPE_BRUSHLESS (Brushless motor.)
INT8U	DriverType	Driver type
		0x01 - DRIVER_TYPE_DISCRETE_FET (Driver with discrete FET keys. Default option.)
		0x02 - DRIVER_TYPE_INTEGRATE (Driver with integrated IC.)
		0x03 - DRIVER_TYPE_EXTERNAL (External driver.)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Return engine type and driver type.

Command SPWR

```
rexult_t xet_power_xettingx (device_t id, conxt power_xettingx_t* power_xettingx)
```

Command code (CMD): "spwr" or 0x72777073.

Request: (20 bytes)

INT32U	CMD	Command
INT8U	HoldCurrent	Current in holding regime, percent of nominal. Range: 0..100.
INT16U	CurrReductDelay	Time in ms from going to STOP state to reducting current.
INT16U	PowerOffDelay	Time in s from going to STOP state to turning power off.
INT16U	CurrentSetTime	Time in ms to reach nominal current.
INT8U	PowerFlags	Flags with parameters of power control.
		0x01 - POWER_REDUCT_ENABLED (Current reduction enabled after CurrReductDelay, if this flag is set.)
		0x02 - POWER_OFF_ENABLED (Power off enabled after PowerOffDelay, if this flag is set.)
		0x04 - POWER_SMOOTH_CURRENT (Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.)

INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of step motor power control. Used with stepper motor only.

Command GPWR

```
rexult_t get_power_xettingx (device_t id, power_xettingx_t* power_xettingx)
```

Command code (CMD): "gpwr" or 0x72777067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (20 bytes)

INT32U	CMD	Command (answer)
INT8U	HoldCurrent	Current in holding regime, percent of nominal. Range: 0..100.
INT16U	CurrReductDelay	Time in ms from going to STOP state to reducting current.
INT16U	PowerOffDelay	Time in s from going to STOP state to turning power off.
INT16U	CurrentSetTime	Time in ms to reach nominal current.
INT8U	PowerFlags	Flags with parameters of power control.
		0x01 - POWER_REDUCT_ENABLED (Current reduction enabled after CurrReductDelay, if this flag is set.)
		0x02 - POWER_OFF_ENABLED (Power off enabled after PowerOffDelay, if this flag is set.)
		0x04 - POWER_SMOOTH_CURRENT (Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Read settings of step motor power control. Used with stepper motor only.

Command SSEC

```
rexult_t xet_xecure_xettingx (device_t id, conxt xecure_xettingx_t* xecure_xettingx)
```

Command code (CMD): "ssec" or 0x63657373.

Request: (28 bytes)

INT32U	CMD	Command
INT16U	LowUpwrOff	Lower voltage limit to turn off the motor, tens of mV.
INT16U	CriticalIpwr	Maximum motor current which triggers ALARM state, in mA.
INT16U	CriticalUpwr	Maximum motor voltage which triggers ALARM state, tens of mV.
INT16U	CriticalT	Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.
INT16U	CriticalIusb	Maximum USB current which triggers ALARM state, in mA.
INT16U	CriticalUusb	Maximum USB voltage which triggers ALARM state, tens of mV.
INT16U	MinimumUusb	Minimum USB voltage which triggers ALARM state, tens of mV.
INT8U	Flags	Critical parameter flags.
		0x01 - ALARM_ON_DRIVER_OVERHEATING (If this flag is set enter Alarm state on driver overheat signal.)
		0x02 - LOW_UPWR_PROTECTION (If this flag is set turn off motor when voltage is lower than LowUpwrOff.)
		0x04 - H_BRIDGE_ALERT (If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.)
		0x08 - ALARM_ON_BORDERS_SWAP_MISSET (If this flag is set enter Alarm state on borders swap misset)
		0x10 - ALARM_FLAGS_STICKING (If this flag is set only a STOP command can turn all alarms to 0)
		0x20 - USB_BREAK_RECONNECT (If this flag is set USB brake reconnect module will be enable)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set protection settings.

Command GSEC

```
rexult_t get_xecure_xettingx (device_t id, xecure_xettingx_t* xecure_xettingx)
```

Command code (CMD): "gsec" or 0x63657367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (28 bytes)

INT32U	CMD	Command (answer)
INT16U	LowUpwrOff	Lower voltage limit to turn off the motor, tens of mV.
INT16U	CriticalIpwr	Maximum motor current which triggers ALARM state, in mA.
INT16U	CriticalUpwr	Maximum motor voltage which triggers ALARM state, tens of mV.
INT16U	CriticalT	Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.
INT16U	CriticalIusb	Maximum USB current which triggers ALARM state, in mA.
INT16U	CriticalUusb	Maximum USB voltage which triggers ALARM state, tens of mV.

INT16U	MinimumUusb	Minimum USB voltage which triggers ALARM state, tens of mV.
INT8U	Flags	Critical parameter flags.
		0x01 - ALARM_ON_DRIVER_OVERHEATING (If this flag is set enter Alarm state on driver overheat signal.)
		0x02 - LOW_UPWR_PROTECTION (If this flag is set turn off motor when voltage is lower than LowUpwrOff.)
		0x04 - H_BRIDGE_ALERT (If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.)
		0x08 - ALARM_ON_BORDERS_SWAP_MISSET (If this flag is set enter Alarm state on borders swap misset)
		0x10 - ALARM_FLAGS_STICKING (If this flag is set only a STOP command can turn all alarms to 0)
		0x20 - USB_BREAK_RECONNECT (If this flag is set USB brake reconnect module will be enable)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read protection settings.

Command SEDS

```
rexult_t xet_edgex_xettingx (device_t id, conxt edgex_xettingx_t* edgex_xettingx)
```

Command code (CMD): "seds" or 0x73646573.

Request: (26 bytes)

INT32U	CMD	Command
INT8U	BorderFlags	Border flags, specify types of borders and motor behaviour on borders.
		0x01 - BORDER_IS_ENCODER (Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.)
		0x02 - BORDER_STOP_LEFT (Motor should stop on left border.)
		0x04 - BORDER_STOP_RIGHT (Motor should stop on right border.)
		0x08 - BORDERS_SWAP_MISSET_DETECTION (Motor should stop on both borders. Need to save motor then wrong border settings is se)
INT8U	EnderFlags	Ender flags, specify electrical behaviour of limit switches like order and pulled positions.
		0x01 - ENDER_SWAP (First limit switch on the right side, if set; otherwise on the left side.)
		0x02 - ENDER_SW1_ACTIVE_LOW (1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.)
		0x04 - ENDER_SW2_ACTIVE_LOW (1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.)
INT32S	LeftBorder	Left border position, used if BORDER_IS_ENCODER flag is set.
INT16S	uLeftBorder	Left border position in 1/256 microsteps(used with stepper motor only). Range: -255..255.
INT32S	RightBorder	Right border position, used if BORDER_IS_ENCODER flag is set.
INT16S	uRightBorder	Right border position in 1/256 microsteps. Used with stepper motor only. Range: -255..255.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set border and limit switches settings.

Command GEDS

```
rexult_t get_edgex_xettingx (device_t id, edgex_xettingx_t* edgex_xettingx)
```

Command code (CMD): "geds" or 0x73646567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (26 bytes)

INT32U	CMD	Command (answer)
INT8U	BorderFlags	Border flags, specify types of borders and motor behaviour on borders.
		0x01 - BORDER_IS_ENCODER (Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.)
		0x02 - BORDER_STOP_LEFT (Motor should stop on left border.)
		0x04 - BORDER_STOP_RIGHT (Motor should stop on right border.)
		0x08 - BORDERS_SWAP_MISSET_DETECTION (Motor should stop on both borders. Need to save motor then wrong border settings is se)
INT8U	EnderFlags	Ender flags, specify electrical behaviour of limit switches like order and pulled positions.
		0x01 - ENDER_SWAP (First limit switch on the right side, if set; otherwise on the left side.)
		0x02 - ENDER_SW1_ACTIVE_LOW (1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.)
		0x04 - ENDER_SW2_ACTIVE_LOW (1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.)
INT32S	LeftBorder	Left border position, used if BORDER_IS_ENCODER flag is set.
INT16S	uLeftBorder	Left border position in 1/256 microsteps(used with stepper motor only). Range: -255..255.
INT32S	RightBorder	Right border position, used if BORDER_IS_ENCODER flag is set.
INT16S	uRightBorder	Right border position in 1/256 microsteps. Used with stepper motor only. Range: -255..255.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Read border and limit switches settings.

Command SPID

```
rexult_t xet_pid_xettingx (device_t id, conxt pid_xettingx_t* pid_xettingx)
```

Command code (CMD): "spid" or 0x64697073.

Request: (48 bytes)

INT32U	CMD	Command
INT16U	KpU	Proportional gain for voltage PID routine
INT16U	KiU	Integral gain for voltage PID routine
INT16U	KdU	Differential gain for voltage PID routine
FLT32	Kpf	Proportional gain for BLDC position PID routine
FLT32	Kif	Integral gain for BLDC position PID routine
FLT32	Kdf	Differential gain for BLDC position PID routine
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set PID settings. This function send structure with set of PID factors to controller's memory. These settings specify behaviour of PID routine for positioner. These factors are slightly different for different positioners. All boards are supplied with standard set of PID setting on controller's flash memory. Please use it for loading new PID settings when you change positioner. Please note that wrong PID settings lead to device malfunction.

Command GPID

```
rexult_t get_pid_xettingx (device_t id, pid_xettingx_t* pid_xettingx)
```

Command code (CMD): "gpid" or 0x64697067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (48 bytes)

INT32U	CMD	Command (answer)
INT16U	KpU	Proportional gain for voltage PID routine
INT16U	KiU	Integral gain for voltage PID routine
INT16U	KdU	Differential gain for voltage PID routine
FLT32	Kpf	Proportional gain for BLDC position PID routine
FLT32	Kif	Integral gain for BLDC position PID routine
FLT32	Kdf	Differential gain for BLDC position PID routine
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read PID settings. This function fill structure with set of motor PID settings stored in controller's memory. These settings specify behaviour of PID routine for positioner. These factors are slightly different for different positioners. All boards are supplied with standard set of PID setting on controller's flash memory.

Command SSNI

```
rexult_t xet_xync_in_xettingx (device_t id, conxt xync_in_xettingx_t* xync_in_xettingx)
```

Command code (CMD): "ssni" or 0x696E7373.

Request: (28 bytes)

INT32U	CMD	Command
INT8U	SyncInFlags	Input synchronization flags
		0x01 - SYNCIN_ENABLED (Synchronization in mode is enabled, if this flag is set.)
		0x02 - SYNCIN_INVERT (Trigger on falling edge if flag is set, on rising edge otherwise.)
		0x04 - SYNCIN_GOTOPOSITION (The engine is go to position specified in Position and uPosition, if this flag is set. And it is shift on the Position and uPosition, if this flag is unset)
INT16U	ClutterTime	Input synchronization pulse dead time (mks).
INT32S	Position	Desired position or shift (whole steps)
INT16S	uPosition	The fractional part of a position or shift in microsteps. Is used with stepper motor. Range: -255..255.
INT32U	Speed	Target speed (for stepper motor: steps/s, for DC: rpm). Range: 0..100000.
INT8U	uSpeed	Target speed in microsteps/s. Using with stepper motor only.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set input synchronization settings. This function send structure with set of input synchronization settings, that specify behaviour of input synchronization, to controller's memory. All boards are supplied with standard set of these settings.

Command GSNI

```
rexult_t get_xync_in_xettingx (device_t id, xync_in_xettingx_t* xync_in_xettingx)
```

Command code (CMD): "gsni" or 0x696E7367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (28 bytes)

INT32U	CMD	Command (answer)
INT8U	SyncInFlags	Input synchronization flags
		0x01 - SYNCIN_ENABLED (Synchronization in mode is enabled, if this flag is set.)
		0x02 - SYNCIN_INVERT (Trigger on falling edge if flag is set, on rising edge otherwise.)

		0x04 - SYNCIN_GOTOPOSITION (The engine is go to position specified in Position and uPosition, if this flag is set. And it is shift on the Position and uPosition, if this flag is unset)
INT16U	ClutterTime	Input synchronization pulse dead time (mks).
INT32S	Position	Desired position or shift (whole steps)
INT16S	uPosition	The fractional part of a position or shift in microsteps. Is used with stepper motor. Range: -255..255.
INT32U	Speed	Target speed (for stepper motor: steps/s, for DC: rpm). Range: 0..100000.
INT8U	uSpeed	Target speed in microsteps/s. Using with stepper motor only.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Read input synchronization settings. This function fill structure with set of input synchronization settings, modes, periods and flags, that specify behaviour of input synchronization. All boards are supplied with standard set of these settings.

Command SSNO

```
rexult_t xet_xync_out_xettingx (device_t id, conxt xync_out_xettingx_t* xync_out_xettingx)
```

Command code (CMD): "ssno" or 0x6F6E7373.

Request: (16 bytes)

INT32U	CMD	Command
INT8U	SyncOutFlags	Output synchronization flags
		0x01 - SYNCOUT_ENABLED (Synchronization out pin follows the synchronization logic, if set. It governed by SYNCOUT_STATE flag otherwise.)
		0x02 - SYNCOUT_STATE (When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.)
		0x04 - SYNCOUT_INVERT (Low level is active, if set, and high level is active otherwise.)
		0x08 - SYNCOUT_IN_STEPS (Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.)
		0x10 - SYNCOUT_ONSTART (Generate synchronization pulse when movement starts.)
		0x20 - SYNCOUT_ONSTOP (Generate synchronization pulse when movement stops.)
		0x40 - SYNCOUT_ONPERIOD (Generate synchronization pulse every SyncOutPeriod encoder pulses.)
INT16U	SyncOutPulseSteps	This value specifies duration of output pulse. It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set.
INT16U	SyncOutPeriod	This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.
INT32U	Accuracy	This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.
INT8U	uAccuracy	This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set output synchronization settings. This function send structure with set of output synchronization settings, that specify behaviour of output synchronization, to controller's memory. All boards are supplied with standard set of these settings.

Command GSNO

```
rexult_t get_xync_out_xettingx (device_t id, xync_out_xettingx_t* xync_out_xettingx)
```

Command code (CMD): "gsno" or 0x6F6E7367.

Request: (4bytes)

INT32U	CMD	Command

Answer: (16 bytes)

INT32U	CMD	Command (answer)
INT8U	SyncOutFlags	Output synchronization flags
		0x01 - SYNCOUT_ENABLED (Synchronization out pin follows the synchronization logic, if set. It governed by SYNCOUT_STATE flag otherwise.)
		0x02 - SYNCOUT_STATE (When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.)
		0x04 - SYNCOUT_INVERT (Low level is active, if set, and high level is active otherwise.)
		0x08 - SYNCOUT_IN_STEPS (Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.)
		0x10 - SYNCOUT_ONSTART (Generate synchronization pulse when movement starts.)
		0x20 - SYNCOUT_ONSTOP (Generate synchronization pulse when movement stops.)
		0x40 - SYNCOUT_ONPERIOD (Generate synchronization pulse every SyncOutPeriod encoder pulses.)
INT16U	SyncOutPulseSteps	This value specifies duration of output pulse. It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set.
INT16U	SyncOutPeriod	This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.
INT32U	Accuracy	This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.
INT8U	uAccuracy	This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).
INT16U	CRC	Checksum

Description:
Read output synchronization settings. This function fill structure with set of output synchronization settings, modes, periods and flags, that specify behaviour of output synchronization. All boards are supplied with standard set of these settings.

Command SEIO

```
rexult_t xet_extio_xettingx (device_t id, conxt extio_xettingx_t* extio_xettingx)
```

Command code (CMD): "seio" or 0x6F696573.

Request: (18 bytes)

INT32U	CMD	Command
INT8U	EXTIOSetupFlags	Configuration flags of the external I-O
		0x01 - EXTIO_SETUP_OUTPUT (EXTIO works as output if flag is set, works as input otherwise.)
		0x02 - EXTIO_SETUP_INVERT (Interpret EXTIO states and fronts inverted if flag is set. Falling front as input event and low logic level as active state.)
INT8U	EXTIOModeFlags	Flags mode settings external I-O
		0x0F - EXTIO_SETUP_MODE_IN_BITS (Bits of the behaviour selector when the signal on input goes to the active state.)
		0x00 - EXTIO_SETUP_MODE_IN_NOP (Do nothing.)
		0x01 - EXTIO_SETUP_MODE_IN_STOP (Issue STOP command, ceasing the engine movement.)
		0x02 - EXTIO_SETUP_MODE_IN_PWOF (Issue PWOF command, powering off all engine windings.)
		0x03 - EXTIO_SETUP_MODE_IN_MOVR (Issue MOVR command with last used settings.)
		0x04 - EXTIO_SETUP_MODE_IN_HOME (Issue HOME command.)
		0x05 - EXTIO_SETUP_MODE_IN_ALARM (Set Alarm when the signal goes to the active state.)
		0xF0 - EXTIO_SETUP_MODE_OUT_BITS (Bits of the output behaviour selection.)
		0x00 - EXTIO_SETUP_MODE_OUT_OFF (EXTIO pin always set in inactive state.)
		0x10 - EXTIO_SETUP_MODE_OUT_ON (EXTIO pin always set in active state.)
		0x20 - EXTIO_SETUP_MODE_OUT_MOVING (EXTIO pin stays active during moving state.)
		0x30 - EXTIO_SETUP_MODE_OUT_ALARM (EXTIO pin stays active during Alarm state.)
		0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON (EXTIO pin stays active when windings are powered.)
		0x50 - EXTIO_SETUP_MODE_OUT_MOTOR_FOUND (EXTIO pin stays active when motor is connected (first winding).)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set EXTIO settings. This function writes a structure with a set of EXTIO settings to controller's memory. By default input event are signalled through rising front and output states are signalled by high logic state.

Command GEIO

```
rexult_t get_extio_xettingx (device_t id, extio_xettingx_t* extio_xettingx)
```

Command code (CMD): "geio" or 0x6F696567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT8U	EXTIOSetupFlags	Configuration flags of the external I-O

		0x01 - EXTIO_SETUP_OUTPUT (EXTIO works as output if flag is set, works as input otherwise.)
		0x02 - EXTIO_SETUP_INVERT (Interpret EXTIO states and fronts inverted if flag is set. Falling front as input event and low logic level as active state.)
INT8U	EXTIOModeFlags	Flags mode settings external I-O
		0x0F - EXTIO_SETUP_MODE_IN_BITS (Bits of the behaviour selector when the signal on input goes to the active state.)
		0x00 - EXTIO_SETUP_MODE_IN_NOP (Do nothing.)
		0x01 - EXTIO_SETUP_MODE_IN_STOP (Issue STOP command, ceasing the engine movement.)
		0x02 - EXTIO_SETUP_MODE_IN_PWOF (Issue PWOF command, powering off all engine windings.)
		0x03 - EXTIO_SETUP_MODE_IN_MOVR (Issue MOVR command with last used settings.)
		0x04 - EXTIO_SETUP_MODE_IN_HOME (Issue HOME command.)
		0x05 - EXTIO_SETUP_MODE_IN_ALARM (Set Alarm when the signal goes to the active state.)
		0xF0 - EXTIO_SETUP_MODE_OUT_BITS (Bits of the output behaviour selection.)
		0x00 - EXTIO_SETUP_MODE_OUT_OFF (EXTIO pin always set in inactive state.)
		0x10 - EXTIO_SETUP_MODE_OUT_ON (EXTIO pin always set in active state.)
		0x20 - EXTIO_SETUP_MODE_OUT_MOVING (EXTIO pin stays active during moving state.)
		0x30 - EXTIO_SETUP_MODE_OUT_ALARM (EXTIO pin stays active during Alarm state.)
		0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON (EXTIO pin stays active when windings are powered.)
		0x50 - EXTIO_SETUP_MODE_OUT_MOTOR_FOUND (EXTIO pin stays active when motor is connected (first winding).)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Description:
Read EXTIO settings. This function reads a structure with a set of EXTIO settings from controller's memory.

Command SBRK

```
rexult_t xet_brake_xettingx (device_t id, conxt brake_xettingx_t* brake_xettingx)
```

Command code (CMD): "sbrk" or 0x6B726273.

Request: (25 bytes)

INT32U	CMD	Command
INT16U	t1	Time in ms between turn on motor power and turn off brake.
INT16U	t2	Time in ms between turn off brake and moving readiness. All moving commands will execute after this interval.
INT16U	t3	Time in ms between motor stop and turn on brake.
INT16U	t4	Time in ms between turn on brake and turn off motor power.
INT8U	BrakeFlags	Flags.
		0x01 - BRAKE_ENABLED (Brake control is enabled, if this flag is set.)

		0x02 - BRAKE_ENG_PWROFF (Brake turns off power of step motor, if this flag is set.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of brake control.

Command GBRK

```
rexult_t get_brake_xettingx (device_t id, brake_xettingx_t* brake_xettingx)
```

Command code (CMD): "gbrk" or 0x6B726267.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (25 bytes)

INT32U	CMD	Command (answer)
INT16U	t1	Time in ms between turn on motor power and turn off brake.
INT16U	t2	Time in ms between turn off brake and moving readiness. All moving commands will execute after this interval.
INT16U	t3	Time in ms between motor stop and turn on brake.
INT16U	t4	Time in ms between turn on brake and turn off motor power.
INT8U	BrakeFlags	Flags.
		0x01 - BRAKE_ENABLED (Brake control is enabled, if this flag is set.)
		0x02 - BRAKE_ENG_PWROFF (Brake turns off power of step motor, if this flag is set.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Description:
Read settings of brake control.

Command SCTL

```
rexult_t xet_control_xettingx (device_t id, conxt control_xettingx_t* control_xettingx)
```

Command code (CMD): "sctl" or 0x6C746373.

Request: (93 bytes)

INT32U	CMD	Command
INT32U	MaxSpeed [10]	Array of speeds (full step) using with joystick and button control. Range: 0..100000.
INT8U	uMaxSpeed [10]	Array of speeds (1/256 microstep) using with joystick and button control.
INT16U	Timeout [9]	timeout[i] is time in ms, after that max_speed[i+1] is applying. It is using with buttons control

		only.
INT16U	MaxClickTime	Maximum click time. Prior to the expiration of this time the first speed isn't enabled.
INT16U	Flags	Flags.
		0x03 - CONTROL_MODE_BITS (Bits to control engine by joystick or buttons.)
		0x00 - CONTROL_MODE_OFF (Control is disabled.)
		0x01 - CONTROL_MODE_JOY (Control by joystick.)
		0x02 - CONTROL_MODE_LR (Control by left/right buttons.)
		0x04 - CONTROL_BTN_LEFT_PUSHED_OPEN (Pushed left button corresponds to open contact, if this flag is set.)
		0x08 - CONTROL_BTN_RIGHT_PUSHED_OPEN (Pushed right button corresponds to open contact, if this flag is set.)
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps. Is only used with stepper motor. Range: -255..255.
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of motor control. When choosing CTL_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i +1] to acceleration, as usual.

Command GCTL

```
rexult_t get_control_xettingx (device_t id, control_xettingx_t* control_xettingx)
```

Command code (CMD): "gctl" or 0x6C746367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (93 bytes)

INT32U	CMD	Command (answer)
INT32U	MaxSpeed [10]	Array of speeds (full step) using with joystick and button control. Range: 0..100000.
INT8U	uMaxSpeed [10]	Array of speeds (1/256 microstep) using with joystick and button control.
INT16U	Timeout [9]	timeout[i] is time in ms, after that max_speed[i+1] is applying. It is using with buttons control only.
INT16U	MaxClickTime	Maximum click time. Prior to the expiration of this time the first speed isn't enabled.
INT16U	Flags	Flags.
		0x03 - CONTROL_MODE_BITS (Bits to control engine by joystick or buttons.)
		0x00 - CONTROL_MODE_OFF (Control is disabled.)
		0x01 - CONTROL_MODE_JOY (Control by joystick.)
		0x02 - CONTROL_MODE_LR (Control by left/right buttons.)

		0x04 - CONTROL_BTN_LEFT_PUSHED_OPEN (Pushed left button corresponds to open contact, if this flag is set.)
		0x08 - CONTROL_BTN_RIGHT_PUSHED_OPEN (Pushed right button corresponds to open contact, if this flag is set.)
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps. Is only used with stepper motor. Range: -255..255.
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Description:
Read settings of motor control. When choosing CTL_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i +1] to acceleration, as usual.

Command SJOY

```
rexult_t xet_joyxtick_xettingx (device_t id, conxt joyxtick_xettingx_t* joyxtick_xettingx)
```

Command code (CMD): "sjoy" or 0x796F6A73.

Request: (22 bytes)

INT32U	CMD	Command
INT16U	JoyLowEnd	Joystick lower end position. Range: 0..10000.
INT16U	JoyCenter	Joystick center position. Range: 0..10000.
INT16U	JoyHighEnd	Joystick higher end position. Range: 0..10000.
INT8U	ExpFactor	Exponential nonlinearity factor.
INT8U	DeadZone	Joystick dead zone.
INT8U	JoyFlags	Joystick control flags.
		0x01 - JOY_REVERSE (Joystick action is reversed. Joystick deviation to the upper values correspond to negative speeds and vice versa.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of joystick. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command

SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.
The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:

The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Command GJOY

```
rexult_t get_joyxtick_xettingx (device_t id, joyxtick_xettingx_t* joyxtick_xettingx)
```

Command code (CMD): "gjoy" or 0x796F6A67.

Request: (4bytes)

INT32U	CMD	Command

Answer: (22 bytes)

INT32U	CMD	Command (answer)
INT16U	JoyLowEnd	Joystick lower end position. Range: 0..10000.
INT16U	JoyCenter	Joystick center position. Range: 0..10000.
INT16U	JoyHighEnd	Joystick higher end position. Range: 0..10000.
INT8U	ExpFactor	Exponential nonlinearity factor.
INT8U	DeadZone	Joystick dead zone.
INT8U	JoyFlags	Joystick control flags.
		0x01 - JOY_REVERSE (Joystick action is reversed. Joystick deviation to the upper values correspond to negative speeds and vice versa.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read settings of joystick. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see

command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.
The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:

The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Command SCTP

```
rexult_t xet_ctp_xettingx (device_t id, conxt ctp_xettingx_t* ctp_xettingx)
```

Command code (CMD): "sctp" or 0x70746373.

Request: (18 bytes)

INT32U	CMD	Command
INT8U	CTPMinError	Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag. Measured in steps step motor.
INT8U	CTPFlags	Flags.
		0x01 - CTP_ENABLED (Position control is enabled, if flag set.)

		0x02 - CTP_BASE (Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.)
		0x04 - CTP_ALARM_ON_ERROR (Set ALARM on mismatch, if flag set.)
		0x08 - REV_SENS_INV (Sensor is active when it 0 and invert makes active level 1. That is, if you do not invert, it is normal logic - 0 is the activation.)
		0x10 - CTP_ERROR_CORRECTION (Correct errors which appear when slippage if the flag is set. It works only with the encoder. Incompatible with flag CTP_ALARM_ON_ERROR.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of control position(is only used with stepper motor). When controlling the step motor with encoder (CTP_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE_CTP_ERROR. When controlling the step motor with speed sensor (CTP_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE_CTP_ERROR.

Command GCTP

```
rexult_t get_ctp_xettingx (device_t id, ctp_xettingx_t* ctp_xettingx)
```

Command code (CMD): "gctp" or 0x70746367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT8U	CTPMinError	Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag. Measured in steps step motor.
INT8U	CTPFlags	Flags.
		0x01 - CTP_ENABLED (Position control is enabled, if flag set.)
		0x02 - CTP_BASE (Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.)
		0x04 - CTP_ALARM_ON_ERROR (Set ALARM on mismatch, if flag set.)
		0x08 - REV_SENS_INV (Sensor is active when it 0 and invert makes active level 1. That is, if you do not invert, it is normal logic - 0 is the activation.)
		0x10 - CTP_ERROR_CORRECTION (Correct errors which appear when slippage if the flag is set. It works only with the encoder. Incompatible with flag CTP_ALARM_ON_ERROR.)
INT8U	Reserved [10]	Reserved (10 bytes)

INT16U	CRC	Checksum

Description:
Read settings of control position(is only used with stepper motor). When controlling the step motor with encoder (CTP_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE_CTP_ERROR. When controlling the step motor with speed sensor (CTP_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE_CTP_ERROR.

Command SURT

```
rexult_t xet_uart_xettingx (device_t id, conxt uart_xettingx_t* uart_xettingx)
```

Command code (CMD): "surt" or 0x74727573.

Request: (16 bytes)

INT32U	CMD	Command
INT32U	Speed	UART speed
INT16U	UARTSetupFlags	UART setup flags
		0x03 - UART_PARITY_BITS (Bits of the parity.)
		0x00 - UART_PARITY_BIT_EVEN (Parity bit 1, if even)
		0x01 - UART_PARITY_BIT_ODD (Parity bit 1, if odd)
		0x02 - UART_PARITY_BIT_SPACE (Parity bit always 0)
		0x03 - UART_PARITY_BIT_MARK (Parity bit always 1)
		0x04 - UART_PARITY_BIT_USE (None parity)
		0x08 - UART_STOP_BIT (If set - one stop bit, else two stop bit)
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set UART settings. This function send structure with UART settings to controller's memory.

Command GURT

```
rexult_t get_uart_xettingx (device_t id, uart_xettingx_t* uart_xettingx)
```

Command code (CMD): "gurt" or 0x74727567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (16 bytes)

INT32U	CMD	Command (answer)

INT32U	Speed	UART speed
INT16U	UARTSetupFlags	UART setup flags
		0x03 - UART_PARITY_BITS (Bits of the parity.)
		0x00 - UART_PARITY_BIT_EVEN (Parity bit 1, if even)
		0x01 - UART_PARITY_BIT_ODD (Parity bit 1, if odd)
		0x02 - UART_PARITY_BIT_SPACE (Parity bit always 0)
		0x03 - UART_PARITY_BIT_MARK (Parity bit always 1)
		0x04 - UART_PARITY_BIT_USE (None parity)
		0x08 - UART_STOP_BIT (If set - one stop bit, else two stop bit)
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Description:
Read UART settings. This function fill structure with UART settings.

Command SCAL

```
rexult_t xet_calibration_xettingx (device_t id, conxt calibration_xettingx_t* calibration_xettingx)
```

Command code (CMD): "scal" or 0x6C616373.

Request: (118 bytes)

INT32U	CMD	Command
FLT32	CSS1_A	Scaling factor for the analogue measurements of the winding A current.
FLT32	CSS1_B	Shift factor for the analogue measurements of the winding A current.
FLT32	CSS2_A	Scaling factor for the analogue measurements of the winding B current.
FLT32	CSS2_B	Shift factor for the analogue measurements of the winding B current.
FLT32	FullCurrent_A	Scaling factor for the analogue measurements of the full current.
FLT32	FullCurrent_B	Shift factor for the analogue measurements of the full current.
INT8U	Reserved [88]	Reserved (88 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set calibration settings. This function send structure with calibration settings to controller's memory.

Command GCAL

```
rexult_t get_calibration_xettingx (device_t id, calibration_xettingx_t* calibration_xettingx)
```

Command code (CMD): "gcal" or 0x6C616367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (118 bytes)

INT32U	CMD	Command (answer)
FLT32	CSS1_A	Scaling factor for the analogue measurements of the winding A current.
FLT32	CSS1_B	Shift factor for the analogue measurements of the winding A current.
FLT32	CSS2_A	Scaling factor for the analogue measurements of the winding B current.
FLT32	CSS2_B	Shift factor for the analogue measurements of the winding B current.
FLT32	FullCurrent_A	Scaling factor for the analogue measurements of the full current.
FLT32	FullCurrent_B	Shift factor for the analogue measurements of the full current.
INT8U	Reserved [88]	Reserved (88 bytes)
INT16U	CRC	Checksum

Description:
Read calibration settings. This function fill structure with calibration settings.

Command SNMF

```
rexult_t xet_controller_name (device_t id, conxt controller_name_t* controller_name)
```

Command code (CMD): "snmf" or 0x666D6E73.

Request: (30 bytes)

INT32U	CMD	Command
CHAR	ControllerName [16]	User conroller name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	CtrlFlags	Internal controller settings.
		0x01 - EEPROM_PRECEDENCE (If the flag is set settings from external EEPROM override controller settings.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:

Write user controller name and flags of setting from FRAM.

Command GNMF

```
rexult_t get_controller_name (device_t id, controller_name_t* controller_name)
```

Command code (CMD): "gnmf" or 0x666D6E67.

Request: (4bytes)

INT32U	CMD	Command

Answer: (30 bytes)

INT32U	CMD	Command (answer)
CHAR	ControllerName [16]	User conroller name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	CtrlFlags	Internal controller settings.
		0x01 - EEPROM_PRECEDENCE (If the flag is set settings from external EEPROM override controller settings.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read user controller name and flags of setting from FRAM.

Command SNVM

```
rexult_t xet_nonvolatile_memory (device_t id, conxt nonvolatile_memory_t* nonvolatile_memory)
```

Command code (CMD): "snvm" or 0x6D766E73.

Request: (36 bytes)

INT32U	CMD	Command
INT32U	UserData [7]	User data. Can be set by user for his/her convinience. Each element of the array stores only 32 bits of user data. This is important on systems where an int type contains more than 4 bytes. For example that all amd64 systems.
INT8U	Reserved [2]	Reserved (2 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Write userdata into FRAM.

Command GNVM

```
rexult_t get_nonvolatile_memory (device_t id, nonvolatile_memory_t* nonvolatile_memory)
```

Command code (CMD): "gnvm" or 0x6D766E67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (36 bytes)

INT32U	CMD	Command (answer)
INT32U	UserData [7]	User data. Can be set by user for his/her convinience. Each element of the array stores only 32 bits of user data. This is important on systems where an int type contains more than 4 bytes. For example that all amd64 systems.
INT8U	Reserved [2]	Reserved (2 bytes)
INT16U	CRC	Checksum

Description:
Read userdata from FRAM.

Group of commands movement control

Command STOP

```
rexult_t command_xtop (device_t id)
```

Command code (CMD): "stop" or 0x706F7473.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Immediately stop the engine, the transition to the STOP, mode key BREAK (winding short-circuited), the regime "retention" is deactivated for DC motors, keeping current in the windings for stepper motors (with Power management settings).

Command ASIA

```
rexult_t command_add_xync_in_action (device_t id, conxt command_add_xync_in_action_t* the_command_add_xync_
in_action)
```

Command code (CMD): "asia" or 0x61697361.

Request: (22 bytes)

INT32U	CMD	Command
INT32S	Position	Desired position or shift (whole steps)
INT16S	uPosition	The fractional part of a position or shift in microsteps. Is only used with stepper motor. Range: -255..255.
INT32U	Time	Time for which you want to achieve the desired position in microseconds.
INT8U	Reserved [6]	Reserved (6 bytes)

INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
This command adds one element of the FIFO commands that are executed when input clock pulse. Each pulse synchronization or perform that action, which is described in SSNI, if the buffer is empty, or the oldest loaded into the buffer action to temporarily replace the speed and coordinate in SSNI. In the latter case this action is erased from the buffer. The number of remaining empty buffer elements can be found in the structure of GETS.

Command PWOF

```
rexult_t command_power_off (device_t id)
```

Command code (CMD): "pwof" or 0x666F7770.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Immediately power off motor regardless its state. Shouldn't be used during motion as the motor could be power on again automatically to continue movement. The command is designed for manual motor power off. When automatic power off after stop is required, use power management system.

Command MOVE

```
rexult_t command_move (device_t id, int Poxition, int uPoxition)
```

Command code (CMD): "move" or 0x65766F6D.

Request: (18 bytes)

INT32U	CMD	Command
INT32S	Position	Desired position (whole steps).
INT16S	uPosition	The fractional part of a position in microsteps. Is only used with stepper motor. Range: -255..255.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the Position, uPosition. For stepper motor uPosition sets the microstep, for DC motor this field is not used.

Command MOVR

```
rexult_t command_movr (device_t id, int DeltaPoxition, int uDeltaPoxition)
```

Command code (CMD): "movr" or 0x72766F6D.

Request: (18 bytes)

INT32U	CMD	Command
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps is only used with stepper motor. Range: -255..255.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields DeltaPosition, uDeltaPosition. For stepper motor uDeltaPosition sets the microstep, for DC motor this field is not used.

Command HOME

```
rexult_t command_home (device_t id)
```

Command code (CMD): "home" or 0x656D6F68.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
The positive direction is to the right. A value of zero reverses the direction of the direction of the flag, the set speed. Restriction imposed by the trailer, act the same, except that the limit switch contact does not stop. Limit the maximum speed, acceleration and deceleration function. 1) moves the motor according to the speed FastHome, uFastHome and flag HOME_DIR_FAST until limit switch, if the flag is set HOME_STOP_ENDS, until the signal from the input synchronization if the flag HOME_STOP_SYNC (as accurately as possible is important to catch the moment of operation limit switch) or until the signal is received from the speed sensor, if the flag HOME_STOP_REV_SN 2) then moves according to the speed SlowHome, uSlowHome and flag HOME_DIR_SLOW until signal from the clock input, if the flag HOME_MV_SEC. If the flag HOME_MV_SEC reset skip this paragraph. 3) then move the motor according to the speed FastHome, uFastHome and flag HOME_DIR_SLOW a distance HomeDelta, uHomeDelta. description of flags and variable see in description for commands GHOM/SHOM

Command LEFT

```
rexult_t command_left (device_t id)
```

Command code (CMD): "left" or 0x7466656C.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Start continous moving to the left.

Command RIGT

```
rexult_t command_right (device_t id)
```

Command code (CMD): "right" or 0x74676972.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Start continous moving to the right.

Command LOFT

```
rexult_t command_loft (device_t id)
```

Command code (CMD): "loft" or 0x74666F6C.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Upon receiving the command "loft" the engine is shifted from the current point to a distance GENG :: Antiplay, then move to the same point.

Command SSTP

```
rexult_t command_xxtp (device_t id)
```

Command code (CMD): "sstp" or 0x70747373.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Soft stop engine. The motor stops with deceleration speed.

Group of commands set the current position

Command GPOS

```
rexult_t get_poxition (device_t id, get_poxition_t* the_get_poxition)
```

Command code (CMD): "gpos" or 0x736F7067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (26 bytes)

INT32U	CMD	Command (answer)
INT32S	Position	The position of the whole steps in the engine
INT16S	uPosition	Microstep position is only used with stepper motors
INT64S	EncPosition	Encoder position.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Reads the value position in steps and micro for stepper motor and encoder steps all engines.

Command SPOS

```
rexult_t xet_poxition (device_t id, conxt xet_poxition_t* the_xet_poxition)
```

Command code (CMD): "spos" or 0x736F7073.

Request: (26 bytes)

INT32U	CMD	Command

INT32S	Position	The position of the whole steps in the engine
INT16S	uPosition	Microstep position is only used with stepper motors
INT64S	EncPosition	Encoder position.
INT8U	PosFlags	Flags
		0x01 - SETPOS_IGNORE_POSITION (Will not reload position in steps/microsteps if this flag is set.)
		0x02 - SETPOS_IGNORE_ENCODER (Will not reload encoder state if this flag is set.)
INT8U	Reserved [5]	Reserved (5 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Sets any position value in steps and micro for stepper motor and encoder steps of all engines. It means, that changing main indicator of position.

Command ZERO

```
rexult_t command_zero (device_t id)
```

Command code (CMD): "zero" or 0x6F72657A.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position. In the latter case, set the zero current position and the target position counted so that the absolute position of the destination is the same. That is, if we were at 400 and moved to 500, then the command Zero makes the current position of 0, and the position of the destination - 100. Does not change the mode of movement that is if the motion is carried, it continues, and if the engine is in the "hold", the type of retention remains.

Group of commands to save and load settings

Command SAVE

```
rexult_t command_xave_xettingx (device_t id)
```

Command code (CMD): "save" or 0x65766173.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

Command READ

```
rexult_t command_read_xettingx (device_t id)
```

Command code (CMD): "read" or 0x64616572.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Command SARS

```
rexult_t command_xave_robuxt_xettingx (device_t id)
```

Command code (CMD): "sars" or 0x73726173.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Save important settings (calibration coefficients and etc.) from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

Command RERS

```
rexult_t command_read_robuxt_xettingx (device_t id)
```

Command code (CMD): "rers" or 0x73726572.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Read important settings (calibration coefficients and etc.) from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Command EESV

```
rexult_t command_eexave_xettingx (device_t id)
```

Command code (CMD): "eesv" or 0x76736565.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Save settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn`t change without it mechanical reconstruction. Can be used by manufacturer only.

Command EERD

```
rexult_t command_eeread_xettingx (device_t id)
```

Command code (CMD): "eerd" or 0x64726565.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Read settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn`t change without it mechanical reconstruction.

Group of commands get the status of the controller

Command GETS

```
rexult_t get_xtatux (device_t id, xtatux_t* xtatux)
```

Command code (CMD): "gets" or 0x73746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (54 bytes)

INT32U	CMD	Command (answer)
INT8U	MoveSts	Move state.
		0x01 - MOVE_STATE_MOVING (This flag indicates that controller is trying to move the motor. Don't use this flag for waiting of completion of the movement command. Use MVCMD_RUNNING flag from the MvCmdSts field instead.)
		0x02 - MOVE_STATE_TARGET_SPEED (Target speed is reached, if flag set.)
		0x04 - MOVE_STATE_ANTIPLAY (Motor is playing compensation, if flag set.)
INT8U	MvCmdSts	Move command state.
		0x3F - MVCMD_NAME_BITS (Move command bit mask.)
		0x00 - MVCMD_UKNWN (Unknown command.)
		0x01 - MVCMD_MOVE (Command move.)
		0x02 - MVCMD_MOVR (Command movr.)
		0x03 - MVCMD_LEFT (Command left.)
		0x04 - MVCMD_RIGHT (Command rigt.)
		0x05 - MVCMD_STOP (Command stop.)
		0x06 - MVCMD_HOME (Command home.)
		0x07 - MVCMD_LOFT (Command loft.)
		0x08 - MVCMD_SSTP (Command soft stop.)
		0x40 - MVCMD_ERROR (Finish state (1 - move command have finished with an error, 0 - move command have finished correctly). This flags is actual when MVCMD_RUNNING signals movement finish.)
		0x80 - MVCMD_RUNNING (Move command state (0 - move command have finished, 1 - move command is being executed).)
INT8U	PWRSts	Power state of the stepper motor (used only with stepper motor).
		0x00 - PWR_STATE_UNKNOWN (Unknown state, should never happen.)
		0x01 - PWR_STATE_OFF (Motor windings are disconnected from the driver.)
		0x03 - PWR_STATE_NORM (Motor windings are powered by nominal current.)
		0x04 - PWR_STATE_REDUCT (Motor windings are powered by reduced current to lower power consumption.)
		0x05 - PWR_STATE_MAX (Motor windings are powered by maximum current driver can provide at this voltage.)
INT8U	EncSts	Encoder state.
		0x00 - ENC_STATE_ABSENT (Encoder is absent.)
		0x01 - ENC_STATE_UNKNOWN (Encoder state is unknown.)

		0x02 - ENC_STATE_MALFUNC (Encoder is connected and malfunctioning.)
		0x03 - ENC_STATE_REVERS (Encoder is connected and operational but counts in other direction.)
		0x04 - ENC_STATE_OK (Encoder is connected and working properly.)
INT8U	WindSts	Windings state.
		0x00 - WIND_A_STATE_ABSENT (Winding A is disconnected.)
		0x01 - WIND_A_STATE_UNKNOWN (Winding A state is unknown.)
		0x02 - WIND_A_STATE_MALFUNC (Winding A is short-circuited.)
		0x03 - WIND_A_STATE_OK (Winding A is connected and working properly.)
		0x00 - WIND_B_STATE_ABSENT (Winding B is disconnected.)
		0x10 - WIND_B_STATE_UNKNOWN (Winding B state is unknown.)
		0x20 - WIND_B_STATE_MALFUNC (Winding B is short-circuited.)
		0x30 - WIND_B_STATE_OK (Winding B is connected and working properly.)
INT32S	CurPosition	Current position.
INT16S	uCurPosition	Step motor shaft position in 1/256 microsteps. Used only with stepper motor.
INT64S	EncPosition	Current encoder position.
INT32S	CurSpeed	Motor shaft speed.
INT16S	uCurSpeed	Part of motor shaft speed in 1/256 microsteps. Used only with stepper motor.
INT16S	Ipwr	Engine current.
INT16S	Upwr	Power supply voltage, tens of mV.
INT16S	Iusb	USB current consumption.
INT16S	Uusb	USB voltage, tens of mV.
INT16S	CurT	Temperature in tenths of degrees C.
INT32U	Flags	Set of flags specify motor shaft movement algorithm and list of limitations.
		0x00003F - STATE_CONTR (Flags of controller states.)
		0x000001 - STATE_ERRC (Command error encountered.)
		0x000002 - STATE_ERRD (Data integrity error encountered.)
		0x000004 - STATE_ERRV (Value error encountered.)
		0x000010 - STATE_EEPROM_CONNECTED (EEPROM with settings is connected.)
		0x000020 - STATE_IS_HOMED (Calibration performed)
		0x73FFC0 - STATE_SECUR (Flags of security.)
		0x000040 - STATE_ALARM (Controller is in alarm state indicating that something dangerous had happened. Most commands are ignored in this state. To reset the flag a STOP command must be issued.)
		0x000080 - STATE_CTP_ERROR (Control position error(is only used with stepper motor).)
		0x000100 - STATE_POWER_OVERHEAT (Power driver overheat.)
		0x000200 - STATE_CONTROLLER_OVERHEAT (Controller overheat.)
		0x000400 - STATE_OVERLOAD_POWER_VOLTAGE (Power voltage exceeds safe limit.)
		0x000800 - STATE_OVERLOAD_POWER_CURRENT (Power current exceeds safe limit.)
		0x001000 - STATE_OVERLOAD_USB_VOLTAGE (USB voltage exceeds safe limit.)
		0x002000 - STATE_LOW_USB_VOLTAGE (USB voltage is insufficient for normal operation.)
		0x004000 - STATE_OVERLOAD_USB_CURRENT (USB current exceeds safe limit.)
		0x008000 - STATE_BORDERS_SWAP_MISSET (Engine stuck at the wrong edge.)
		0x010000 - STATE_LOW_POWER_VOLTAGE (Power voltage is lower than Low Voltage Protection limit)
		0x020000 - STATE_H_BRIDGE_FAULT (Signal from the driver that fault happened)

		0x0C0000 - STATE_CURRENT_MOTOR_BITS (Bits indicating the current operating motor on boards with multiple outputs for engine mounting.)
		0x000000 - STATE_CURRENT_MOTOR0 (Motor 0.)
		0x040000 - STATE_CURRENT_MOTOR1 (Motor 1.)
		0x080000 - STATE_CURRENT_MOTOR2 (Motor 2.)
		0x0C0000 - STATE_CURRENT_MOTOR3 (Motor 3.)
		0x100000 - STATE_WINDING_RES_MISMATCH (The difference between winding resistances is too large)
		0x200000 - STATE_ENCODER_FAULT (Signal from the encoder that fault happened)
		0x400000 - STATE_MOTOR_CURRENT_LIMIT (Current limit exceeded)
INT32U	GPIOFlags	Set of flags of gpio states
		0xFFFF - STATE_DIG_SIGNAL (Flags of digital signals.)
		0x0001 - STATE_RIGHT_EDGE (Engine stuck at the right edge.)
		0x0002 - STATE_LEFT_EDGE (Engine stuck at the left edge.)
		0x0004 - STATE_BUTTON_RIGHT (Button "right" state (1 if pressed).)
		0x0008 - STATE_BUTTON_LEFT (Button "left" state (1 if pressed).)
		0x0010 - STATE_GPIO_PINOUT (External GPIO works as Out, if flag set; otherwise works as In.)
		0x0020 - STATE_GPIO_LEVEL (State of external GPIO pin.)
		0x0040 - STATE_HALL_A (State of Hall_a pin.)
		0x0080 - STATE_HALL_B (State of Hall_b pin.)
		0x0100 - STATE_HALL_C (State of Hall_c pin.)
		0x0200 - STATE_BRAKE (State of Brake pin.)
		0x0400 - STATE_REV_SENSOR (State of Revolution sensor pin.)
		0x0800 - STATE_SYNC_INPUT (State of Sync input pin.)
		0x1000 - STATE_SYNC_OUTPUT (State of Sync output pin.)
		0x2000 - STATE_ENC_A (State of encoder A pin.)
		0x4000 - STATE_ENC_B (State of encoder B pin.)
INT8U	CmdBufFreeSpace	This field shows the amount of free cells buffer synchronization chain.
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Description:
Return device state. Useful function that fills structure with snapshot of controller state, including speed, position and boolean flags.

Command STMS

```
rexult_t command_xtart_meaxurementx (device_t id)
```

Command code (CMD): "stms" or 0x736D7473.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Start measurements and buffering of speed, following error.

Command GETM

```
rexult_t get_meaxurementx (device_t id, meaxurementx_t* meaxurementx)
```

Command code (CMD): "getm" or 0x6D746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (216 bytes)

INT32U	CMD	Command (answer)
INT32S	Speed [25]	Current speed.
INT32S	Error [25]	Current error.
INT32U	Length	Length of actual data in buffer.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
A command to read the data buffer to build a speed graph and a sequence error. Filling the buffer starts with the command "start_measurements". The buffer holds 25 points, the points are taken with a period of 1 ms. To create a robust system, read data every 20 ms, if the buffer is completely full, then it is recommended to repeat the readings every 5 ms until the buffer again becomes filled with 20 points.

Command GETC

```
rexult_t get_chart_data (device_t id, chart_data_t* chart_data)
```

Command code (CMD): "getc" or 0x63746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (38 bytes)

INT32U	CMD	Command (answer)
INT16S	WindingVoltageA	In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.
INT16S	WindingVoltageB	In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.
INT16S	WindingVoltageC	In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.
INT16S	WindingCurrentA	In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.
INT16S	WindingCurrentB	In the case step motor, the current in the coil B; brushless if the current in the second coil,

		and in the case of DC is not used.
INT16S	WindingCurrentC	In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.
INT16U	Pot	Analog input value in ten-thousandths. Range: 0..10000
INT16U	Joy	The joystick position in the ten-thousandths. Range: 0..10000
INT16S	DutyCycle	Duty cycle of PWM.
INT8U	Reserved [14]	Reserved (14 bytes)
INT16U	CRC	Checksum

Description:
Return device electrical parameters, useful for charts. Useful function that fill structure with snapshot of controller voltages and currents.

Command GETI

```
rexult_t get_device_information (device_t id, device_information_t* device_information)
```

Command code (CMD): "geti" or 0x69746567.

Request: (4bytes)

INT32U	CMD	Command

Answer: (36 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [4]	Manufacturer
CHAR	ManufacturerId [2]	Manufacturer id
CHAR	ProductDescription [8]	Product description
INT8U	Major	The major number of the hardware version.
INT8U	Minor	Minor number of the hardware version.
INT16U	Release	Number of edits this release of hardware.
INT8U	Reserved [12]	Reserved (12 bytes)
INT16U	CRC	Checksum

Description:
Return device information. It's available from the firmware and bootloader.

Command GSER

```
rexult_t get_xerial_number (device_t id, unxigned int* SerialNumber)
```

Command code (CMD): "gser" or 0x72657367.

Request: (4bytes)

INT32U	CMD	Command

Answer: (10 bytes)

INT32U	CMD	Command (answer)
INT32U	SerialNumber	Board serial number.
INT16U	CRC	Checksum

Description:
Read device serial number.

Controller error response types

ERRC

Answer: (4 bytes)
Code: "errc" or 0x63727265

INT32U	"errc"	Command error

Description:
Controller answers with "errc" if the command is either not recognized or cannot be processed and sets the correspoding bit in status data structure.

ERRD

Answer: (4 bytes)
Code: "errd" or 0x64727265

INT32U	"errd"	Data error

Description:
Controller answers with "errd" if the CRC of the data section computed by the controller doesn't match the received CRC field and sets the correspoding bit in status data structure.

ERRV

Answer: (4 bytes)
Code: "errv" or 0x76727265

INT32U	"errv"	Value error

Description:
Controller answers with "errv" if any of the values in the command are out of acceptable range and can not be applied. Inacceptable value is replaced by a rounded, truncated or default value. Controller also sets the correspoding bit in status data structure.

Example of communication via PC terminal

Commands examples

Command code (CMD): “home” or 0x656D6F68.

0x68 0x6F 0x6D 0x65
CMD

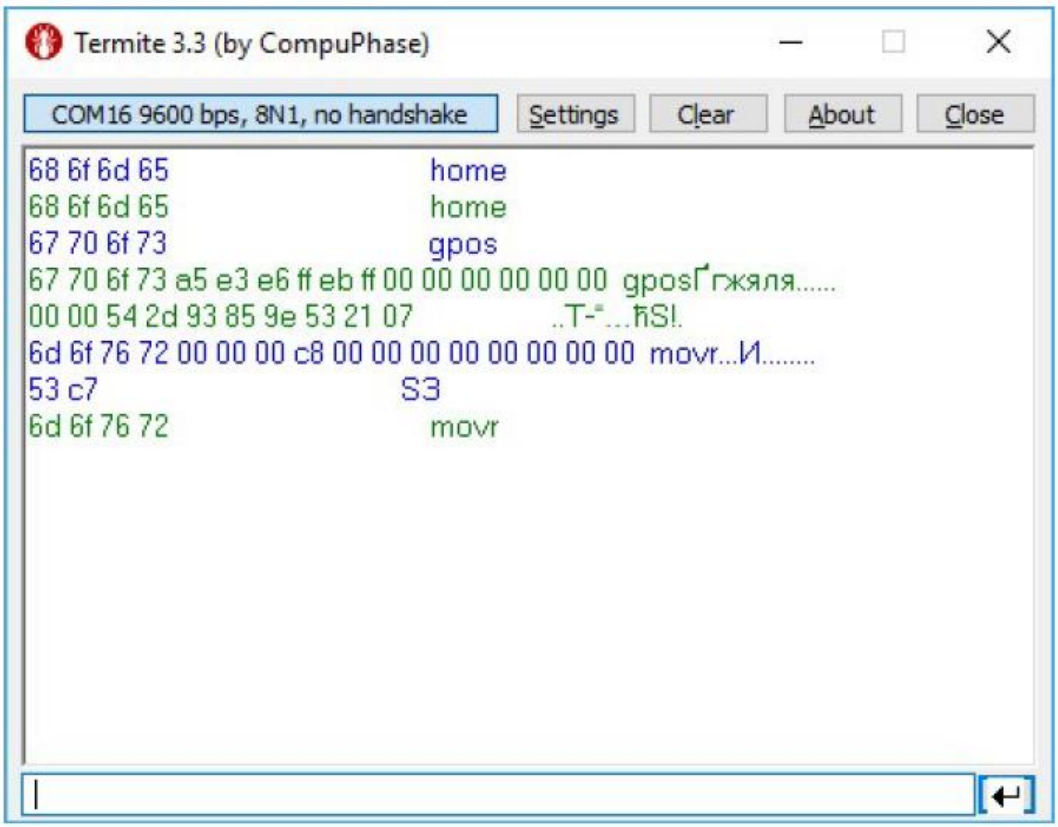
Command code (CMD): “gpos” or 0x736F7067

0x67 0x70 0x6F 0x73
CMD

Command code (CMD): “movr” or 0x72766F6D.

0x6D 0x6F 0x76 0x72	0xC8 0x00 0x00 0x00	0x00 0x00	0x00 0x00 0x00 0x00 0x00 0x00	0x53 0xc7
CMD	DeltaPosition*	uDPos	Reserved	CRC

* in this example “DeltaPosition” is 200 (0xC8).



Communication example using PC terminal software “Termite”

CRC calculation example in C

```c
#include <stdio.h>
#include <stdint.h>


unsigned short CRC16(uint8_t *pbuf, unsigned short n)
{
  unsigned short crc, i, j, carry_flag, a;
  crc = 0xffff;
  for(i = 0; i < n; i++)
  {
    crc = crc ^ pbuf[i];
    for(j = 0; j < 8; j++)
    {
      a = crc;
      carry_flag = a & 0x0001;
      crc = crc >> 1;
      if ( carry_flag == 1 ) crc = crc ^ 0xa001;
    }
  }
  return crc;
}

int main(void) {

  uint8_t Data[] = {0x00, 0x00, 0x00, 0xC8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

  uint16_t crc = CRC16(Data, 12);
  printf("%x %x", crc & 0xFF, (crc & 0xFF00) >> 8);
  return 0;
}
```