

Motor Controller MC-800
Art. No. 0890-0800

MANUAL



TABLE OF CONTENT

SAFETY INSTRUCTION	2
DESTINATED USE	2
DISCLAIMER	2
1 Product description	2
1.1 Features	2
1.2 Applications	2
1.3 General descriptions	2
1.4 Connection and operation	3
1.5 Delivery content	3
2 Installation	3
2.1 Mounting	3
2.2 Wiring	3
2.2.1 Overview	3
3 CAN Messages	4
3.1 Set Position	4
3.2 Set Velocity	4
3.3 Get Data	5
3.4 Status Position	5
3.5 Status Input	5
3.6 Set ID	5
4 Requirements for Operation	6
5 Operation with miCon-L	6
5.1 Connecting the mini-PLC	6
5.2 Using miCon-L	6
6 Programming Templates	8
6.1 Template 3.1 - SET POSITION	8
6.2 Template 3.2 - SET VELOCITY	10
6.3 Template 3.3 - GET DATA	11
6.4 Template 3.4 - STATUS POSITION	13
6.5 Template 3.5 - STATUS INPUT	15
6.6 Template 3.6 - SET ID	16
7 Appendix	19
7.1 Specifications	19
7.1.1 General	19
7.1.2 Power Supply	19
7.1.3 Interfaces	19
7.1.4 Security Features	19
7.1.5 Electrical Connection	19
7.1.6 Electromagnetic Compatability (EMC)	19
7.1.7 Environmental Conditions	19
7.1.8 Weight and Dimensions	19
7.1.9 Ordering Information	19
7.2 Disposal	20
7.3 Conformity Declaration	20

SAFETY INSTRUCTIONS

This manual contains safety instruction that should be followed to ensure your own personal safety, as well as the product's, and any connected equipment. These instructions are highlighted in the manual by a warning sign and are marked as follows according to the level of danger:



Only qualified personnel should be allowed to install and work on this equipment. Qualified is defined as a person who is authorized to commission, to ground and to tag circuits, equipment and systems in accordance with established safety practices and standards.



Turn off the power supply before performing any wiring operations! Short circuits can be harmful, critical and can cause explosions and serious burns!



Please read this manual carefully and observe all safety instructions!

DESTINATED USE

The MC-800 is designed as a stepper motor controller for universal motion & control applications. It must not be used for life critical, medical, or fail safe applications.

DISCLAIMER

BARTH Elektronik GmbH assumes no liability for usage and functionality of the MC-800 in case of disregarding this manual. The strict accordance to this manual is important since the installation methods, peripheral connections, usage and maintenance can not be controlled by BARTH Elektronik GmbH. Therefore BARTH Elektronik GmbH assumes no liability for any claim.

1 Product description

The picture below shows the motor controller MC-800 lococube[®] (Art. No. 0890-0800).



1.1 Features

- Ultracompact CAN motor controller up to 2.8A
- Parameter communication via CAN
- Dynamic motor current adjustment via CAN
- Speed and ramps freely configurable
- Step and stop modes dynamically adjustable
- Open-loop or closed-loop operation
- Inputs for encoder and limit switches
- Wide operation voltage range from 7 to 32 VDC
- Engineered and manufactured in Germany

1.2 Applications

- Universal motion & control
- Robotics and CNC systems
- Industrial process control
- X-by wire systems
- Technical education / University

1.3 General description

The MC-800 is a highly integrated CAN motor controller to directly drive one bipolar stepper motor up to 2.8 A. It has never been easier to realize simple Motion & Control or Pick & Place applications without the need of programming.

The integrated CAN 2.0A/B interface provides easiest Plug & Play interfacing to a lococube[®] mini-PLC without the need to program the MC-800. All motor parameters are communicated via CAN bus between mini-PLC and MC-800.

These outstanding features open up a variety of application fields in industrial, automotive and 12/24V battery-powered applications. Also robotic systems can be easily served using the powerful and easy-to-use MC-800.

The Motor Controller is also available as customer-tailored OEM version within 8 weeks.

1.4 Connection and operation

The motor controller MC-800 can be directly connected to any BARTH® mini-PLC via the CAN interface.

In case you use „miCon-L“ as your graphical programming tool, you can use the templates that come pre-installed in the firmware of the device. They can easily

Please see section 5 on how to get started quickly with the free miCon-L software samples we provide. They support you in sending the correct CAN-messages if you use one of your PLCs.

Please see section 3 for a list of the CAN-commands the MC-800 will accept. These can be useful, if you want to use your own CAN-hardware.

1.5 Delivery content

- Motor controller MC-800 lococube®
- 1x Connector for supply and CAN
- 1x Connector for for I/O and stepper motor

2 Installation

2.1 Mounting



The MC-800 must be installed and wired by a trained technician who knows and complies with both the unisversally applicable engineering rules and the standards that apply to specific cases.

Fastening the MC-800 follows using either the integrated mounting holes for screws or the holes for cable ties. The cable tie installation method is recommended for fastening the MC-800 on wiring harness, tubes or other mechanical parts.



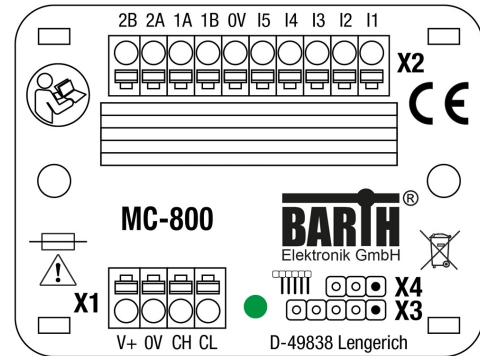
The heatsink of the MC-800 may reach temperatures up to 100°C. Take care not to cover the the heatsink to ensure proper heat dissipation and to protect your peripheal components against damage due to heat!



The MC-800 is intended to be mounted in enclosed cabinets (indoor use), and similar environments, that afford protection against fure hazards, such as environmental condiditions and mechanical impact. Do not operate outside the specified environmental conditions and ensure proper heat dissipation of the heatsink.

2.2 Wiring

2.2.1 Overview



X1: Power supply and CAN connector:

1	VDD	positive supply (+7 to 32 VDC)
2	GND	ground terminal (GND)
3	CANH	CAN high terminal
4	CANL	CAN low terminal

X2: Motor, input and output connector

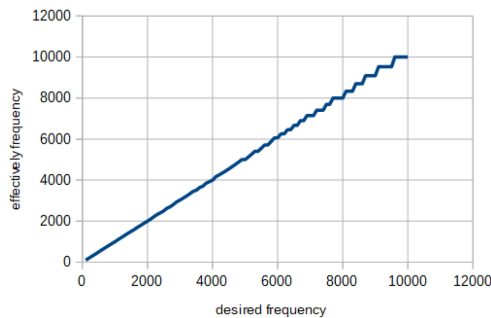
1	I1	analog input AIN1
2	I2	analog input AIN2
3	I3	analog input AIN3
4	I4	digital input DIN4 / encoder A
5	I5	digital input DIN5 / ecnoder (GND)
6	0V	analog / encoder (GND)
7	1B	bipolar stepper coil 1B
8	1A	bipolar stepper coil 1A
9	2A	bipolar stepper coil 2A
10	2B	biplar stepper coil 2B



Both X3 and X4 connectors are reserved for factory programming only! Connecting these terminals may cause irreversible damage to the MC-800!

3 CAN Messages

There is no programming necessary to operate the MC-800. The MC-800 will be parameterized using a common 2-wired CAN2.0A/B interface using a fixed baud rate of 250 kBit/s. The MC-800 uses 6 sequenced CAN-ID's. Default CAN-ID's are 0x200 - 0x205, the base CAN-ID is 0x200. The MC-800 receives all CAN-messages types with 11 or 28 Bit identifier. Messages from 0x000 to 0x7FF will be sent with an 11 Bit identifier, all other messages with a 29 Bit identifier.



At high frequencies, the effective frequency will get increasingly inaccurate.

Let's take a closer look at the six CAN-messages the MC-800 will accept.

3.1 Set Position

Format

Default ID: 0x201

DLC: 8 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bit0: Enable Bit2+3: Stepping (0: Full, 1: Half, 2: Quarter, 3: Eighth)	Current 0.1 - 2.8 A (Value x 100)	Moving steps -32768 - 32767		Minimal stepper frequency 1 - 10000 Hz		Maximal stepper frequency 1 - 10000 Hz	

3.2 Set Velocity

Format

Default ID: 0x205

DLC: 8 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bit0: Enable Bit2+3: Stepping (0: Full, 1: Half, 2: Quarter, 3: Eighth)	Current 0.1 - 2.8 A (Value x 100)	Moving direction -10: reverse -1: ramp down to stop 0: immediately stop 1: ramp down to stop 10: forward		Minimal stepper frequency 1 - 10000 Hz		Maximal stepper frequency 1 - 10000 Hz	

Common Stepper Motor Pin Layouts

Delta

Pin 1	Black	A
Pin 2	Yellow	COM A
Pin 3	Green	A/
Pin 4	Red	B
Pin 5	White	COM B
Pin 6	Blue	B/

Igus

Pin 1	White	A
Pin 2	Brown	A/
Pin 3	Blue	B
Pin 4	Black	B/

Nanotech

Pin 1	Black	A
Pin 3	Green	A/
Pin 4	Red	B/
Pin 6	Blue	B

3.3 Get Data

Format

Default ID: 0x202
 DLC: 1 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bit0: Request input status Bit1: Request position							

3.4 Status Position

Format

Default ID: 0x204
 DLC: 4 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Remaining step		Encoder position		Bit0: DIN4 Bit1: DIN5			

3.5 Status Input

Format

Default ID: 0x205
 DLC: 7 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
AIN1 [mV]		AIN2 [mV]		AIN3 [mV]		Bit0: DIN4 Bit1: DIN5	

3.6 Set ID

Format

Default ID: 0x200
 DLC: 4 Byte

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
New base CAN-ID							

4 Requirements for Operation

To operate the MC-800 motor controller you will need the following items:

1. The lococube[®] motor controller MC-800 itself.
2. A mini-PLC with a CAN interface (the lococube[®] STG-800 is a good choice to get started and will be used in all our upcoming examples: [lococube[®] mini-SPS STG-800](#))
3. A VK-16 cable is needed to connect the mini-PLC to your computer:
[Verbindungskabel VK-16](#)
4. A cable to connect the CAN-Bus of the mini-PLC to the one of the MC-800. A KS-85 wiring harness, which has those connections already included can be found here:
[Wiring harness KS-85](#)
5. A resistor of 60Ω needs to be placed between the CANH and the CANL connector of the DMA-15 and the mini-PLC. (You can omit this step if you're using a KS-85 wiring harness since it already includes those resistances).
6. A power adapter to power the mini-PLC and the MC-800 that supplies a voltage between 7 - 32 VCD. (If you're using a KS-85 wiring harness you can plug in a hollow plug with these dimensions: Ø 5,5 mm / 2,1 mm).
7. A computer with a Windows operating system installed.
8. The miCon-L software which can be downloaded here: [miCon-L](#)
9. The template programs which can be downloaded here: [Template Programs](#)

5 Operation with miCon-L

5.1 Connecting the mini-PLC

Connect the mini-PLC to your computer. We'll be using the lococube[®] STG-800. Plug the USB connector of the VK-16 cable into your computer and plug the other end into the mini-PLC into the TTL232 slot.

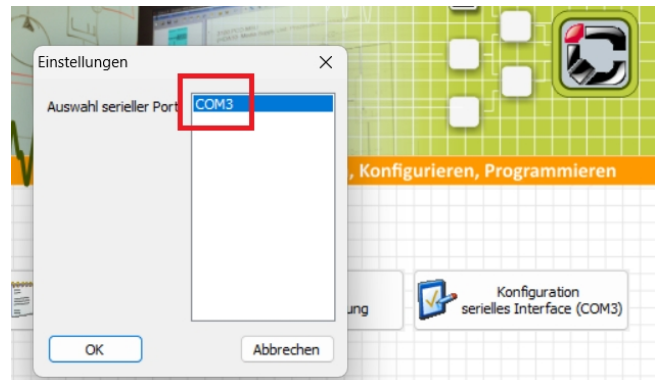
The orientation of the connector matters! Please orient the connector, with the little bump, facing **left**, as shown below.



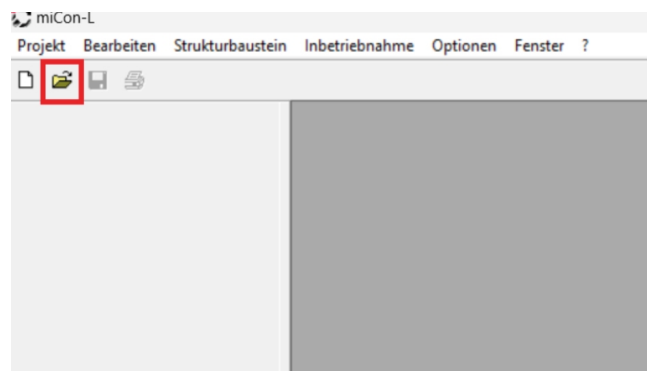
Connect the CANH terminal of the mini-PLC to the CANH terminal of the MC-800 and connect the CANL terminal of the mini-PLC to the CANL terminal of the MC-800. If you are using our KS-85 wiring harness, this can be done simply by plugging the X1 connector into the mini-PLC and the X2 connector into the MC-800 or vice versa. If you are not using a KS-85 wiring harness, make sure to put a resistor of 60Ω between the CAN terminals! Plug in the power adapter and power the mini-PLC and the MC-800.

5.2 Using miCon-L

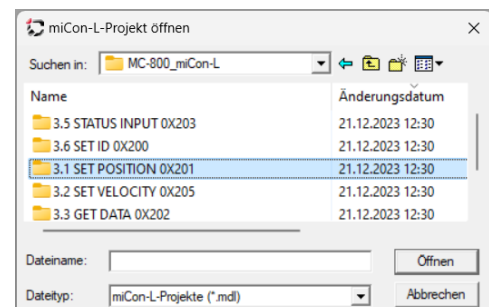
Download and install the miCon-L software on your computer. Download the template programs for the here and place them at a location of your choosing: After the STG-800 and the MC-800 have been properly wired up and connected, start the miCon-L software.



It's vital that you choose the correct serial port before you continue into the actual programming environment!

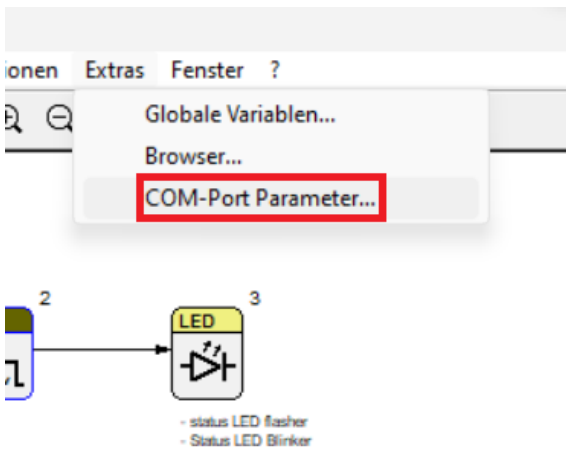


Now use the „folder icon“ in the top left hand corner to navigate to the folder you have placed the template projects in.

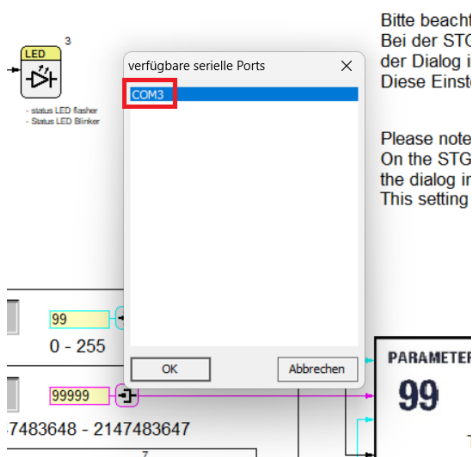


Navigate to the location you chose to put the template folder in and open one of the projects using the .MDL file.

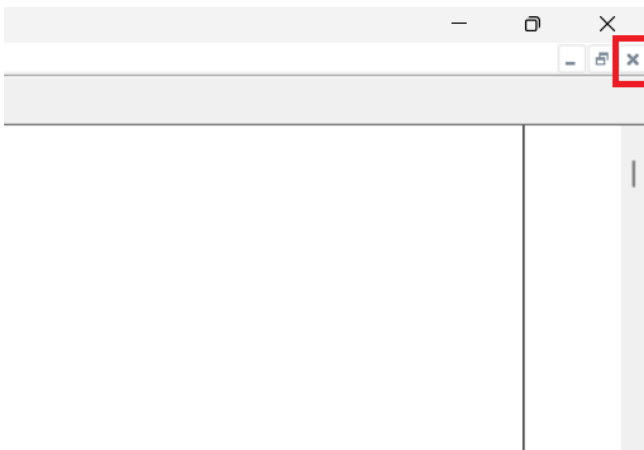
After you have successfully opened the project of your choice, there is one additional step that you must always do when creating or opening a new project! Go to Extras -> COM-Port Parameter.



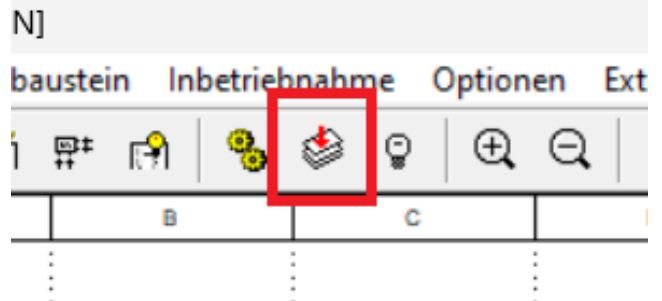
Choose the correct COM-Port again and click „OK“.



Before you can edit the program you will need to close the currently active window first. Press the ‚X‘ in the upper right hand corner to close it.

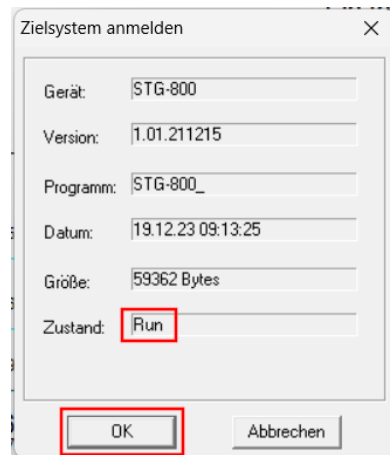


Next, we need to download the program onto the STG-800.

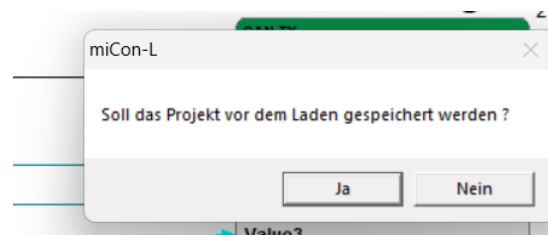


Now you will see a status window pop up that has various kinds of information on it, including what type of BARTH[®] mini-PLC you're using, version, date, etc.

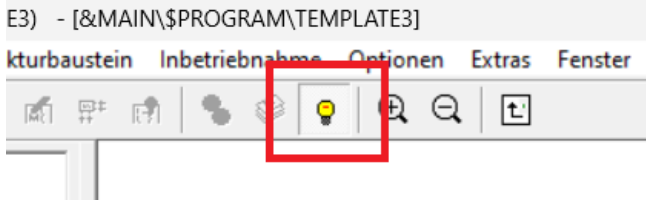
If the status bar at the bottom finished and the the status says „Run“ at the bottom, hit „OK“.



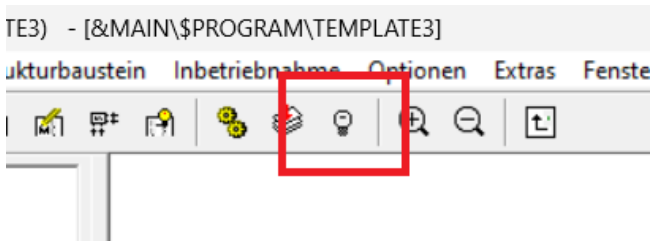
If you run the program for the first time or made any changes to it and have not saved it yet, miCon-L will ask you if you want to save the program before running. Click on „Ja“ if you want to save it or „Nein“ if you don't.



The program will automatically be set online now, indicated by the little yellow light bulb icon at the top, among the now greyed out icons.



If you want to stop the program and take it offline just click on that light bulb.
If you have not made any changes to the program and just want to run it again, you can click on the light bulb to put it online again.



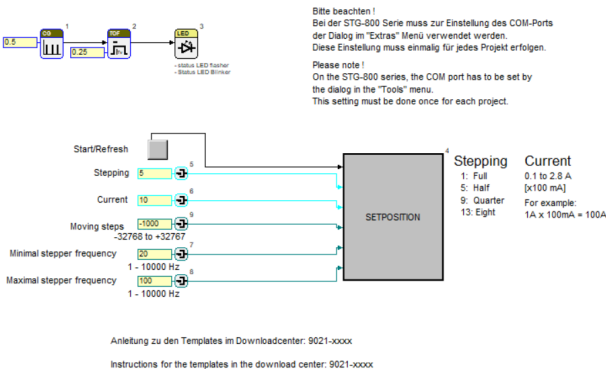
You only have to use the download button again if you made any changes to the program.

6 Programming Templates

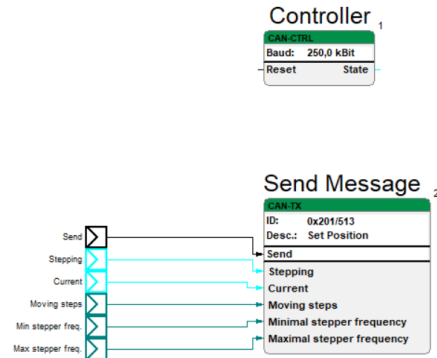
Now, we will take a closer look at the provided programing templates.

6.1 Template 3.1 - SET POSITION

Here we can see an overview of the main program.

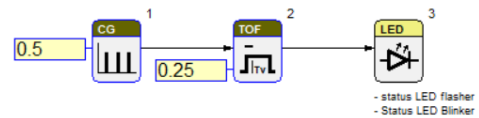


And the macro block „SETPOSITION“.

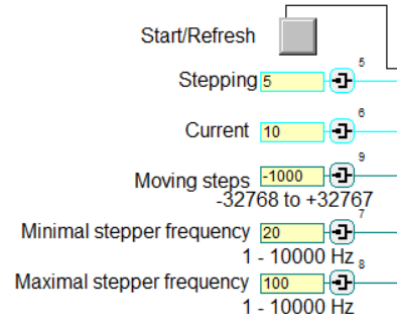


The program sets all the inputs and parameters, and the macro block is what actually controls what the buttons and those parameters do.

Let's take a closer look the the **program** first.

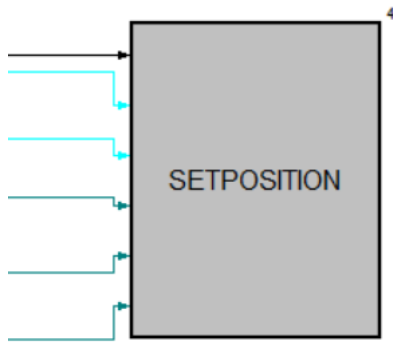


This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.



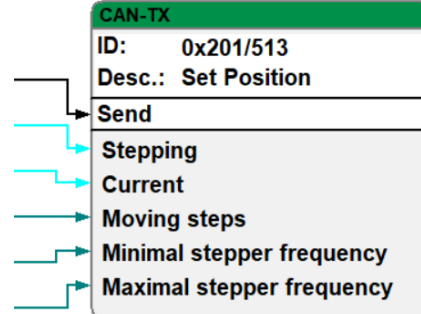
Here we have the inputs, that are used to set the different parameters of the MC-800.

1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "Stepping" decides if the spool in the motor does a full, half, quarter, or eighth turn with each instruction. There's a legend in the program, which shows the numbers you can enter and what they do.
3. The input, "Current" set the amperage, the motor will receive. The MC-800 supports a current of 0.1 to 2.8 A. There's a legend in the program that shows the number range you can enter, and how to calculate the actual amperage.
4. The input "Moving Steps" sets the distance, the motor shaft will turn, with each instruction.
5. The input "Minimal stepper frequency" sets the smallest step frequency allowed.
6. The input "Maximal stepper frequency" sets the biggest step frequency allowed.



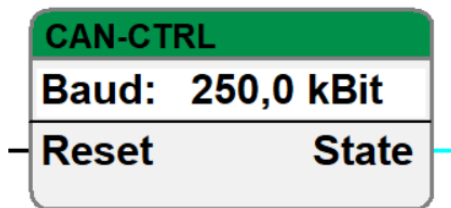
Here, we can all the inputs being sent into the macro block, which we will take a closer look now.

Send Message ²

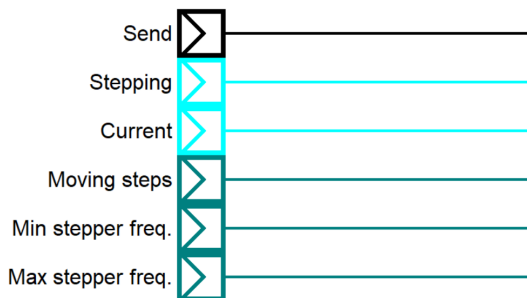


The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."

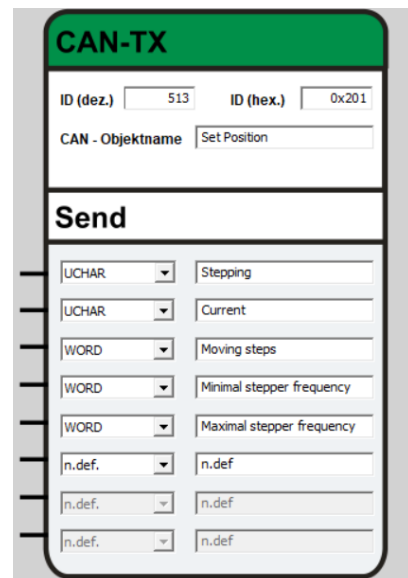
Controller ¹



Here we have the CAN controller that always has to be included, when we want to send or receive CAN messages. We use it to set the baud rate, which needs to be the same between the sending and receiving function blocks.



These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.

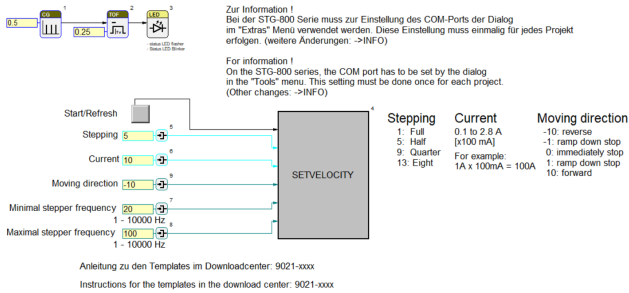


This is the parameter view of the function block.

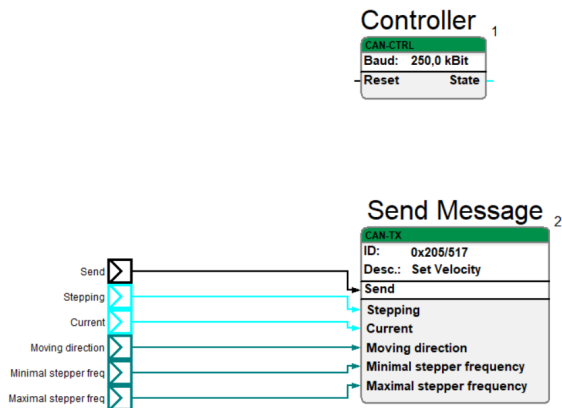
1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name.

6.2 TEMPLATE 3.2 - SET VELOCITY

Here we can see an overview of the main program.

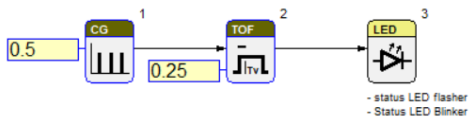


And the macro block „SETVELOCITY“.

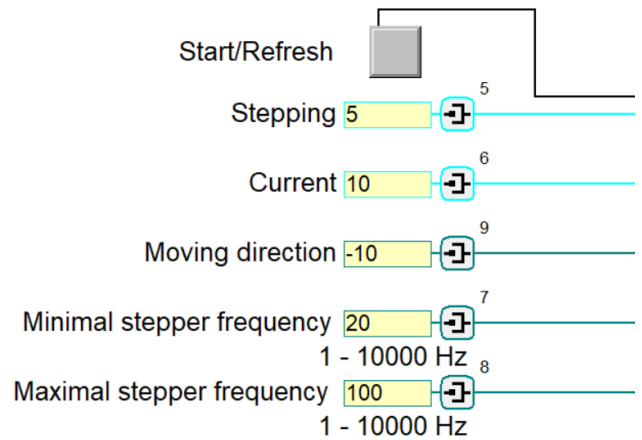


The program sets all the inputs and parameters, and the macro block is what actually controls what the buttons and those parameters do.

Let's take a closer look the the **program** first.

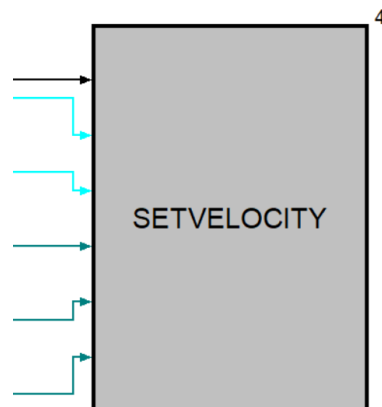


This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.

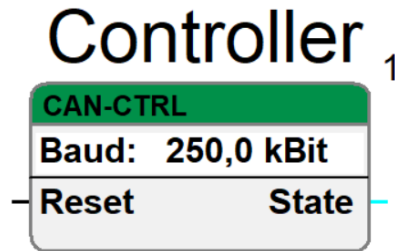


Here we have the inputs that are used to set the different parameters of the MC-800.

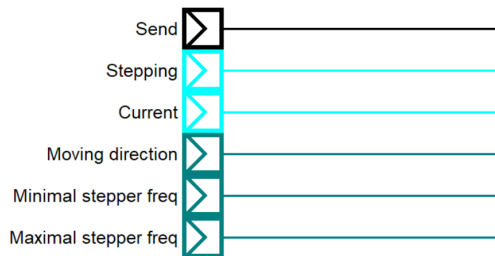
1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "Stepping" decides if the spool in the motor does a full, half, quarter, or eighth turn with each instruction. There's a legend in the program, which explains what numbers you can enter.
3. The input, "Current" set the amperage, the motor will receive. The MC-800 supports a current of 0.1 to 2.8A. There's a legend in the program that shows the numbers you can enter and what they do.
4. The input "Moving direction" decides in which direction the motor shaft will turn, and if it either ramps down or immediately stops. There's a legend in in the program that explains what numbers you can enter and what they do.
5. The input "Minimal stepper frequency" sets the smallest step frequency allowed.
6. The input "Maximal stepper frequency" sets the biggest step frequency allowed.



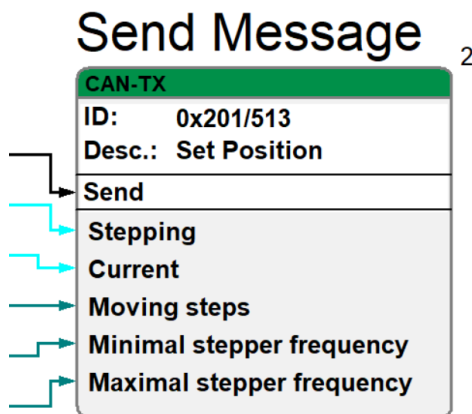
Here, we can all the inputs being sent into the macro block, which we will take a closer look now.



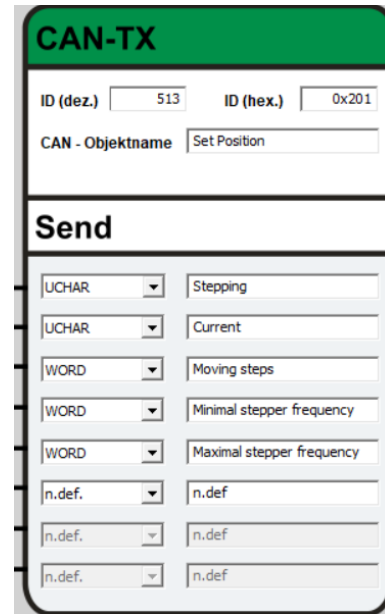
Here we have the CAN controller that always has to be included, when we want to send or receive CAN messages. We use it to set the baud rate, which needs to be the same between the sending and receiving function blocks.



These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.



The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message, once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."

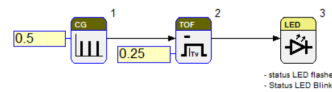


This is the parameter view of the function block.

1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name.

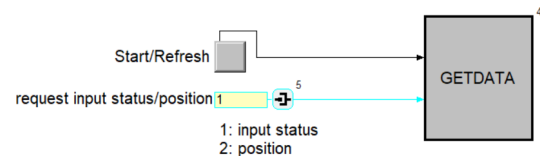
6.3 TEMPLATE 3.3 - GET DATA

Here we can see an overview of the main program.



Zur Information !
Bei der STG-800 Serie im "Extras" Menü verwerfen erfolgen. (weitere Änderungen)

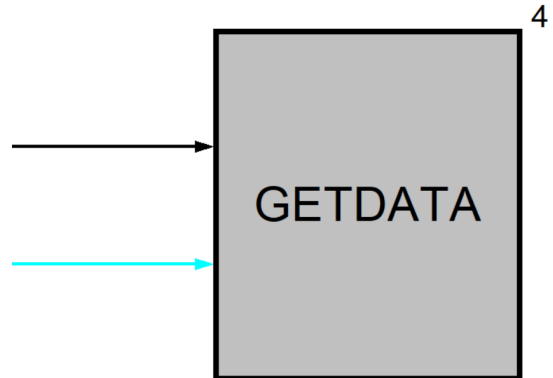
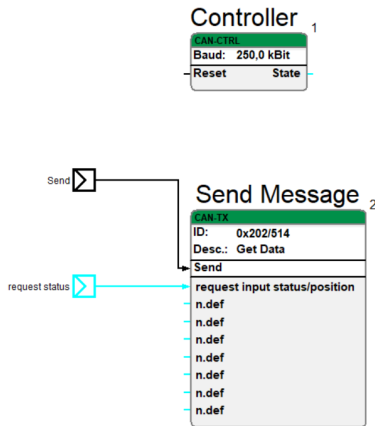
For information !
On the STG-800 series, in the "Tools" menu. This (Other changes: ->INFO)



Anleitung zu den Templates im Downloadcenter: 9021-xxxx

Instructions for the templates in the download center: 9021-xxxx

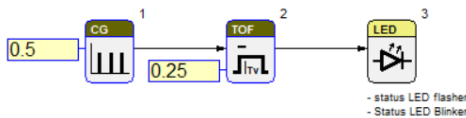
And the **macro block** „GETDATA“.



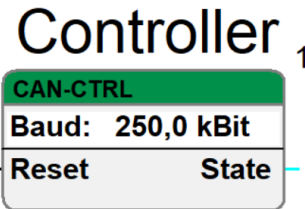
Here, we can all the inputs being sent into the macro block, which we will take a closer look now.

The program sets all the inputs and parameters, and the macro block is what actually controls what the buttons and those parameters do.

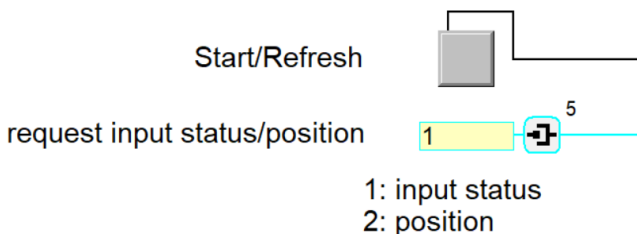
Let's take a closer look the the program first.



This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.



Here we have the CAN controller that always has to be included, when we want to send or receive CAN messages. We use it to set the baud rate, which needs to be the same between the sending and receiving function blocks.

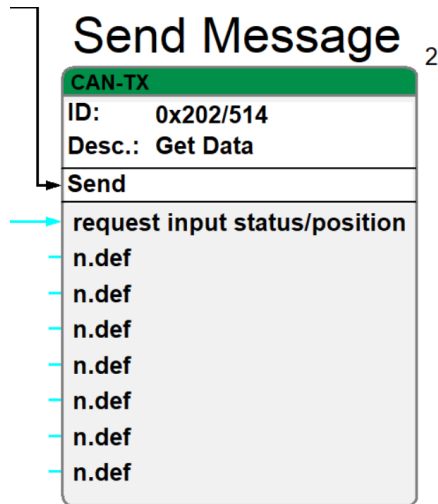


Here we have the inputs that are used to set the different parameters of the MC-800.

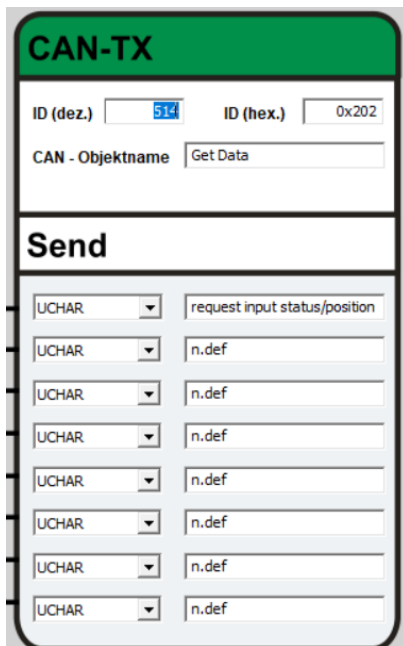
1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "request input status/position" sends a request for either the input status or the position of the the motor shaft.



These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.



The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message, once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."

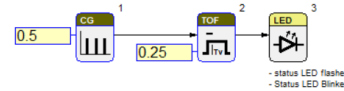


This is the parameter view of the function block.

1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name. The not defined inputs that have a datatype assigned to them, need to be there to ensure the correct length of the CAN message (DLC).

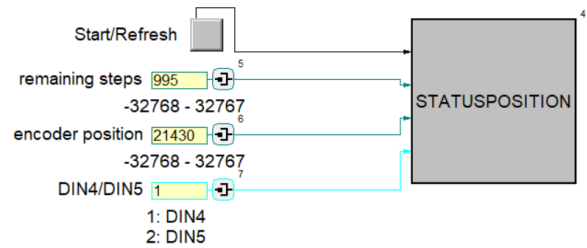
6.4 TEMPLATE 3.4 - STATUS POSITION

Here we can see an overview of the main program.



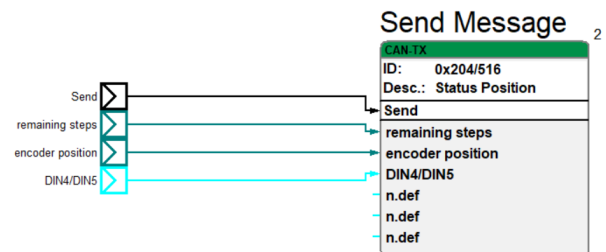
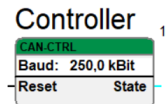
Zur Information!
Bei der STG-800 Serie muss im "Extras" Menü verwendet werden. (weitere Änderungen)

For information!
On the STG-800 series, the C in the "Tools" menu. This setting (Other changes: ->INFO)



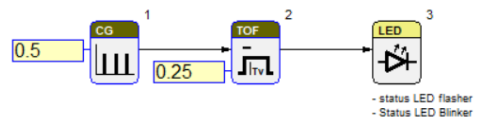
Anleitung zu den Templates im Downloadcenter: 9021-xxxx

And the macro block „STATUSPOSITION“.

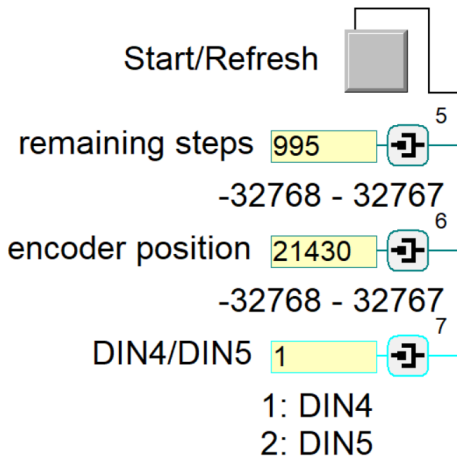


The program sets all the inputs and parameters, and the macro block is what actually controls what the buttons and those parameters do.

Let's take a closer look the the program first.

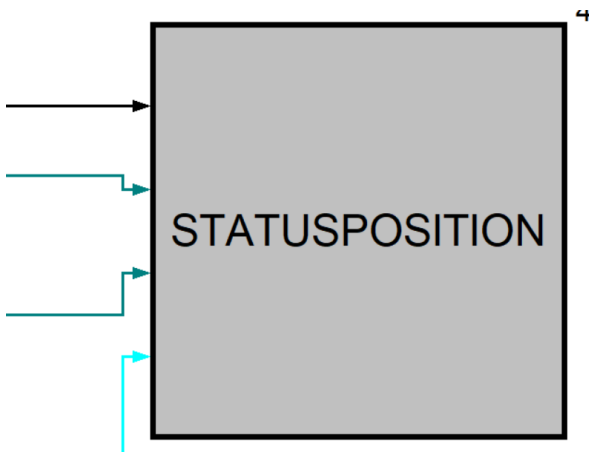


This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.



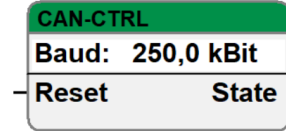
Here we have the inputs that are used to set the different parameters of the MC-800.

1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "remaining steps" sets how many steps, the MC-800 will tell the motor to do, before stopping.
3. The input, "encoder position"
4. The input "DIN4/DIN5" sets whether the the digital input 4 or 5 will be used.

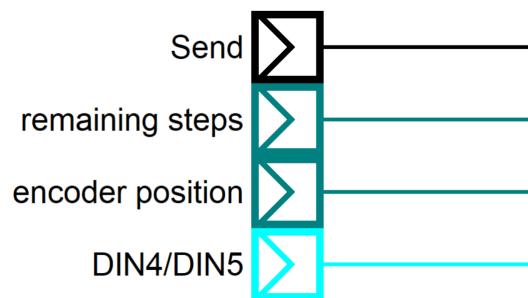


Here, we can all the inputs being sent into the macro block, which we will take a closer look now.

Controller ¹

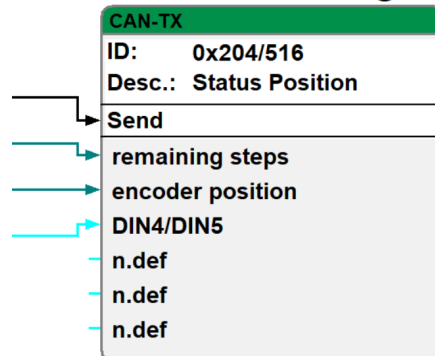


Here we have the CAN controller that always has to be included, when we want to send or receive CAN messages. We use it to set the baud rate, which needs to be the same between the sending and receiving function blocks.

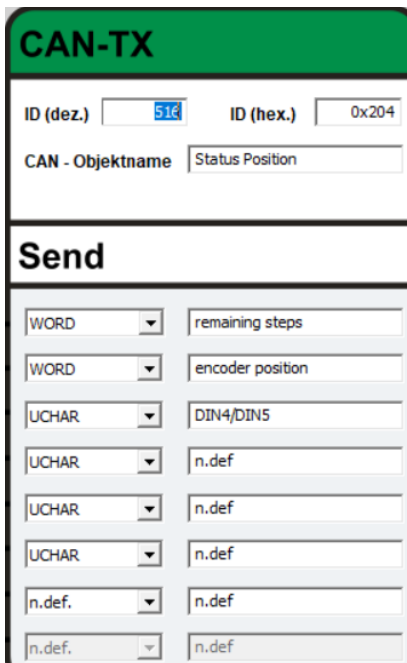


These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.

Send Message ²



The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message, once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."

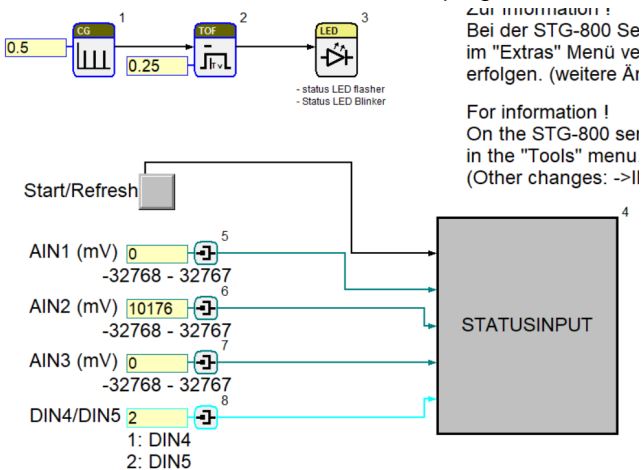


This is the parameter view of the function block.

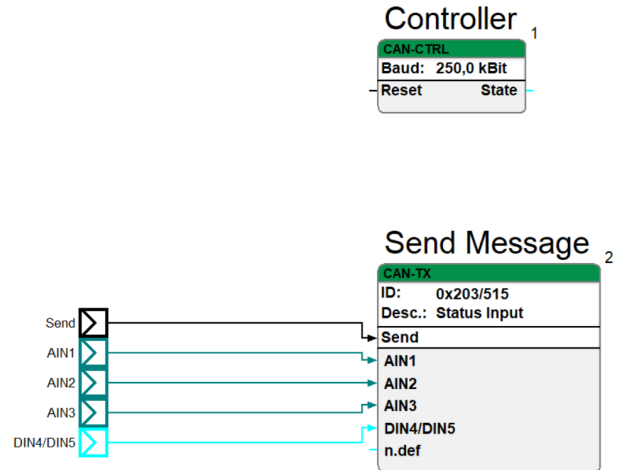
1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name. The not defined inputs that have a datatype assigned to them, need to be there to ensure the correct length of the CAN message (DLC).

6.5 TEMPLATE 3.5 - STATUS INPUT

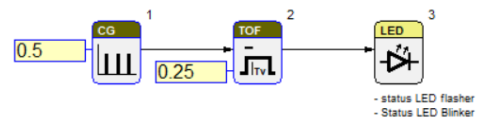
Here we can see an overview of the main program.



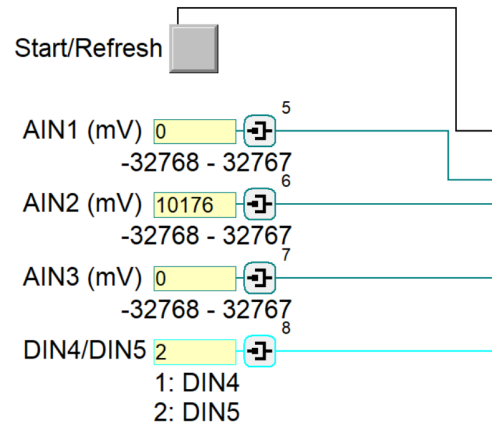
And the macro block „STATUSINPUT“.



Let's take a closer look the the program first.

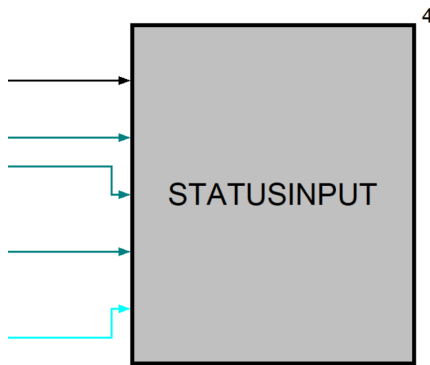


This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.

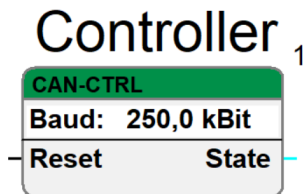


Here we have the inputs that are used to set the different parameters of the MC-800.

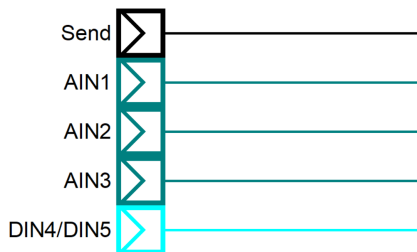
1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "AIN1" through "AIN3" set the corresponding voltages of analog input 1-3 in mV.
3. The input "DIN4/DIN5" sets whether the the digital input 4 or 5 will be used.



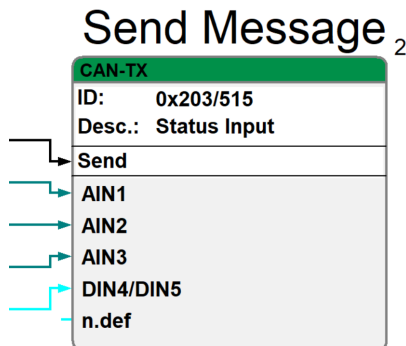
Here, we can all the inputs being sent into the macro block, which we will take a closer look now.



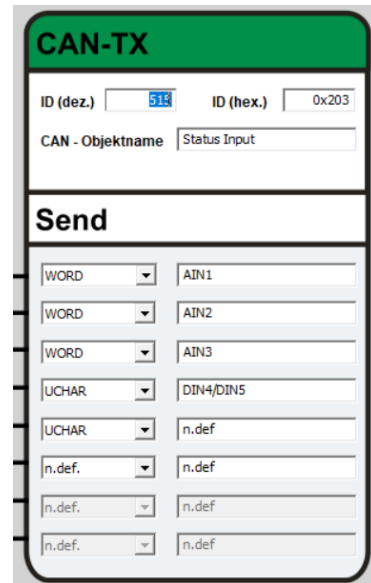
Here we have the CAN controller that always has to be included, when we want to send or receive CAN messages. We use it to set the baud rate, which needs to be the same between the sending and receiving function blocks.



These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.



The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message, once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."



This is the parameter view of the function block.

1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name. The not defined inputs that have a datatype assigned to them, need to be there to ensure the correct length of the CAN message (DLC).

6.6 TEMPLATE 3.6 - SET ID

Here we can see an overview of the main program.

Zur Information!
Bei der STG-800 Serie muss zur Einstellung des COM-Ports der Dialog im "Extras" Menü verwendet werden. Diese Einstellung muss einmalig für jedes Projekt erfolgen. (weitere Änderungen: ->INFO)

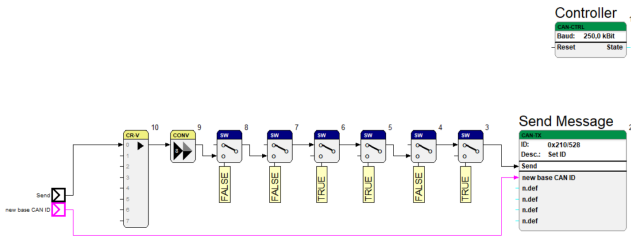
For information!
On the STG-800 series, the COM port has to be set by the dialog in the "Tools" menu. This setting must be done once for each project. (Other changes: ->INFO)

Bitte lesen!
Wenn Ihre Base CAN-ID nicht der Standard Can-ID (0x200) entspricht, können Sie diese unter Makrobausteine -> SETID -> Send Message -> Rechtsklick -> Parameter-Dialog -> ID ändern!
Damit die CAN-ID geändert werden kann müssen die Parameter an den Schalter-Bausteinen zu TRUE TRUE FALSE FALSE TRUE FALSE geändert werden!

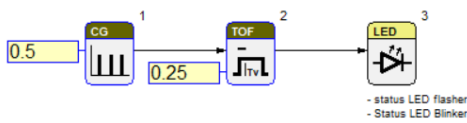
Anleitung zu den Templates im Downloadcenter: 9021-xxxx
Instructions for the templates in the download center: 9021-xxxx

Please read!
If your base CAN-ID is not the default Can-ID (0x200) you can change this under macroblocks -> SETID -> Send Message -> right click -> Parameter dialog -> change ID!
To change the CAN-ID, you must change the parameters of the switch blocks to TRUE TRUE FALSE FALSE TRUE FALSE under macro blocks -> SETID!

And the macro block „SETID“.



Let's take a closer look the the program first.

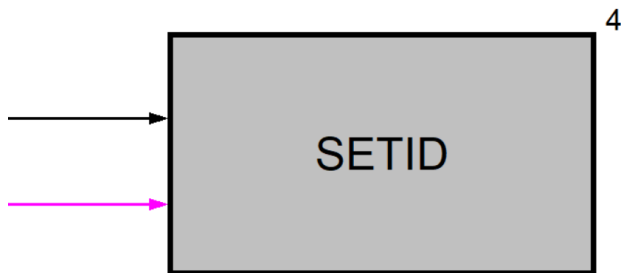


This part of the program makes the LED on the STG-800 blink every 5 s with a 0.25 s long pulse. It's there to indicate whether or not the program has successfully downloaded and is running correctly.

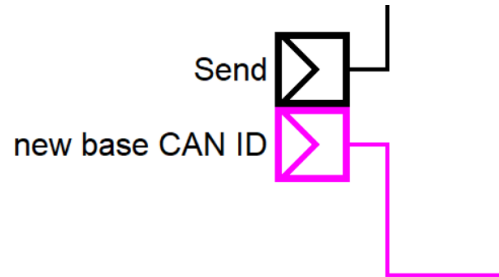


Here we have the inputs that are used to set the different parameters of the MC-800.

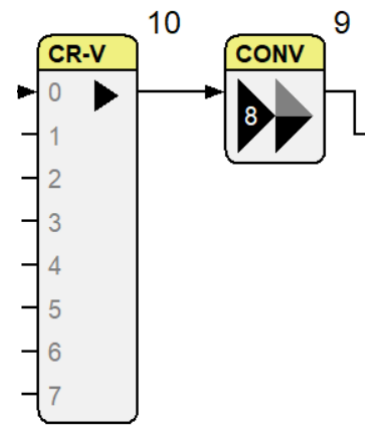
1. The input "Start/Refresh" will be used in the macro block to activate the sending of the CAN message.
2. The input "new CAN-ID (DEC)" can be used to set a new base CAN-ID, in case it's not "0x200".



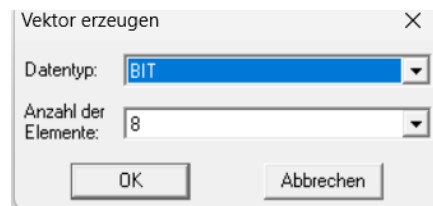
Here, we can all the inputs being sent into the macro block, which we will take a closer look now.



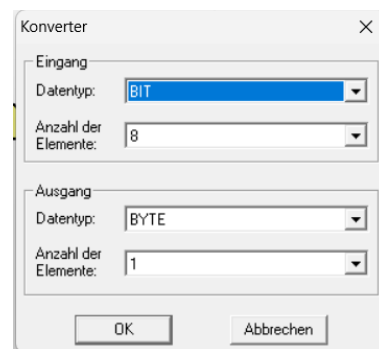
These are the inputs we defined in the program part. We call them here and connect them to the CAN-TX function block.

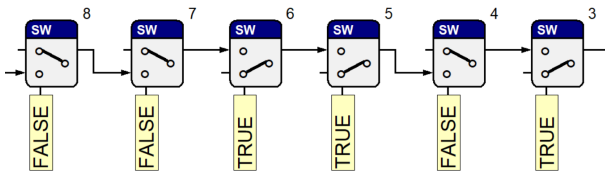


Here, we use a vector and a converter function block to change the datatype from bit to byte. We do this by connecting our "Send" input that has the datatype bit to a vector. Here, we choose our datatype, which is bit in our case.



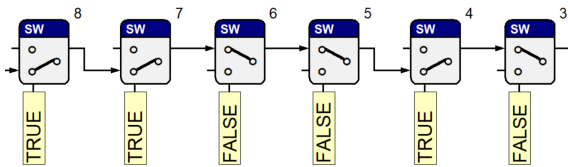
We choose eight for number of elements, turning our one bit into eight. Next, we connect that to the converter function block to change the datatype from bit to byte.



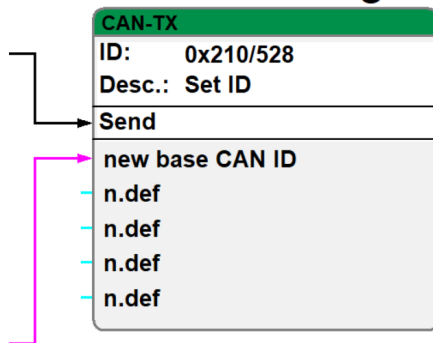


Next, we run that through six switches in series that are toggled in a very specific way.

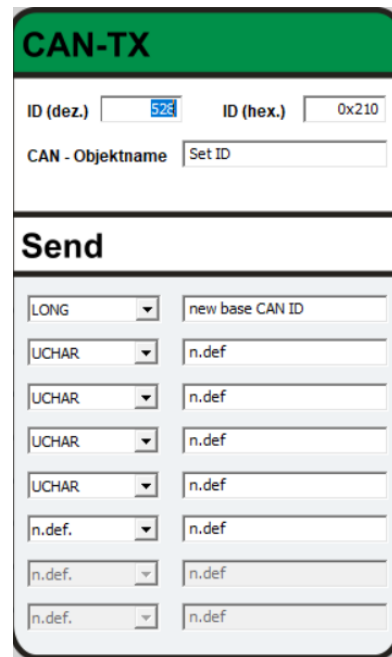
If we want to actually send the message to change the base CAN-ID we have to change all the switches to the exact opposite. Like this:



Send Message ²




The CAN-TX function block handles the actual sending of CAN messages. It has an ID and a description or name we can assign it. Below that, we have the "Send" function that starts the actual sending of the message, once it receives a signal. Let's take a closer look at the parameter view of the function block. Right click and select "Parameter-Dialog..."



This is the parameter view of the function block.

1. The field ID (dez.) and ID (hex.) are linked together and can be used interchangeably. When you enter an ID into one field, the other automatically fills, and vice versa.
2. Below that we have the field, where you can enter a name for the function block. It can be left blank or filled with any name of your choosing.
3. Lastly we have the part of the function block, where we choose the datatype for each input and give it a name. The not defined inputs that have a datatype assigned to them, need to be there to ensure the correct length of the CAN message (DLC).

	Motor Controller MC-800 lococube® Art. No. 0890-0800	<h1>MANUAL</h1>	Page/of: 19/20
			Document: 9021-0032-B
			Date: 11.01.2024
			Revision: B

7 Appendix

7.1 Specifications

7.1.1 General

Hardware design	BARTH® CAN Bus Motor Controller
Interfaces	CAN 2.0A/B

7.1.2 Power supply

Operating voltage	7 to 32 VDC
Current consumption	depending of the connected stepper motor, 2.8 A max
Fusing	3A max. (external) mandatory for voltage reversal protection
Voltage reversal protection	yes (combined with external fuse)
ESD/TVS protection	yes, integrated
Heat dissipation air (at full load)	normally < 5 W

7.1.3 Interfaces

CAN	CAN 2.0A/B: 11/29 bit ID, base frame format supported baud rates: 250 (other rates customer-tailored)
	no internal termination resistor (120R)
	Meets or exceeds the requirements of applications ISO 11898-2, loss of ground protection from -32V to +32V, thermal shutdown protection

7.1.4 Security features

Security Features	System and independent watchdog Fail safe oscillator Power on/down reset Supply voltage supervisor
--------------------------	---

7.1.5 Electrical connection

Electrical Connection	Plugable screw type connector 0.25 - 1.5 mm ² Manufacturer: Phoenix Contact Series: COMBICON Type: MC1,5/4-ST-3,5(-BK)
------------------------------	--

7.1.6 Electromagnetic compatibility (EMC)

Electrostatic discharge (ESD) at supply terminals	20 kV air discharge 30 kV contact discharge (IEC/EN 61 000-4-2, level 3)
Electromagnetic fields	Field strength 10 V/m (IEC/EN 61000-4-3)
CAN bus terminals (CANH, CANL to GND)	IEC 61000-4-2: Unpowered Contact Discharge ±15000 V IEC 61000-4-2: Powered Contact Discharge ±8000 V

7.1.7 Environmental conditions


Operation temperature	-20 to +50 °C (IEC 60068-2-1/2)
Storage temperature	-30 to 70 °C (IEC 60068-2-1/2)
Relative humidity	5 to 80% non-condensing (IEC 60068-2-30)
Air pressure (in operation)	500 to 1500 hPa
Shock resistance	min. 50 m/s ² (IEC 60068-2-27)
Vibration resistance	min. 10 m/s ² @ 10..100 Hz (IEC 60068-2-6)
Degree of protection	IP 40 (without additional gasket) IP65 (with Gasket SEA-23) (EN 50178, IEC 60529)
Free fall (packaged)	1000 mm (IEC 60068-2-32)

7.1.8 Weight and Dimensions

Weight	50 g (without connectors)
Dimensions	60 x 45 x 18 mm (LxHxW) (without connectors)

7.1.9 Ordering Information

Ordering Information	Motor Controller MC-800 Art. No. 0890-0800 GTIN 42513294041566
Ordering information accessory	mini-PLC STG-800 Art. No. 0850-0800 GTIN 4251329401207
	Connection Cable VK-16 (graphical programming) Art. No. 0091-0016 GTIN 4251329400187
	Connection Cable VK-35 (Open Source programming) Art. No. 0091-0035 GTIN 42513129401276
	Programmer ST-Link/V2 ISOL (Open source programming) Art. No. 0017-0066 GTIN 4251329401269

	Motor Controller MC-800 lococube® Art. No. 0890-0800	MANUAL	Page/of: 20/20 Document: 9021-0032-B Date: 11.01.2024 Revision: B
---	---	---------------	--


7.2 Disposal



If you wish to dispose of the product, ask you local recycling centre for details about how to do this in accordance with the applicable disposal regulations.

7.3 Conformity declaration

For the following designated product it is hereby confirmed, that the construction in that technical design brought by us in traffic corresponds to the standards specified below. In the event of any alternation which has not been approved by us being made to any device as designated below, this statement shall thereby be made invalid.

Description	Motor Controller
Type	MC-800
Art. No.	0890-0800
Directive 2004/108/EG relating to electromagnetic compatability (EMC) 	Applied norms: 2004/108/EG 2004/108/EC 2014/30/EU
RoHS Directive 2011/65EU	We herby declare that our product is compliant to the RoHs Directive on restriction of the use of certain hazardous substances in eletrical and electronic appliances

BARTH® Elektronik GmbH
Lengerich, 10.01.2024

D. Barth

Dipl.-Ing. (FH) D. Barth
Managing Director