Introduction

After carefully examination of the first 7 chapters you should get familiar with the ECOSTEP and you should be capable to use it as a positioning drive. This could be:

- Point-to-Point-Positioning
- electronic gear box
- analog nominal value +/-10 V

More complicated applications, especially multi-axe-systems using interfaces like RS485, CANopen or Profibus DP, may require the use of the object list. As a registered user of the ECOSTEP® you will get release notes as soon as there are any innovations.

Wichtig: This documentation requires a thorough knowledge of electric plants and relevant security rules (You will find the security rules with every delivered ECOSTEP®).

Printing this documentation is possible with the print function of the browser. On the left you find Hyperlinks for quick navigation within the content frame. The text contains hyperlinks to other sides. To get back to the starting point press the << - button once or twice.

The drive ECOSTEP/ ECOLIN is distributed with the program **hsio.exe**. For running and adjusting the controller a serial extension lead is needed:

TOP

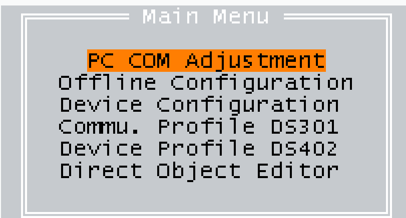
RS232 connection

PC COM1 / IRQ4 oder COM2 / IRQ3	ECOSTEP X5
RxD 2	2
TxD 3	3
GND 5	5

TOP

Parametriersoftware HSIO

If you are within this workshop asked to address the controller with hsio, after starting the program and one click on the RETURN button appears the following picture:



Now make under PC-Interface Settings the above described inputs for COM and IRQ and put in the corresponding ID number (Default 01 H) in accordance with the switch position of the DIP-switch. Press the ESC-button if you want to get back to the former menu. In case of a communication mistake, check the cable or change PC-Interface settings (com1/irq4 or com2/irq3). Otherwise you have reached communicative control over the controller.

The rather abstract structure of the hsio is based on the clear definition of CANopen at the DS402 according to **CIA** , which means "can in automation".

According to this rules the structure of an object- or address - list is generated.

1.0 Installation

This installation is not to be understood as an installation according to matters of EMD (see the following PDF-pages safety instructions), but as a laboratory construction for testing the function of the tool. The following materials are needed:

- Logic - supply (low voltage, safetely saperated from 230 VAC) or JAT ECOBRAX200 with transformator
- ECOSTEP®200 + connector set
- motor + leads (power, encoder, RS232)
- recommended is a mechanical system with limit switches
- power pack (24 - 150 V DC)/ could be supplied by JAT

First you connect motor cables and controller.

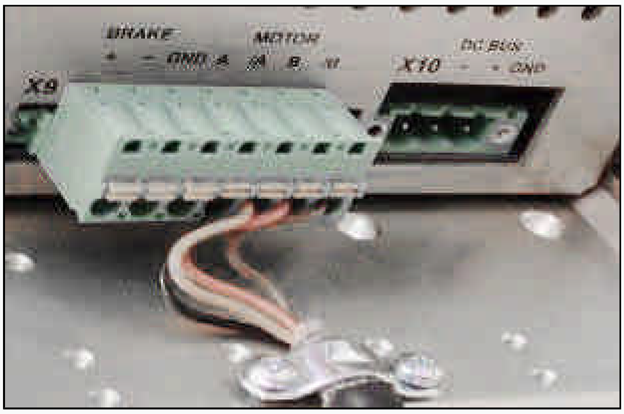
The motor encoder uses signals according to RS422. The encoder is supplied with 5 V by the controller at input X8.

TOP

1.1 Encoder connectors / D-Sub 9 pol. to female X8

ECOSTEP X8	Signal	Pin connector
1	5 V	12
2	A	5
3	B	8
4	N	3
5	free/ (24V)	free/ (24 V)
6	GND	10
7	/A	6
8	/B	1
9	/N	4

The motors possess 2 phases, whose 4 cables are connected with (A/A + B/B) of the connector X9. With the inputs (+) and (-) a holding brake (24 V, 1A) is switched on.



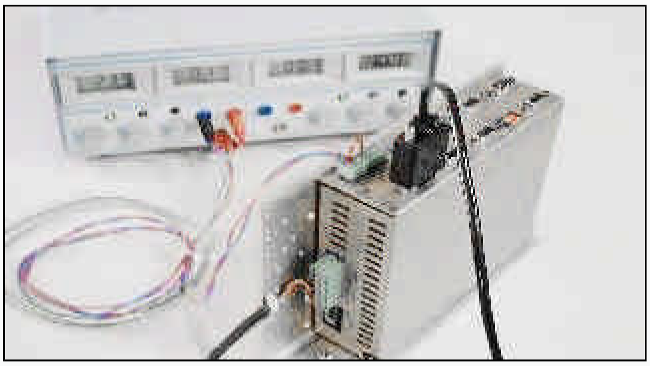
The wiring system of X9 you find in the table below. The values in brackets are in case you are using female connectors. The ground cable and shield should be well mounted on the controller (see picture above). Over 60 VDC the GND-lead is needed.

TOP

1.2 Motor power connector (with or without brake) at X9

ECOSTEP	Nema 34/42 4 Leads	Nema 23 4 Leads	Nema 17 4 Leads
A	3	black	white
/A	1	orange	yellow
B	4	red	red
/B	2	brown	blue
GND	green-yellow	free/GND	free
Brake (–)	separated cable Pin 3	thin black	no brake
Brake (+)	separated cable Pin 1	thin brown	no brake

The logic is separated galvanically from the suppliant. Please mind when connecting the motor with 24 V. The hardware-enable is also connected with 24 V DC if not switched by a PLC.



TOP

1.3 Logic supply

The logic-suppliant of connector X4/Phoenix MC-1,5//3,81 must lie between 18 and 30 V. The outputs need 3 x 0,5 A, the brake 1 A and the controller about 0,3 A.

TOP

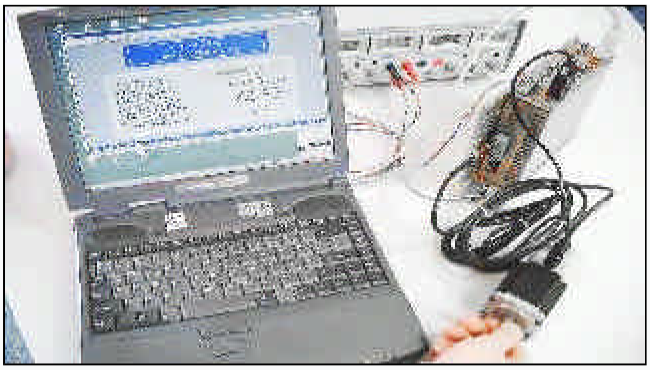
1.4 Hardware reset

After connecting the LED "run" blinks and "24 V" shines steadily green. If not, please check the logic tension and the encoder connections. You can RESET the mistake in turning the logic on and off or shortly lay 24 V at the reset- input of connector X4.

TOP

1.5 Feedback check

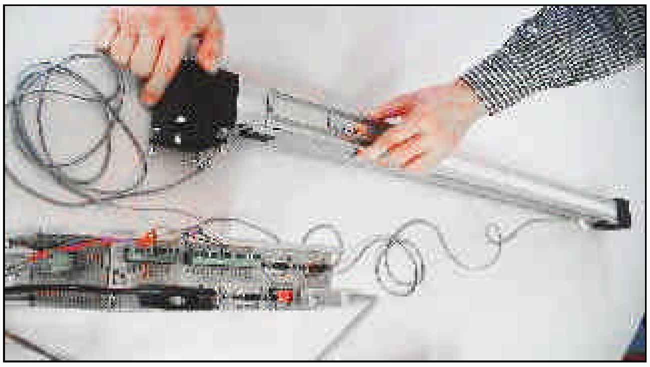
The connector is now in its basic status. If you connect a PC via RS232 with the connector and call up the program hsio, it is possible to see the encoder data when turning the motor shaft (**Main Menu \ Device Profile DS402 \ Profile Position Mode**). If you have the master encoder also connected you should control its working inside menu (**Main Menu \ Device Configuration \ Master Encoder**).



TOP

1.6 Limit switch adjustment

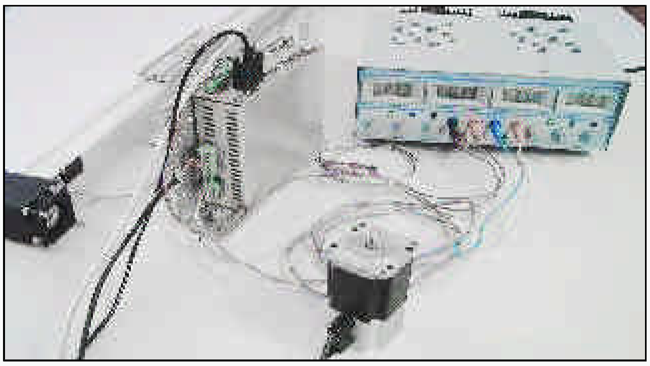
The next step is adjusting the limit switch. This is generally a low-volt active/high-volt-active. We connect (+) with 24 V from connector X4 and the signal leads with the inputs of DIN6 and 7 and the signal mass with "ground" under DIN8 on connector X3. The setting is generally connected to the reference drive, that is why both items have got a separate chapter Appendix A-Limit switches.



TOP

1.7 Powersupply

As last step in laboratorial installation we connect the controller with the power supply. The DC bus of the amplifier needs not to be stabilised but smoothed. It arrives at connector X10/Phoenix MSTB - 2,5//5,08. Now the motor is ready and we can start with the next chapter.



2.0 First move

Switching on the motor without load ! Before doing this we make a final check and we proof whether the commutation parameters set correctly and the maximum current is set according to the motor data sheet. Then we learn a little bit about a state machine to get the system running.

2.1 Commutation

To check the first point we go inside hsio to:
(*Main Menü\Device Profile DS402\Commutation Parameters*)

Commutation Parameters

tc_para.tc_commu_length	160
tc_para.tc_commu_poles	0
tc_para.tc_commu_v_preph_factor	80
tc_para.tc_commu_c_preph_factor	0
tc_para.tc_commu_max_prephase	60
tc_para.tc_commu_find_current	1032
tc_para.tc_commu_find_delay	1000
tc_para.tc_commu_find_damping	0
tc_para.tc_commu_find_method	0
tc_para.tc_commu_offset	40
pos_enc_res.encoder_increments	8000

If you purchased a rotative motor you will have normally the above parameter preset. We see the 10 parameters of the object block 0x60F6. But anyway the important parameters are *find_current (60F6,06)*, the velocity dependend flux angle *v_preph_factor (60F6,03)* and the encoder resolution parameter *commu_length (60F6,01)* together with the amount of electrical poles *commu_poles (60F6,02)*, which is identical with the pole pair number at rotative motors and identical 1 for linear motors.

For example a servostepper with 50 pole pairs and an encoder with resolution 2000 inc/ revolution has after the internal interpolation by four 160 pulses per pole pair. If we write a zero into *commu_poles* the controller thinks we have done the division 8000 by 50 and so we have to write 160 into *commu_length*.

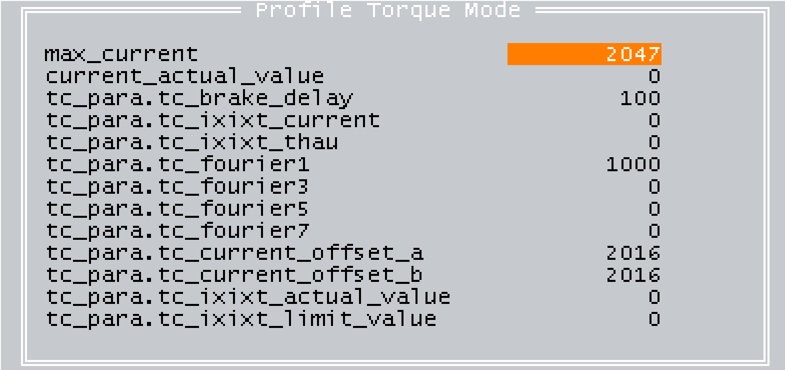
The internal division is useful if one want to drive a motor with 30 pole pairs with a 2000 puls encoder. The quotient is non regular, but the controller takes care about that. In this case we write 8000 into *commu_length* and 30 into *commu_poles.*

If you use a **linear motor** of pole pair length 32 mm together with an linear encoder of resolution 1 µm you write 32 000 into *commu_length* and 1 into *commu_poles*! At the object *commu_find_method* you choose the commutation method. In general 3 is used for horizontal and 1 for vertical systems.

TOP

2.2 Current adjustment

To adjust the **maximum current** we go into the following menu *Profile Torque Mode*.



Important parameters for you as a user are:

- *max_current (6073,00)* that is the upper limits of the the current
- *current_actual_value* is the actual value set by the controller. With this value we set the integral gain parameter according to the friction current.
- in *tc_ixixt_current* and *tc_ixixt_thau* we could set the current limit in time measured in time period thau/seconds. This is used as a safety function if one drives on block and the following error couldn't come.

Finally we proof the default values of the position and velocity loop parameter, which are set for **free shaft operation**:

	Main Menu\Device Profile DS402\Profile Velocity Mode	
	vel_para.vc_kp	50
	vel_para.vc_ki	1
	vel_para.vc_ilim	400
	vel_para.vc_error_filter_length	1
	vel_para.vc_output_filter_length	3
	Main Menu\Device Profile DS402\Position Control Function	
	pc_para.pc_kp	3000
	pc_para.pc_vfff	10000
	Main Menu\Device Profile DS402\Profile Position Mode	
	profile_acceleration	10 000
	profile_deceleration	10 000

TOP

2.3 State control

Now we have made the presets and can look at the state control of the drive:

Main Menu\Device Profile DS402\Device State Control:

```
═══════════════════════ Device State Control ═══════════════════════
      Control Word:                                  Status Word:

 Switch On                        1    Ready to Switch On            1
 Disable Voltage                  1    Switched On                   1
 Quick Stop                       1    Operation Enable              1
 Enable Operation                 1    Fault                         0
 Setpoint  | Res. |  Start Home   0    Voltage Disabled              1
 Immed.    | Res. | Res.          0    Quick Stop                    1
 abs./rel.| Res. | Res.           0    Switch On Disable             0
 Reset Fault                      0    Warning                       0
 Halt                             0    Manufacturer specific 1       0
 Reserved 1                       0    Remote                        0
 Reserver 2                       0    Target Reached                1
 Manufacturer specific 1          0    Internal Limit Active         0
 Manufacturer specific 2          0    Setp.Ack. | v=0  | Hom.att.  0
 Manufacturer specific 3          0    Foll.Err. | Res. | Hom.Err.  0
 Manufacturer specific 4          0    Commutation Found             1
 Manufacturer specific 5          0    Reference Found               0
```

With the help of the cursor we can move at the left side up and down. By pressing RETURN one could set 0 to 1 and vice versa. The numbers in the column define the "control word" which sends orders to the controller. The left side is just readable because it shows the status of the controller. For more see state machine.

Control word (6040,00): With this 16 bit object we control the status of the controller:

- Default state after logic on and power off :
  (0000 0110) = 2^1+2^2 = 0x6 Hex
- Controller on:
  (0000 1111) = 2^0+...2^3 = 15 DEC = 0xF Hex
- Homing Start (if operation mode is 6):
  (0001 1111) = 31 DEC = 0x1F Hex
- Relative positioning:
  (0100 1111)- >(0101 1111) = 0x4F - > 0x5F Hex
- Absolute positioning immediately:
  (0011 1111) = 63 DEC = 0x3F Hex
- Error Reset:
  (1000 0000) = 128 DEC = 0x80 Hex

At the *Status word (6041,00)* we have the following important states:

- Ready to Switch on = 1 DEC = 0x01 Hex
- Fault = 8 DEC = 0x08 Hex
- Target Reached = 1024 DEC = 0x400 Hex
- Commutation Found = 0x4000 Hex
- Reference Found = 0x8000 Hex

TOP

2.4 Modes of operation

In the menu **Device Control** one can see the status word as hexadecimal number and can set the control word also as hexadecimal number. But mainly we use this menu to set the *modes_of_operation (6060,00)*.

```
                    Device Control
controlword                              0000000f
statusword                               00004437
shutdown_option_code                            0
disable_operation_option_code                   0
quick_stop_option_code                          0
stop_option_code                                0
fault_reaction_option_code                      0
modes_of_operation                              1
modes_of_operation_display                      1
```

The most important **operation modes** are:

- Value 1 = Positioning (absolute or relative)
- Value 3 = velocity mode with following error control
- Value - 3 = velocity mode without following error control
- Value - 4 = positioning mode without profile control, is used if one uses the interface master encoder X7 as nominal value (e.g. Step & Direction or electronic gearing)
- Value 6 = Homing mode
- Value 7 = interpolation mode with CANopen

TOP

2.5 Fast start

Now we switch on the drive::

- the shaft is free
- the LED 24V is green and the LED from RUN is flashing green
- In *Device Control* the modes of operation is 1 and the control word is set to 0x0F
- we check, whether the bit Commutation Found in Device State Control is 1 or the status word has the value 0x4437 and the motor shaft resists against manually rotation
- if so we set the control word in *Device Control* to 0x3F

after this we go to menu ***Profile Position Mode***:

```
                    Profile Position Mode
target_position                            8000 inc
min_position_limit                            0 inc
max_position_limit                            0 inc
max_profile_velocity                   30000000 inc / 64 s
profile_velocity                        3000000 inc / 64 s
profile_acceleration                      50000 * 16 inc/s²
profile_decelaration                      50000 * 16 inc/s²
quick_stop_deceleration                  100000 * 16 inc/s²
motion_profile_type                           0
position_demand_value*                     8000 inc
position_actual_value*                     8000 inc
```

Now we set the *target_position* to 8000 if *profile_velocity* equals 0, nothing happens and we set the *profile_velocity* to 3 000 000 (350 U/min).

Attention - the motor should rotate one revolution:

- It is making one revolution and the green LED's are still on. Congratulation!

- The motor is not running. Pity! Please check the following things again: Switch off the logic - switch it on - start again at controller adjustment
- If you are here the second time without success, then check the error with the error code (main menu\device configuration\error flags) and call your application engineer. If you've been successfully you can proof also the homing and the limit switches.

To do this go back to menu **Device Control**, set the *control word* to 0x0F and choose 6 for the *modes_of_operation* and switch the control word again to 0x1F - for more info click here. We proof the limit switches by setting a new positive *target_position* and activate DIN6. The motor shouldn't run further unless DIN6 is deactivated - for more info click here.

Anyway if you have still problems, check this. Above steps could be automated by the use of offline programming.

Appendix A - Limit switches and homing

This chapter shows how to configure the end switch with **free motor shaft** and how to choose the reference run mode. This is necessary if your drive system is one which possesses clearly defined starting and ending positions. These are all screw drives and belt systems with a permanently installed vehicle.

Limit switches

The end switches are connected with the **inputs of X3**:

- **DIN6** for limit +, i.e. limit in positive counting direction of the motor
- **DIN7** for limit-, i.e. limit in negative counting direction of the motor
- **DIN8** for the reference sensor

You can make out the counting direction of the motor by the change of the actual position when turning the motor shaft by hand. The actual position is shown in hsio as *position_actual_value* in (Main Menu\Device ProfileDS402 \Profile Position Mode). With the delivery configuration it happens generally when the motor shaft is turned anti-clockwise.

Now appears as described in the introduction the main menu. By pressing "return" you arrive the **device configuration**. Within this menu you can navigate with the arrow buttons. By pressing ESC, you get one level higher. Value inputs and submenus are reached by RETURN.

```
Device Configuration

       Transient Recorder
      Application Commands
        Application Data
          Error Flags
          Analog Ports
 Master Encoder Configuration
    Joystick Configuration
   Digital Input Configuration
 Digital Output Configuration
 Analog Monitor Configuration
      Direct Object Editor
```

```
Digital Input Configuration

Input polarity mask          00
Input state                  40

LIMIT+        OR mask        00
             AND mask        20
             CMP mask        20
               Result        00

LIMIT-        OR mask        00
             AND mask        40
             CMP mask        40
               Result        01
```

Arrived in Digital Input Configuration you see by Input State which inputs are set. The value is coded with a normal SPS bitcode (glossary).

Input 6 "high as logic high" means 0x20
Input 7 high means 0x40

Input 6 + 7 high mean 0x60

High in our context means always *logic activ* . Which physically level is meaned will be determined by the Input Polarity Mask.

In the above picture the Input Polarity Mask (inverts the masked inputs) is zero. The Input State is 0x40, which means that input 7 is "high". If you enter 0x60 for Input Polarity Mask, in Input State appears a 0x20 Hex and input 6 is now logically active whereas input 7 is logically inactive, because we have defined the inputs 6 and 7 as closers!

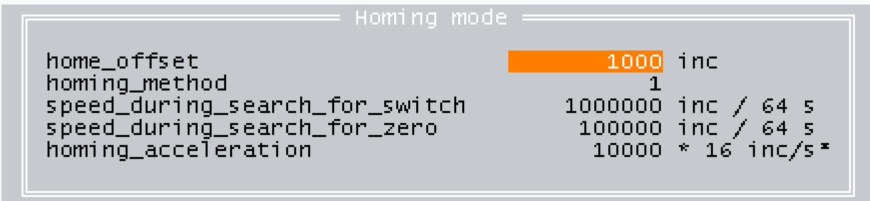
In generell the value in Input State has to be compared with an AND or OR connection with the value in CMP mask and gets a "1" in result if the boolean value is TRUE. This 1 will be postprocessed within the controller (e.g. blocking the motor).

Now you can check if your limit switches function on the right side by activating them (1.0 Installation > Limit switch-adjustment). Please activate also the reference switch DIN8. Within Input State must be added 0x80.

TOP

Homing

After configuring the limit switches we choose the reference run. We **start the homing** in chapter First move. To configure the reference run we enter the Device Profile DS402 menu and there the submenu **Homing Mode**



with the following parameters:

- **home_offset** moves the zero (origin) by X increments
- **homing_method** defines the reference method.
- **speed_during_search_for_switch** defines the velocity to find the limit switches
- **speed_during_search_for_zero** defines the velocity to move to the index or reference switch after the limit switch has been found
- **homing_acceleration** defines overall acceleration during homing mode

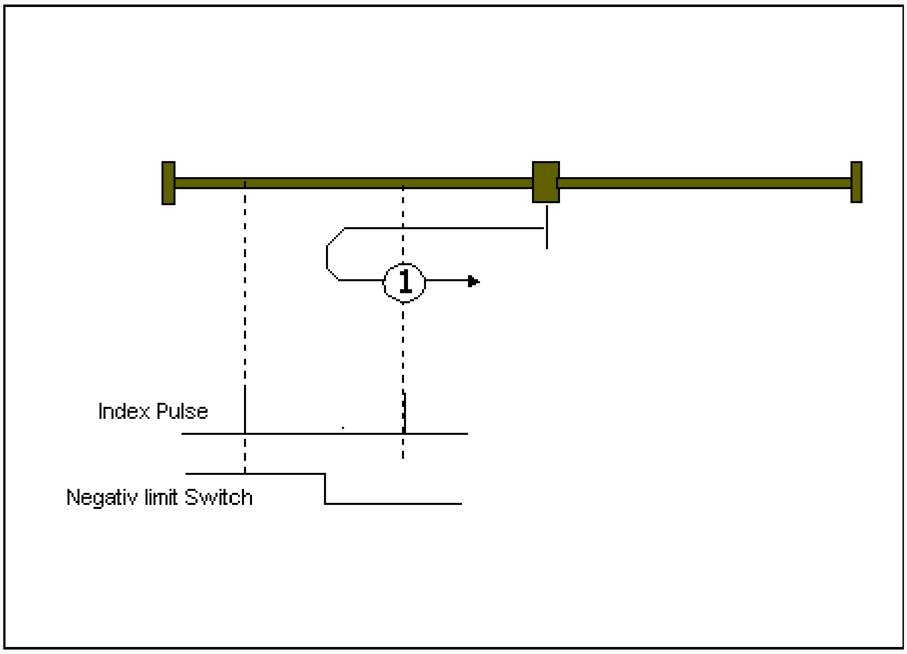
In the above example one moves with 117 rpm to the low high side of the negative limit switch. It is kept in the low district of the limit switch with about 11,7 rpm on the first index impulse of the encoder. The acceleration is about 20 rad/s².

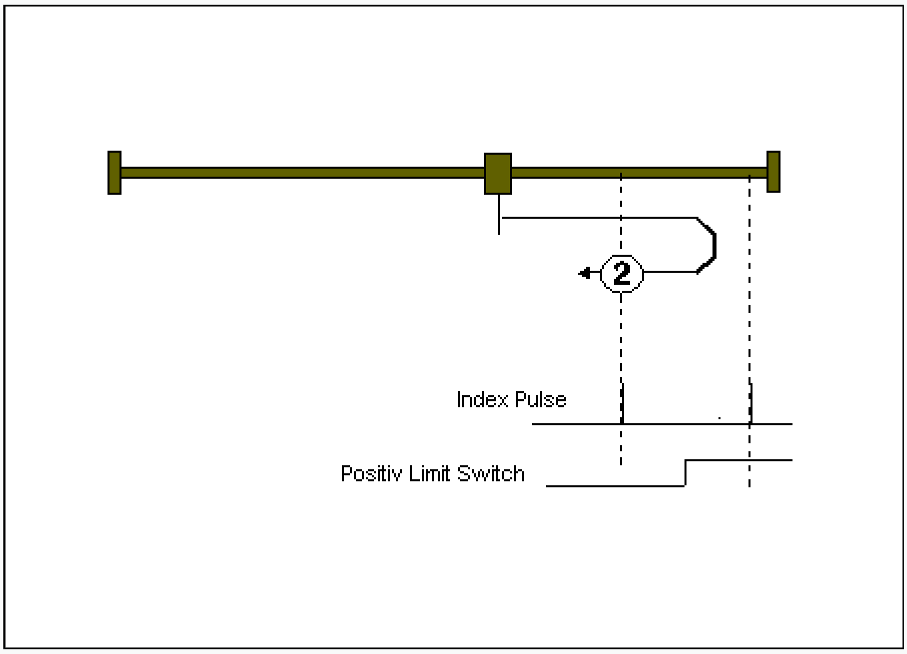
TOP

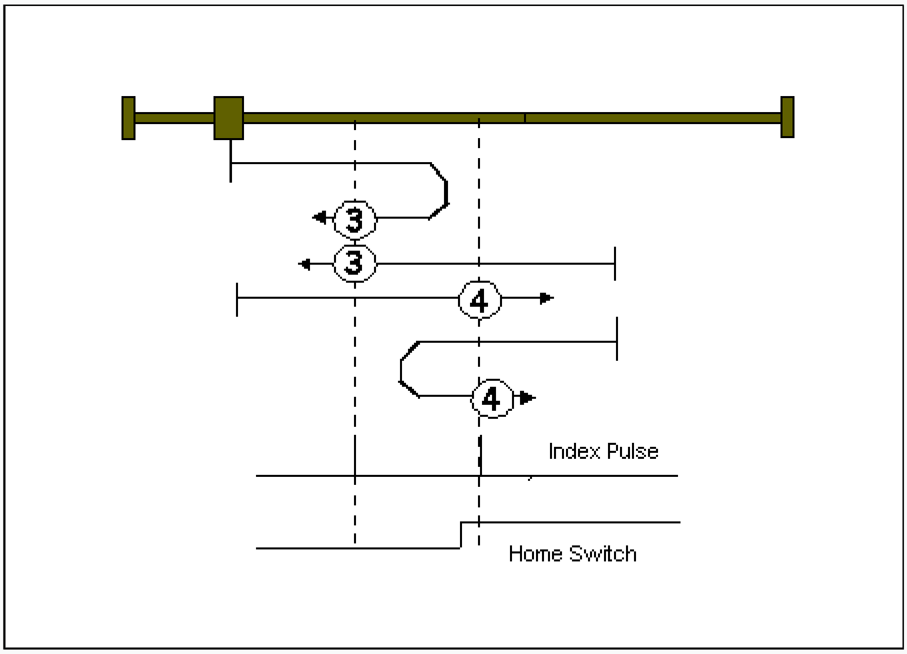
Homing methods

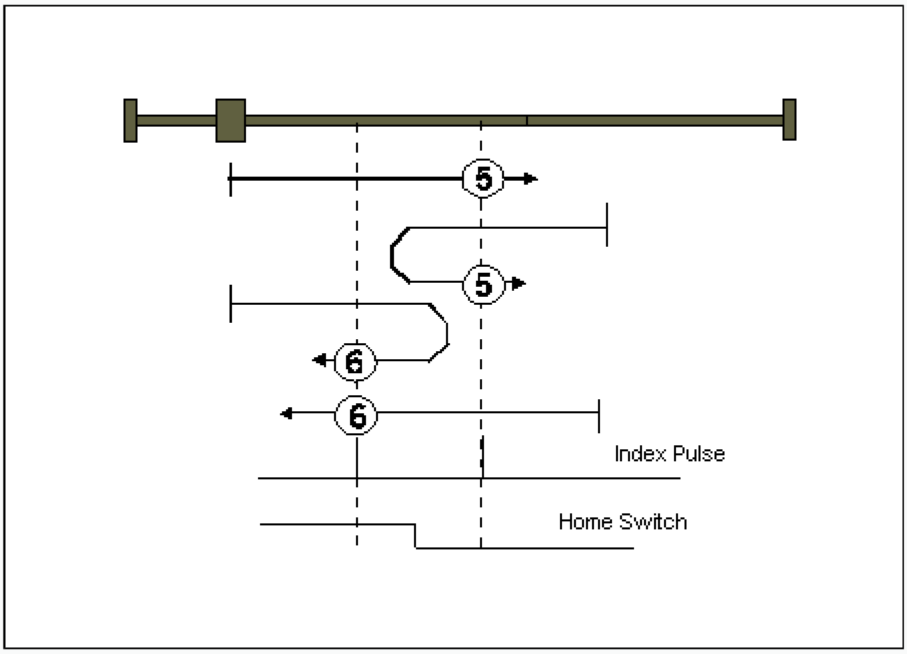
The table below connects the type reference run with the according number of CANopen DSP402.

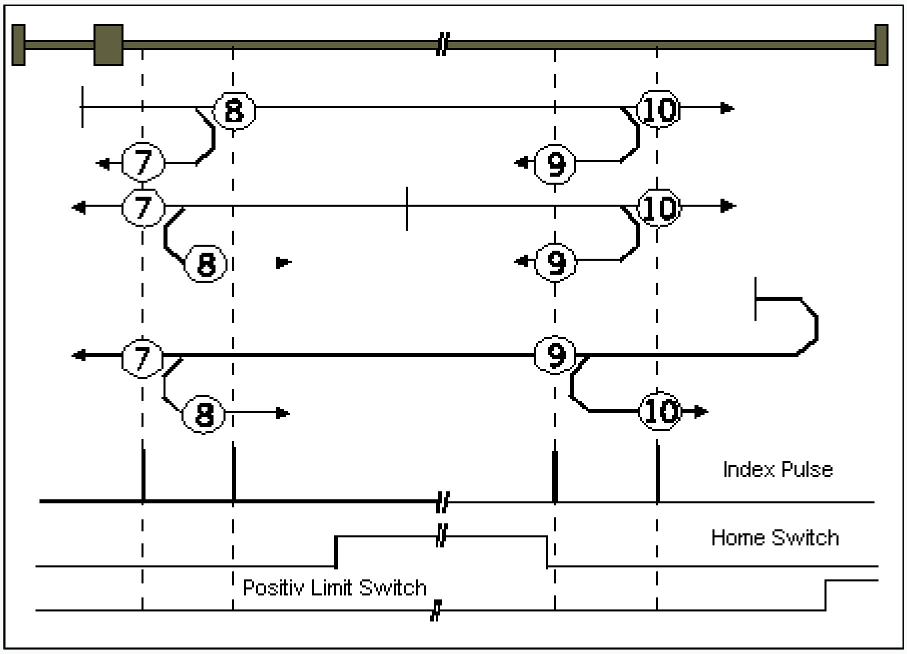
Reference modi (Homing)

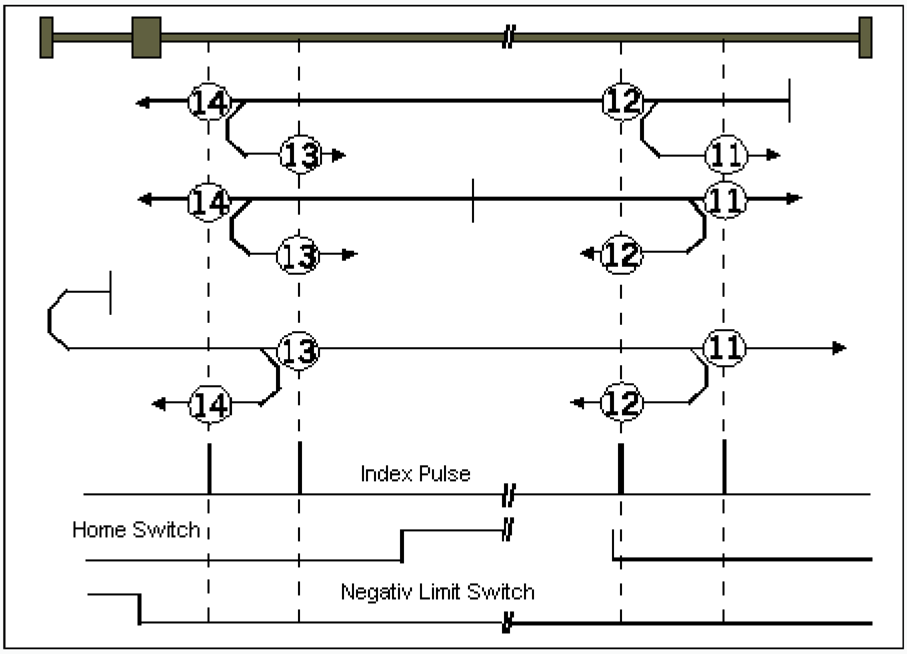


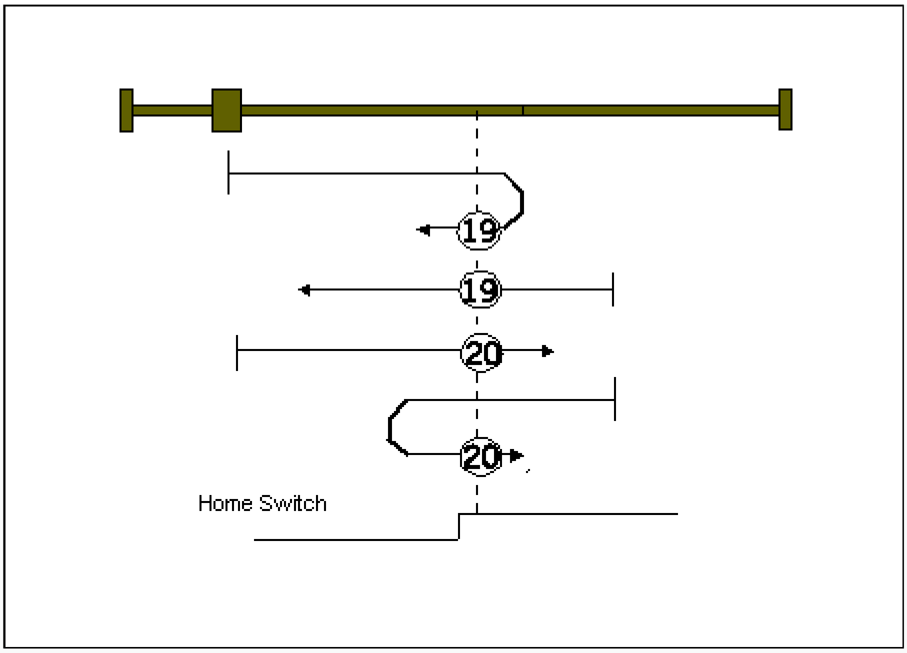


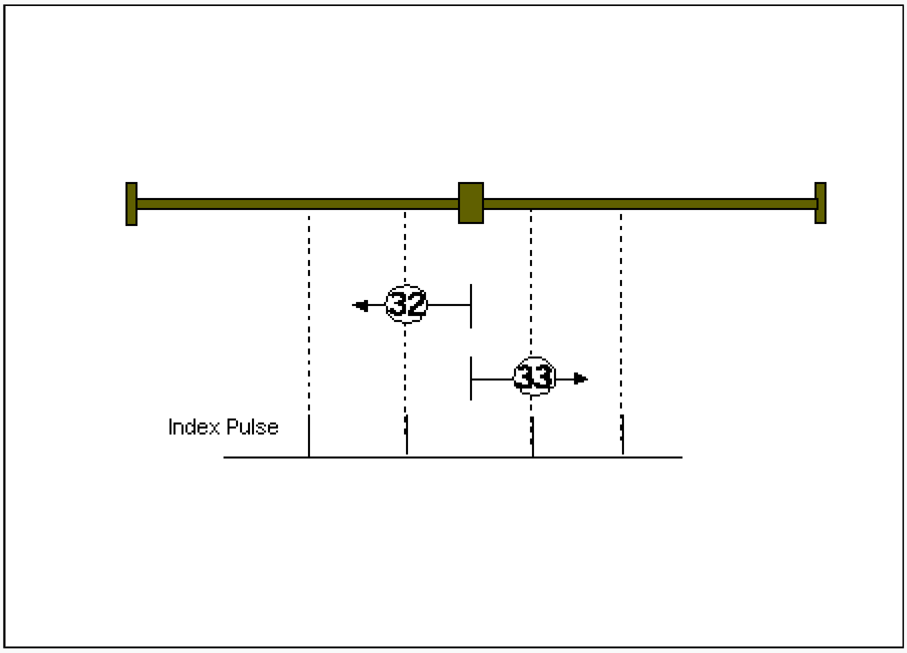

3
3
4
4
Index Pulse
Home Switch


5
5
6
6
Index Pulse
Home Switch


Index Pulse
Home Switch
Positiv Limit Switch


Index Pulse
Home Switch
Negativ Limit Switch





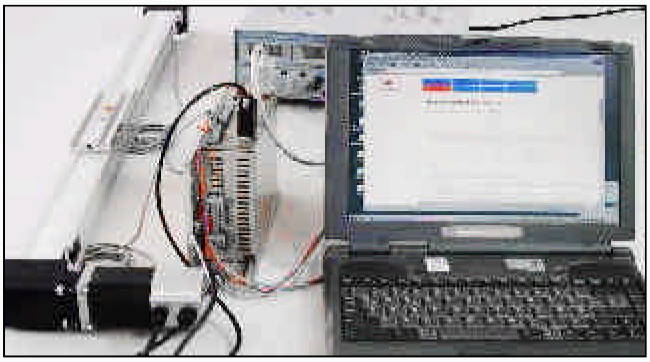
Because the methods are quite similar, only some are explained:

Method 1: see above.
Method 3: It is driven up to the L-H-puls of the reference sensor. At the 1. index impulse it is stopped where the reference switch is low.
Method 7: It is driven from sidewards to the L-H-flank of the reference switch. The positive limit switch is low and zeroes with the next index impulse, where the reference switch is low.
Method 17 - 31: Are like methods 1 -14 but without index impulse. The example shows 19 and 20 which are comparable to 3 and 4.
Method 32: first index impulse anti-clockwise

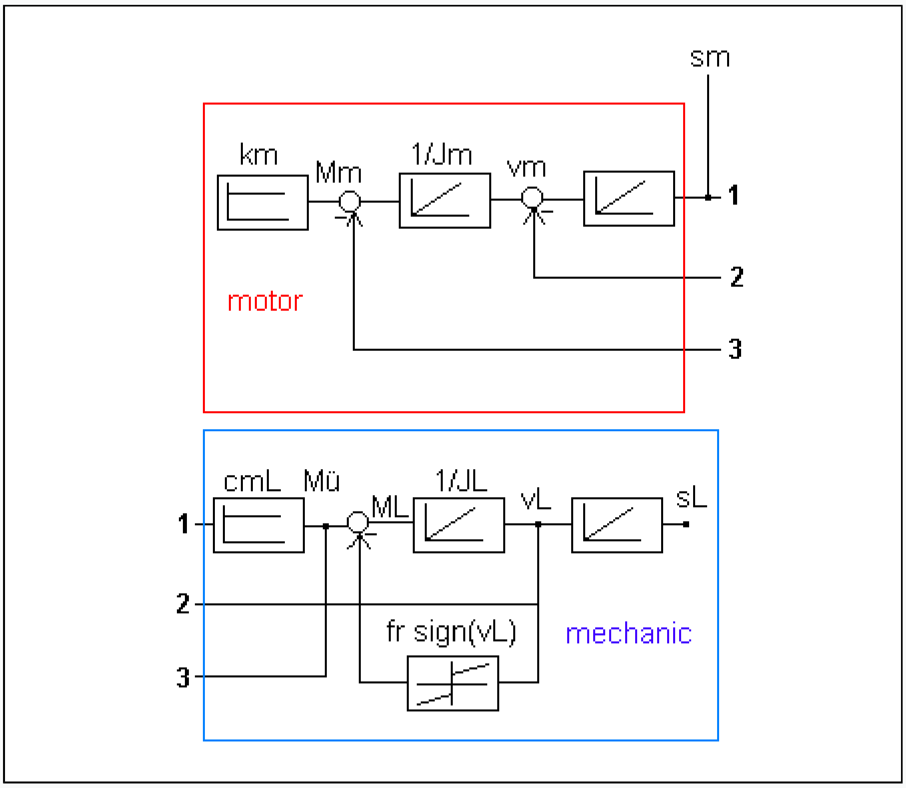
Method 34: not described, assigns the zero on actual position.

Appendix B - Mechatronic

Now we put the motor on the load:



☐ **Electromechanical System:**



Welcome to adjust the above motor drive system.

Take some time to understand the above situation. The abbreviations are explained at the next table otherwise we used the common signs for integration and multiplication in a control network.

	Abbreviation	Meaning
	km	torque constant
	Mm	torque
	Jm	motor inertia

	vm	motor speed
	sm	value motor encoder
	cml	mechanical elasticity
	Mü	transfered torque
	ML	load torque
	JL	load inertia
	vL	load speed
	fr sign(vL)	nonlinear friction function
	sL	position of the load

You have to consider that one moves a load just by generating a static angle between the load- and the motor shaft. Because of this there is always an elastic reaction of the load (torque transmission). The only system where this is not like this is the direct linear motor, but only if the bottom under the motor is not swinging and the friction force is low compared to the cogging of the motor, otherwise we have the same as in the rotative case. What happened often is that under wrong controller parameter the motor tends to swing and one could hear and feel it. The result is that the position stiffness is low.

TOP

Problem outline

To understand this oscillation and what is the reason for it and what kind of parameter have to be adjusted to prevent this we draw the following picture in mind:

We think of two ball systems, one with mass 10 kg coupled by a spring with your hand and another 10 kg ball coupled with a steal rod with your hand. If you try to move the one with the spring you will generate a special stretching length depending on the acceleration and the friction constant of the bottom. If you have a variation of the friction constant it will be quite difficult to generate a constant moving of the ball, especially if the frequency of the variation is in a special area. Then you push and pull but the ball doesn't do what you want. In this case the system is instabil. With the rod system there will be no problem as long as your arm are not getting elastic. May be after some time you check it also with the spring and depending on the spring stiffness you realize two things:

1. the time to change the velocity of the ball could just generated up to a special inverse frequency ($T = 1/f$)
2. you keep cool if there is just a small but high frequency riple on the velocity according to friction nonlinearities

What did you do? - You have just set in your mind a low pass filter, which means it does not matter you if there are some high frequency oscillations on the velocity due to external distortions with rather low amplitude.

TOP

Control strategy

Therefore we learn here to set a low pass filter into the drive and to find the right gain of the error correction due to the proportional and integral parameter of the velocity control and the proportional parameter of the position control.

Firstly we calculate according to the load inertia and the available torque the maximum acceleration needed in that way, that there will be enough torque to compensate the friction. This is normally done together with the sales engineer before buying. The result is a special motor and the right drive size (ECOSTEP200 + 42N or 34N or ECOSTEP100 + 23S or 17S).

Secondly we determine the highest gain crossover frequency depending on the resolution of the encoder, the maximum torque and the inertia of the load. This is theoretical done with the help of our Applet *adjust.xls* and then by moving the load by hand against the motor torque with modes of operation set to - 3 and integral gain vel_para.vc_ki set to Zero. You should notice a high damping force without hearing oscillations in the determined frequency range. Otherwise slow down the gain vel_para.vc_kp and take care that the bode curve of the position loop doesn't get a resonance. In this case turn pc_para.pc_kp down.

^ TOP

Adjustment

Finally we optimize the stabile system with the integral part of the velocity control objects *vel_para.vc_ki* and *vel_para.vc_ilim* and with the velocity feed forward parameter *pc_para.pc_vfff* to obtain a minimum average following error. Sometimes (the mechanical rigidity is too low) you might set also the torque feed forward parameter *pc_para.pc_afff* to achieve better performance. It is suitable to move between 2 position automatically during this tuning.

During the above tuning we will stay in the two following menu at the software hsio:

Profile Velocity Mode

target_velocity	0
vel_para.vc_kp	100
vel_para.vc_ki	1
vel_para.vc_ilim	500
vel_para.vc_error_filter_length	1
vel_para.vc_output_filter_length	3
velocity_sensor_actual_value	0
velocity_demand_value	0
velocity_actual_value	0
velocity_control_effort	2

Position Control Function

following_error_window	2000	inc
position_window	10	inc
position_window_time	0	inc
position_demand_value*	14826	inc
position_actual_value*	14645	inc
control_effort	16000	
pc_para.pc_kp	4000	
pc_para.pc_amax	0	
pc_para.pc_vfff	16384	
pc_para.pc_afff	0	
pc_para.pc_cffo	0	
pc_para.pc_cfff	0	
pc_para.pc_cffl	0	
pc_para.following_error	-2	
pc_para.prof_vel_val	20000	
pc_para.long_spion1	58	
pc_para.long_spion2	0	
pc_para.long_spion3	0	
pc_para.long_spion4	0	

Appendix C - Interfaces

Overview

The ECOSTEP contains several interfaces, that can act parallel to set new nominal values in the drive. We will describe the most importent in this chapter.

Analogous Interface (+/-10 V)

One of the still common interface is the is the +/-10 V (connector X3: AIN+/signal and AIN- /ground where GND means shield) interface. To tell the ECOSTEP to read from this interface, one could set the parameter through hsio or direct address access . We will try to do it through the parameter set up software. Please go to the following menu (.\device configuration\analog ports):

```
Device Configuration

      Transient Recorder
     Application Commands
       Application Data
          Error Flags
         Analog Ports
 Master Encoder Configuration
    Joystick Configuration
  Digital Input Configuration
 Digital Output Configuration
 Analog Monitor Configuration
      Direct Object Editor
```

```
Analog Ports

Analog Input Register Mapping 60FF0020

+00000 * 03000 * (2^+003) =            0

highres_mode  0      highres_active  0

value[6]  = {     1,       4,       1,
                  0,      73,     801 }
offset[6] = {   502,     503,     509,
                506,       0,       0 }
ADC[6]    = {   502,     507,     509,
                514,      73,     801 }
```

You have to change the red marked fields. The first one defines the target object that will get the nominal value. In the above picture it is the nominal velocity if you want to run the controller in closed loop velocity mode, that means mode of operation is - 3. If you don't have an external position controller you should use mode of operation 3, where the position controller is included. The second field determines the resolution together with the third field - see also scaling formula (Appendix F - Glossary). In our example the maximum velocity at 10 V would be 1440 rpm. .

To start the regulation through the analogous port, one has to set the control word to 0x000F and the mode of operation to - 3 or 3.

If one wants to set up a continuous change of limitation of the drive current through the analogous port, one uses the maximum current (6073,00) as target object and defines the resolution with factor = 2 and shift = 0. This would give resolution of 10 mA if the maximum value at 10 V should be 6 A.

TOP

Master Encoder - Interface (RS422)

The master encoder interface X7 has the same pin code as the motor encoder interface (1.0 Installation > Encoder connection diagram), except that it can supply at PIN 5 additionaly 24 V for special measuring systems. To configure a master- slave connection to several drives an encoder output X8 is connected with the master encoder input X7 of one or more drives, which are then slaves in the chain. To implement this interface, also directly with the object catalogue, one has to follow up the next steps at (.\ device configuration\ master encoder configuration):

```
Master Encoder Configuration

Master Pos Mapping      00000000
Master Vel Mapping      60FF0020
Gear factor                10000
Gear divider               10000

Counter mode                  01
Master Pos Actual             -1
Gear Pos Actual               -1
Gear Vel Actual                0
```

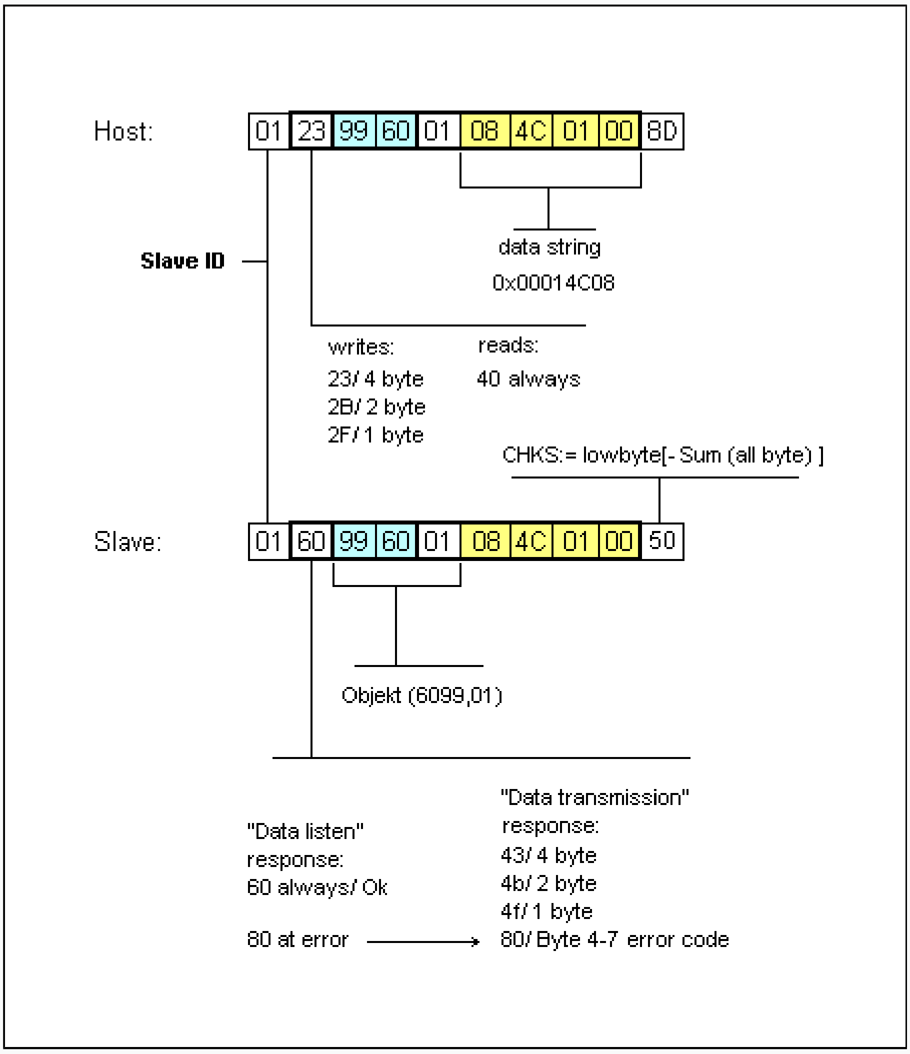
```
Master Encoder Configuration

Master Pos Mapping      00000000
Master Vel Mapping      60FF0020
Gear factor                10000
Gear divider               10000

Counter mode                  01
Master Pos Actual             -1
Gear Pos Actual               -1
Gear Vel Actual                0
```

We have to configure the red marked fields. In the first field we find again the target object address to assign the values from the RS422 port. Normally we take the velocity object (60FF,00). In the second field we choose the countermode" 1 " for Step & Direction or " 2 " for Master/Slave. With gear factor and gear divider one can scale the slave to the master drive (Electronic Gear), that can modified during online processing. If you turn the control word to 0x0F and mode of operation to - 4, the controller is already configured and the slaves act according to the master encoder increments.

TOP

Serial Interface

The protocol of serial communication at the ECOSTEP (RS232, RS485, CANopen) is leant on the CANopen-Standard DS301. The datatransfer is the same for all this interfaces therefore we will show it just once. Just the transport protocol parameter are different. For the RS232 we have 9600 Baud, 8 Databit, no parity, 1 stop bit. Before the slave answers the master gets always an echo. At the dataprotocol the low bytes are sent first. In the example below the host PLC sets the velocity to 85000 [inc/ 64 s] to find the limit switch at homing mode :



TOP

CANopen Interface

Connector X1, close up resistor 150 Ohm at the end of the bus between pin 2 and 7:

	Master	ECOSTEP-Slave 1	ECOSTEP-Slave 2
	2 CAN_L	2	2
	7 CAN_H	7	7
		6 GND	6 GND
		9 CAN_V+ (8 - 24 VDC)	9 CAN_V+ (8 - 24 VDC)

At CANopen we find two ways of data transmissions. The first is the serial data object, SDO, which is leant on the standard DS301. At this type of data transmission the slave tells the host that it has recieved the message (like RS232 but faster, see baudrate object.

The other type is also very common when the communication is very fast and the receive of the message need not to be echod. Ths type is called processdataobject, PDO. Within a PDO one can transmit 8 byte data, which means 1, 2 up to 8 objects of the object catalogue according to their size. There are listener- and talker - PDO. By configuring the message - ID and the cycle time of transmission a PDO is configured.

It is very usefully that the slaves could send message to each other without a master if the message - ID's are known by transmitter and receiver. In the example below in the message - ID 182 it is sent every 10 ms the status word of that slave. It is important to know which message - ID belongs to which slave. In bigger nets the master configures the message -ID's. At nets with less than 15 slaves there are default - ID's according to the value of the DIP - switches of the slaves. For example : Slave (DIP - Switch - ID 1) has the message - ID 181 and slave (DIP - Switch - ID 2) has message - ID 182.

```
                 Communication Parameter
RX-PDO 1 [3] = { 0x00000201, 0xFF,       0 }
RX-PDO 2 [3] = { 0x00000301, 0xFF,       0 }
RX-PDO 3 [3] = { 0x80000000, 0xFF,       0 }
RX-PDO 4 [3] = { 0x80000000, 0xFF,       0 }
RX-PDO 5 [3] = { 0x80000000, 0xFF,       0 }
RX-PDO 6 [3] = { 0x80000000, 0xFF,       0 }
RX-PDO 7 [3] = { 0x80000000, 0xFF,       0 }
RX-PDO 8 [3] = { 0x80000000, 0xFF,       0 }

TX-PDO 1 [3] = { 0x00000181, 0xFF,    1000 }
TX-PDO 2 [3] = { 0x00000281, 0xFF,    1000 }
TX-PDO 3 [3] = { 0x80000000, 0xFF,    1000 }
TX-PDO 4 [3] = { 0x80000000, 0xFF,    1000 }
TX-PDO 5 [3] = { 0x80000000, 0xFF,    1000 }
TX-PDO 6 [3] = { 0x80000000, 0xFF,    1000 }
TX-PDO 7 [3] = { 0x80000000, 0xFF,    1000 }
TX-PDO 8 [3] = { 0x80000000, 0xFF,    1000 }

Sync: 0x0080   Guard: 0x0701   Emerg: 0x0081
```

```
                           TX-PDO Mapping

TX-PDO 1 [2] = { 60410010, 60400010, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 2 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 3 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 4 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 5 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 6 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 7 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
TX-PDO 8 [0] = { ????????, ????????, ????????, ????????,
                 ????????, ????????, ????????, ???????? }
```

TOP

Programmable Input / Output

At the ECOSTEP one can call small subroutines according to digital signals at the inputs at X3, so called inputs events. The 2 free configurable output one can programm like below
(.\Device Configuration\Digital Output Configuration):

```
Digital Output Configuration

Output 1: Mapping    60410010
          Offset     00000000
          AndMask    00004437
          CmpMask    00004437
          Modus      00000000
          TempReg    00000000
          Value      00004437

Output 2: Mapping    60410010
          Offset     00000000
          AndMask    0000D437
          CmpMask    0000D437
          Modus      00000000
          TempReg    00000000
          Value      00004437
```

The value of the comparator of the outputs could be described according the following formula:

Value = modus { ((value_mapping + offset)& AndMask)& CmpMask }

modus = 1 means negation, 0 no change

At the example output OUT1 turns "high" if the status word (6041,00) says, the drive found the commutation and the reference. Actually the value says that the drive only found the commutation. This will be discussed in the next chapter. The address of the status word "0x60410010" is mapped to the lower bytes of the address of the output_target_object_address 0x21600120. The code 0xC037 tells us that the controller is in a specific status (Bitcode : 1110 1100 0000 0011).

Output 2 (OUT2): Another status word comparison. It turns "high", if the target of the controller task is reached and the "SET-Point" of the control word is also set.

Appendix D - Offline Control

Courses, programming the I / Os

Apart from online operation (CANopen, RS232, RS485) the motion controller ECOSTEP 200 is able to control configured sub-routines independently. So standardised as well as mixed applications are possible (e.g. configuration by RS232, sub-routine controlled by SPS).

Sequence

255 so called sequences can be deposited within the device. Each sequence has got a sequence number to be called up with. Content of one sequence can be up to eight writing accesses (allocation of value) to any writable objects of the device's object catalogue . It is important to understand, that within one controller cycle there will be excecuted just one sequence on block. In the example below within sequence called number 2 a relative positioning of 20 000 encoder increments with a certain trapezium velocity profile (acceleration 25 000, braking acceleration 25 000, velocity 833 000) is shown. The control word is set to 0x1F (should better done by the object "28400210" to prevent setting the first 4 bits in the control word all the time).

```
Sequencer Entries

Sequence Nr. 02:  Valid = 1

1.: 60400010 =>           15  (0000000FH)
2.: 60810020 =>       833000  (000CB5E8H)
3.: 60830020 =>        25000  (000061A8H)
4.: 60840020 =>        25000  (000061A8H)
5.: 607A0020 =>        20000  (00004E20H)
6.: 60400010 =>           31  (0000001FH)
7.: 21400110 =>        32771  (00008003H)
8.: 00000000 =>            0  (00000000H)
```

After calling up a sequence these writings are settled in their configured order at once, so no other sequence or online-access can push between.

The sequences can be called up by events. These events are generated by event generators. Each event can be given a sequence number, that causes the right sequence to run, if the event takes place. Because some events can happen at the same time, the sequence numbers are arranged in a waiting loop and are run one by one. Events can be e.g:

TOP

I / O Event

A puls on the input board X3 , in the example below are all 5 input events (L-H/ H-L) active, because Input Event Mask is 0x1F1F (where the low byte defines the active L-H events and the high byte the H-L events), but just the first 2 inputs are used for calling sequences:

```
Input Events

Input Event Mask:  1F1F,  Input State:  00

DIN1   L->H => Seq 0001,  H->L => Seq 0011
DIN2   L->H => Seq 0002,  H->L => Seq 0012
DIN3   L->H => Seq 0000,  H->L => Seq 0000
DIN4   L->H => Seq 0000,  H->L => Seq 0000
DIN5   L->H => Seq 0000,  H->L => Seq 0000
LIM+   L->H => Seq 0000,  H->L => Seq 0000
LIM-   L->H => Seq 0000,  H->L => Seq 0000
HOME   L->H => Seq 0000,  H->L => Seq 0000
```

TOP

Progammable Events

- the time course of a programmable timer
- TRUE as boolean result of a comparison
- after a successful positioning like in the first example Target reached
- Call by another sequence which is done by writing the sequence number into object (2118,00)

Some events can be related to the instruction if the sequence should be run once or more times, other events are able to work out from the context a sequence number depending on the situation of the inputs .

Online change of courses

What is very useful is to implement a writing access into one or more sequences and just alter the position or another main variable through the field bus. In the following sequences we wish to:

- drive to zero after calling sequence 3
- after reaching zero the drive goes immediately in master-slave mode
- change the position in sequence 3 by writing online a new value to object (2003,03)

1. Write access to target position with value 0
2. Write access to profile velocity with value 100 000
3. Write access to operation mode (1 = positioning)
4. Write access to put sequence 4 into next queue

```
Sequencer Entries

Sequence Nr. 03:  Valid = 1

1.: 607A0020 =>          0  (00000000H)
2.: 60810020 =>     100000  (000186A0H)
3.: 60600008 =>          1  (00000001H)
4.: 21180008 =>          4  (00000004H)
5.: 00000000 =>          0  (00000000H)
6.: 00000000 =>          0  (00000000H)
7.: 00000000 =>          0  (00000000H)
8.: 00000000 =>          0  (00000000H)
```

1. Write access to control word "start motion"
2. Write access to call sequence 5 if event "target reached" is true

```
Sequencer Entries

Sequence Nr. 04:  Valid = 1

1.: 60400010 =>          31  (0000001FH)
2.: 21400110 =>       32773  (00008005H)
3.: 00000000 =>           0  (00000000H)
4.: 00000000 =>           0  (00000000H)
5.: 00000000 =>           0  (00000000H)
6.: 00000000 =>           0  (00000000H)
7.: 00000000 =>           0  (00000000H)
8.: 00000000 =>           0  (00000000H)
```

1. Write access to map the master encoder input to the target velocity
2. Write access to run mode "profile velocity with position control"
3. Write access to set the control word to 15.

Master-slave is active now!

```
Sequencer Entries

Sequence Nr. 05:  Valid = 1

1.: 25090220 =>  1627324448  (60FF0020H)
2.: 60600008 =>          -4  (FFFFFFFCH)
3.: 60400010 =>          15  (0000000FH)
4.: 00000000 =>           0  (00000000H)
5.: 00000000 =>           0  (00000000H)
6.: 00000000 =>           0  (00000000H)
7.: 00000000 =>           0  (00000000H)
8.: 00000000 =>           0  (00000000H)
```

First row:
In sequence number 3 subindex 2 and 3 we have placed the write access to target position with value 2000 by writing on object (2003,02) the value 0x607A0020 and to object (2003,03) the decimal value 2000. This could be done also via all field bus.

```
Direct Object Editor

Index:         2003   Index:        2003
Subindex:         3   Subindex:        2
Value:         2000   Value:   607A0020H

Index:         1400   Index:        1400
Subindex:         2   Subindex:        3
Value:          255   Value:   00000000H
```

We see the result of this writing access. We have changed online the position in the sequence structure without loading all data of the sequences again into the drive.

```
Sequencer Entries

Sequence Nr. 03:  Valid = 1

1.: 607A0020 =>       2000  (000007D0H)
2.: 60810020 =>     100000  (000186A0H)
3.: 60600008 =>          1  (00000001H)
4.: 21180008 =>          4  (00000004H)
5.: 00000000 =>          0  (00000000H)
6.: 00000000 =>          0  (00000000H)
7.: 00000000 =>          0  (00000000H)
8.: 00000000 =>          0  (00000000H)
```

This is a great advantage in time and for the bus activity.

Appendix E - Object catalogue

We are going to review in a brief form all the important parts of the object table of the ECOSTEP. You can find the most objects in the initial explanation of the software hsio. You have direct access to the object for writing and reading by use of the *Direct Object Editor* in the hsio-software.

On the toolkit you will find also the programm rwosio2.exe, which is called by the batch programms Upload and Download. After calling upload you will find the objects and their values in the file *read.dat. The selection of the objects you want to read or write you can chose in *reg.cfg. With save.dat you save your download values on the flash of the ECOSTEP

All actions above can be done also with your own software written in JAVA, VISUAL C or DELPHI if you follow the data protocoll explained in the menu Interface. This object table should be used for those programmers. All values should be hexadecimal in the datatransfer. In the text below we express hexadecimal numbers through the standard notation 0x2F00. Index+Subindex form a data record like 0x60400010 which you can read or write (RW), read or write only (RO,WO) or map on another address (M).

Modes & Control: 0x6040 ff.

	Index	Sub	Bits	CMD	Unit	Description
	6040	00	10	RW, M	bitcode	*control word* changes status of drive =>Statemachine 0x06 – power off 0x0F – power on 0x3F – immediate absolute positioning 0x4F – relative positioning 0x2F – absolute positioning 0x0F => 0x1F – start motion
	6041	00	10	RO, M	bitcode	*Status word* shows status of drive 0x0001 – ready to be powered on 0x0008 – error detected 0x0400 – target reached 0x4000 – commutation found 0x8000 – reference found
	6060	00	08	WO, M	number	*modes of operation:* 1 – positioning with position loop 3 – velocity with position loop - 4 – position loop

						(master/slave) - 3 velocity loop (e.g. +/- 10V) 6 homing

TOP

Sensor value Object: 0x6063 ff.

	Index	Sub	Bits	CMD	Unit	Description
	6063	00	20	RO, M	inc	*actual position value*
	6069	00	20	RO, M	inc/ 64 s	*actual velocity value*
	606B	00	20	RO, M	inc/ 64 s	*velocity demand value*
	606C	00	20	RO, M	inc/ 64 s	velocity filtered actual value
	6078	00	10	RO	integer	*current demand value*
	60FD	00	20	RO	bitcode	status of the *10 digital inputs* DIN1-8, RESET, ExtEnable: 0x0201 0000 DIN1 is "high" 0x0280 0000 DIN8 (home switch) is "high" 0x02C8 0000 DIN8,7 and 4 are "high"

TOP

Target value object 0x607A ff.

	Index	Sub	Bits	CMD	Unit	Description
	607A	00	20	RW, M	inc	*target position* in operation mode 1, shift to demand position if control word starts motion
	60FC	00	20	RO, M	inc	*demand position* in operation mode 1
	6081	00	20	RW, M	inc/ 64 s	maximum *velocity* of trapezium profile in mode 1
	6083	00	20	RW, M	16 inc/s²	*acceleration* of the trapezium profile 1000 rad/s² is roughly 80 000 [inc/s²]
	6084	00	20	RW, M	16*inc/s²	*deceleration* of trapezium profile
	60FF	00	20	RW, M	inc	*target velocity* at mode 3,- 3 and - 4

6073	00	10	RW	integer	*maximum current* see glossary ->Idac
607F	00	20	RW,M	inc/ 64 s	maximal possible *profile velocity* at mode 1 and 3 example: resolution 8000 inc 1000 rpm are 8 533 333 [inc/ 64 s]

TOP

Performance object 0x6065 ff.

Index	Sub	Bits	CMD	Unit	Description
6065	00	20	RW, M	inc	***maximum following error* at which the drive switch on an error** **2000 default value** **60 good tuned drive**
6067	00	20	RW, M	inc	*position window* for "target reached flag" - default is 10
607D	01	20	RW, M	inc	*minimum software endpositon*
607D	02	20	RW, M	inc	*maximum Software endposition* - if both are zero they're not active

TOP

Homing 0x6098 ff.

Index	Sub	Bits	CMD	Unit	Description
6098	00	08	RW, M	integer	***methods:* important homing methods from 1 to 34** **34 put the zero at the actual position**
6099	01	20	RW, M	inc/ 64 s	*velocity for searching limit switch*
6099	02	20	RW, M	inc/ 64 s	*velocity for searching zero*
609A	00	20	RW, M	16*inc/s²	*acceleration*
607C	00	20	RW, M	inc	*home offset*

TOP

Commutation Object: 0x60F6

Index	Sub	Bits	CMD	Unit	Description
60F6	01	20	RW, M	integer	Number of increments per *pole length* e.g. 50 poles and 8000 inc resolution makes 160
60F6	02	10	RW, M	integer	*number of poles* if value above is encoder increments per revolution - otherwise 0
60F6	03	10	RW, M	integer	*phase offset of current angle* is proportional to the velocity, values between 80 and 400
60F6	05	10	RW, M	integer	*maximum phase offset of current angle* , a quarter of pole length [inc]
60F6	06	10	RW, M	integer	*current peak value* to find the commutation
60F6	07	10	RW, M	integer	*time delay* at commutation method 1 and 3 500: small load 1- 5 times the motor inertia 1000: high load 20 - 50 times the motor inertia
60F6	09	10	RW, M	integer	*commutation method* 3: horizontal applications 1: weight compensation, vertical applications
60F6	11	10	RW, M	12 A/2047	$i^2 \cdot t$ current limit in time period 0 - 2047: effective current range
60F6	12	10	RW, M	[s]	*time period* to measure the effective current 0 - 12000: time constant of all motors are near 20 min 10: at small values safety function e.g. *driven on block* if the following error couldn't come

TOP

Velocity Loop Object: 0x60F9

Index	Sub	Bits	CMD	Unit	Description
60F9	01	10	RW, M	inc/s	*vc_kp* proportional value of velocity loop 50: soft gain 200: strong gain
60F9	02	10	RW, M	integer	*vc_ki* integral value of velocity loop 0: no correction of transient

	Index	Sub	Bits	CMD	Unit	Description
						deviations 1 default 2 strong correction, can cause oszillations
	60F9	03	10	RW, M	integer	*vc_kilim* limit value of vc_ki, vc_kilim should be 10% higher than the static digital friction current
	60F9	04	10	RW, M	integer	***efilt* digital input filter - velocity loop gain is then efilt • vc_kp** **1 default, no lead compensation** **2 stronger lead compensation** **3 - 5 heavy load - low pass filter**
	60F9	05	10	RW, M	integer	*ofilt* digital output filter for the velocity loop 2 - 5 1-10 times load/motor inertia 15 - 45 10 - 100 times load/motor inertia see Bode Analyse Applet

TOP

Position Loop Object: 0x60FB

	Index	Sub	Bits	CMD	Unit	Description
	60FB	01	10	RW, M	unsigned	*pc_kp* proportional value of position loop 1000 default, soft correction 3000 necessary for middle performance 8000 good performance in low following error - high position stiffness
	60FB	02	20	RW, M	unsigned	*pc_amax* prevents oscillations cause to saturation effects in the motor value = M0 / (load-inertia • 16 • 2 • pi) • encoder-resolution
	60FB	03	10	RW, M	integer	*pc_vfff* feed forward velocity for higher dynamic 0 - 3000 smooth starting 12000 - 16384 dynamic motion, low following error, important for master-slave
	60FB	04	10	RW, M	integer	pc_afff feed forward current for strong dynamic value = 12A / 2^23 • profile_acceleration

TOP

Save Objekt: 0x1010 ff.

	Index	Sub	Bits	CMD	Unit	Description
	1010	01	10	RW	logic	Store of all parameter - value 65766173
	1010	02	10	RW	logic	Store communication parameter - value 65766173
	1010	03	10	RW	logic	Store application parameter - value 65766173
	1010	04	10	RW	logic	Store offline programm flow - value 65766173

TOP

CAN-PDO Objects: 0x1400-0x1A00

0x1400-7 (rx_parameter / read)
0x1600-7 (rx_mapping)
0x1800-7 (tx_parameter / write)
0x1A00-7 (tx_mapping)

	Index	Sub	Bits	CMD	Unit	Description
	1400	01	20	RW	unsigned	*Identifier* for *CAN-Read-PDO*, Default (201) for ID = 1
	1400	02	08	RW	unsigned	*chose of listening* e.g. 0xFF := Listen if PDO-value changed 0xXX < 0xFF := Listen at every XX-times SYNC-event
	1400	03	10	RW	0.1 ms	*maximum cycle time* to listen
	1600	00	08	RW	unsigned	value stands for *number of objects* - attention one PDO can carry up to 8 byte = 0x40 bit
	1600	01	20	RW	unsigned	*first object* to be read, e.g. the status word 0x60410010
	1600	07	20	RW	unsigned	8th object to be read within one PDO - just possible if all the other objects just have 8 bit length
	1800	01	20	RW	unsigned	*Identifier* for *CAN-Write-PDO*, default 181 for ID = 1
	1800	02	08	RW	unsigned	*Choice of transmission type* e.g.. 0xFF := Write if PDO-value has changed 0xXX < 0xFF := Write at every XX-

						times SYNC-event
	1800	03	10	RW	0.1 ms	*maximal cycle time for next writing*
	1A00	00	08	RW	unsigned	value stands for *number of objects* - attention one PDO can carry up to 8 byte = 0x40 bit
	1A00	01	20	RW	unsigned	*first object* to be read, e.g. the control word 0x60400010
	1A00	07	20	RW	unsigned	8th object to be write within one PDO - just possible if all the other objects just have 8 bit length
	1F80	00	20	RW	unsigned	who is boot-master ? 0 SPS/ PLCis boot-master 3 ECOSTEP is boot-master

TOP

Sequenz Objektregion: 0x2000-FF, 0x2120, 0x2121

	Index	Sub	Bits	CMD	Unit	Description
	2000	01	08	RW	logic	*activation* - 20XX mentiones sequence 0xXX 0 sequence is not active 1 sequence is active
	2000	02	20	RW	unsigned	*target address* of object 0x20000220 0x607A0020 the first object in sequence 0 corperates the target position
	2000	03	20	RW	unsigned	*value* of previous object at 0x20000220 in sequence 0 0x20000320 0x00001F40 The value of the tarhet position is now 8000 inc

You have up to 8 target mappings in one sequence XX (20XX*02*20 => 20XX*03*20) up to (20XX*16* => 20XX*17*20).

If one sequence is called, than all the constructed mappings are processed one after the other in the drive.

	2120	01	08	RW	unsigned	sequence, activated by event *DIN1 L-H* 0x0020 sequence 0x20 are called
	2120	09	08	RW	unsigned	like above for event *DIN1 H-L* 0x0010 sequence 0x10 is called
	2120	08	08	RW	unsigned	like above for event *DIN8 L-H*

	2120	10	08	RW	unsigned	like above for event *DIN8 H-L*

TOP

(BCD-coding)

0x 2120 06 08 = > 0x0*F* 00 is another example of coding the sequence number to be called. In the example the logical sum of DIN1- 4 equals the sequence number to be called at that time point where a strobe pulse activates DIN5 (*F*).

	Index	Sub	Bits	CMD	Unit	Description
	2121	00	10	RW	unsigned	*mask that activates the inputs* 0x1F1F DIN1- 5 L-H and H-L are active 0x010F DIN1 L-H and DIN 1- 5 H-L are active
	2118	00	08	RW	unsigned	direct call of sequence number, used in programming 0x21180008 => 0x20 *sequence 0x20 are called*

TOP

Timer Object: 0x2130

	Index	Sub	Bits	CMD	Unit	Description
	2130	01	10	RW	80XX	number of sequence that starts after a waiting time 0x8012 starts sequence 0x12
	2130	02	20	RW	[ms]	waiting period 0x21300110 = > 0x8012 0x21300220 = > 0x03E8 starts sequence 0x12 after 1 s

TOP

Event Object: 0x2140

	Index	Sub	Bits	CMD	Unit	Description
	2140	01	10	RW	80XX	sequence 0xXX starts after *target reached flag*
	2140	02	10	RW	80XX	sequence 0xXX starts after *reference found flag*
	2140	09	10	RW	80XX	sequence 0xXX starts after *switch on*

	Index	Sub	Bits	CMD	Unit	Description
						disable flag, ready output is low!
	2140	0A	10	RW	80XX	sequence 0xXX starts after *ready to switch on flag*, ready output is high!
	2140	0B	10	RW	80XX	sequence 0xXX starts after *Switched on flag*
	2140	0C	10	RW	80XX	sequence 0xXX starts after *Operation enable flag* 0x20020120 = > 0x01 0x20020220 = > 0x60600008 (modes of operation) 0x20020320 = > 0x06 0x20020420 = > 0x28400210 (control word.or) 0x20020520 = > 0x0010 0x21400C10 = > 0x8002 We've just programmed a homing start sequence, that starts after *Operation enable flag* see also =>
	2140	10	10	RW	80XX	sequence starts after *hardware enable L = >H*

TOP

Output Object: 0x2160 (OUT1), 0x2161 (OUT2)

	Index	Sub	Bits	CMD	Unit	Description
	2160	01	20	RW	unsigned	object address that is mapped to *Output 1*
	2160	02	20	RW	unsigned	*offset value* will be added to the value of he mapped_output1_object
	2160	03	20	RW	unsigned	*and_value*, is the value that is compared "and_boolean" to the result of the previous operation
	2160	04	20	RW	unsigned	*Compare_value* 0x21600120 = > 0x60410010 (status word) 0x21600320 = > 0xC037 0x21600420 = > 0xC037 Output 1 goes high if the drive found commutation and reference and is in operation. Offset is not often used.

TOP

Limit Switch Object: 0x2170, 0x2171 (Limit+), 0x2172 (Limit-)

	Index	Sub	Bits	CMD	Unit	Description

	2170	00	08	RW	logic	changes polarity of the 8 digital inputs 0x21700008 = > 0x60 (DIN6+7 are low active)
	2171	02	08	RW	logic	*boolean and_value* for the positive limit switch (DIN6) 0x21710208 = > 0x20 (DIN6 is high if the input polarity is 0x00)
	2171	03	08	RW	logic	*boolean compare_value* of the positiv limit switch(DIN6) 0x21710208 = > 0x20 proofs whether DIN6 is high 0x21710308 = > 0x20 default DIN6 (0x40 default DIN7) Mechanismus
	2171	04	08	RW	logic	*status* of positiv limit switch 0x00 not high 0x01 high, no further motion in (+) direction possible

TOP

Comparison Objects: 0x2180-3 Comparator 1 to 4.

If once the comparison is TRUE, the whole comparator has to be activated again!

	Index	Sub	Bits	CMD	Unit	Description
	2180	01	20	RW	unsigned	object mapped to be compared *cmp_object*
	2180	02	20	RW	unsigned	*offset* to be added to the cmp_object
	2180	03	20	RW	unsigned	*and_value* for boolean operation with cmp-object, default 0xFFFF FFFF
	2180	04	20	RW	unsigned	*cmp_value* which is compared to cmp_object according to next operator
	2180	05	10	RW	unsigned	choice of operation: 0x0001 = *equal* 0x0002 < *smaller* 0x0003 <= *equal or smaller* 0x0004 > *bigger* 0x0005 >= *bigger or equal* 0x0006 <> *not equal*
	2180	06	10	RW	0x80XX	*sequence* 0xXX startet after TRUE status of comparison 0x21800120 = > *0x606C0020*

					(velocity) 0x21800420 = > *0x00823555* (1000 rpm) 0x21800510 = > *0x0005* 0x21800610 = > *0x8012* called once
2180	07	20	RW,M	unsigned	mappable temporary storage address
2180	08	20	RO	unsigned	value is 1 if comparison is TRUE otherwise 0

TOP

Counter Object: 0x2190 (Counter 1) bis 0x2193 (Counter 4)

Index	Sub	Bits	CMD	Unit	Description
2190	01	20	RW	unsigned	value *added* to counter
2190	02	20	RW	unsigned	*counter value*

TOP

Arithmetic Object: 0x21A0.

Index	Sub	Bits	CMD	Unit	Description
21A0	01	20	RW	unsigned	*source object* that should be modified
21A0	02	20	RW	unsigned	*destination object* that gets the result of the operation
21A0	03	20	RW	unsigned	*number* that is operated with the source object
21A0	04	10	RW	unsigned	choice of operation: 0x0000 *copy* 0x0001 + 0x0002 - 0x0003 * 0x0004 /
21A0	05	20	R	0x80XX	*result* of operation 0x21A00120 = > *0x2D010020* (value out of table[1]) 0x21A00320 = > *0x00000002* 0x21A00410 = > *0x0003* (*) 0x21A00220 = > *0x607A0020* target position Value from table[1] is multiplied by 2 and is copied into target position as destination.

TOP

Table object: 0x21B0

Used to write values into the internal table 0x2D00-FF.

Index	Sub	Bits	CMD	Unit	Description
21B0	01	20	RW	unsigned	*source object* which values should be put into the table.
21B0	02	08	RW	unsigned	*write order*
21B0	03	08	RW	integer	*position in the table* 0x21B00120 = > *0x60630020* actual position 0x21B00308 = > *0xFF* table position 255 0x21B00208 = > *0x01* write order the actual position is written into table position 255 - used for teach-in!!

TOP

Capture Object: 0x21C0

Used to strobe actual positions.

Index	Sub	Bits	CMD	Unit	Description
21C0	01	10	RW	80XX	*jump* starts sequence 0xXX after 21C0,02 goes from 0 to 1
21C0	02	20	RW	integer	*counter* if there is a L-H event on the N-limpuls of the master encoder the counter is incremented (=+)
21C0	03	20	RW	integer	*result* - if event 21C0,02 takes place the actual position value is copied into this address
21C0	04	20	RW	integer	*strobe* contains the actual position if object 21C0,02 goes from 0 to 1

TOP

Recording Object: 0x2201 ff.

One can specify and record up to 4 objects into arrays of size 1000 with a minimum time step of one ms.

Index	Sub	Bits	CMD	Unit	Description
2201	01	20	RW	unsigned	first recording object

2203	01	20	RO	unsigned	array with recorded values of previous object
2201	03	20	RW	unsigned	second recording object
2203	02	20	RO	unsigned	array with recorded values of previous object
2201	05	20	RW	unsigned	third recording object
2203	03	20	RO	unsigned	array with recorded values of previous object
2201	07	20	RW	unsigned	fourth recording object
2203	04	20	RO	unsigned	array with recorded values of previous object
2210	00	10	RW	unsigned	*counter size* specifies how big the array is
2211	00	10	RW	unsigned	*position* in the recorded array
2214	00	10	RW	unsigned	*time resolution* of the recorded arrays

recording example: actual velocity

0x22010120	= > *0x606C0020* actual velocity
0x22140010	= > *0x0005* time resolution 5 ms
0x22100010	= > *0x01F4* starts recording of 500 values

reading out of the recorded array:

0x22110010	= > *0x01F4* first recorded actual velocity value
0x22030120	*read value* v[t = 0.005]
0x22110010	= > *0x0001* last recorded value
0x22030120	*read value* v[t = 2.500]

TOP

Monitor Object: 0x2400-1

The ECOSTEP can output two independent analog monitors, each monitor is mappable to any internal value. The output range is 0...5 V, zero is represented by 2.5 V. The scaling formula is:

[V / dimension] = 1 V x internal dimension x *factor* / (256^(1 + *preshift*)) / 120

Index	Sub	Bits	CMD	Unit	Description
2400	01	20	RW	unsigned	Objekt, das auf *MON1* gemappt wird
2400	02	08	RW	unsigned	*preshift* je nach Wertedimension 0, 1 oder 2
2400	03	10	RW	unsigned	*Faktor* normal zwischen 0x0001 - 0x7FFF

Beispiel: *MON2 ist der aktuelle Stromwert*	
0x24010120	= > *0x60780010* aktueller Stromwert
0x240102008	= > *0x0000* preshift ist 0
0x24010310	= > *0x001E* Faktor ist 30
mit der *internen Unit* des digitalen Stromwertes 2047/12A erhalten wir eine Auflösung von 0.166 V/A an MON2	

TOP

Analog Port: 0x2508

The analog port AIN+ and AIN- could be mapped to every internal object (RW), mainly it is the velocity or the limit of the current value. The resolutionis -512 ... 512 DAC.

Index	Sub	Bits	CMD	Unit	Description
2508	01	20	RW	unsigned	*target_object*
2508	02	10	RW	unsigned	*factor* according to formula (glossar): maximum value/ internal unit / 2^shift / 512
2508	03	08	RW	unsigned	*Shift* often 0, 1, 2 or 3

example: *+/-10 V input for velocity loop* with max. velocity of 1500 rpm	
0x25080120	= > *0x60FF0020* target velocity
0x25080210	= > *0x03* shift is 3
0x24010310	= > *0x0C35* factor is 3125
with the *internal dimension factor* of 60/8000/64 to return "rpm" to the base of "64 inc/s" we get the factor through 1500 / 60*8000*64 / 2^3 / 512	

TOP

Master-Slave Object: 0x2509

This is the right object to specify the target object for entries from connector X7. In case of Master-Slave or Step/ Direction the velocity mapping is mainly used.

Index	Sub	Bits	CMD	Unit	Description
2509	02	20	RW	unsigned	*velocity_mapping* to (60FF,00)
2509	03	10	RW	unsigned	*gear_factor*
2509	04	10	RW	unsigned	*gear_divider*
2509	05	08	RW	unsigned	*mode* 0, 1 is 4 times Decoding 2 is step/ direction
					master position value one can write and read -

	2509	06	20	RW	unsigned	essential for phase synchronous motion
	2509	07	20	RW	unsigned	*slave position value*

example: *electronic gear box*

0x25090220	= > *0x60FF0020* target velocity
0x25090310	= > *0x07D0* factor is 2000
0x25090310	= > *0x03E8* divider is 1000
0x25090410	= > *0x00*

The *slave* runs double as fast as the master without having less good performance. It is possible to change the gear during operation according to a comparator or external analog input to get an electronic disc.

TOP

Joystick Object: 0x250A

Dieses Objekt ist in dieser Dokumentation noch nicht erläutert. Die Idee besteht darin, eine nichtlineare Kennlinie einzugeben, die die analogen Spannungssignale mit den digitalen Sollwerten eines Objektes verknüpft. Mehr Info auf Anfrage.

	Index	Sub	Bits	CMD	Unit	Description
	250A	02	20	RW	unsigned	*target_object* mainly (60FF,00)
		03	10	RW	integer	*offset*
		04	10	RW	integer	*filter*
		05	10	RW	integer	*hysteresis*
		06	10	RW	integer	*plimit*
		07	10	RW	integer	*nlimit*
		08	20	RW	integer	*pwindow*
		09	20	RW	integer	*nwindow*
		10	20	RW	integer	*jdefault*
		11	20	RW	integer	*pposition*
		12	20	RW	integer	*nposition*
		13	08	RW,M	unsigned	*joy_control*
		14	08	RO,M	unsigned	*joy_status*
		15	10	RO,M	integer	*joy_val*

	16	10	RO,M	integer	*joy_output*
	17	10	RO	integer	*joy_act*
	18	10	RO	integer	*joy_last*
	19	10	RO	integer	*joy_new*

TOP

Error Object: 0x2600

Index	Sub	Bits	CMD	Unit	Description
2600	01	20	RW	logic	*mask* could be used to disable error types
2600	02	20	RW	logic	*error code*: im hsio

0x0000 0004	antivalence - encoder, e.g. distortion
0x0000 0008	encoder counting
0x0000 0010	drive temperature > 80°C
0x0000 0020	logik voltage < 18 V
0x0000 0040	bus voltage > 180 V
0x0000 0080	bus voltage < 24 V
0x0000 0100	short circuit "phase A"
0x0000 0200	short circuit "phase B"
0x0000 0400	short circuit at "Ready" or "OUT1,2" or "brake"
0x0000 1000	external Enable low, although the drive has been activated (control word -> 0x000F)
0x0000 2000	following error during operation
0x0000 4000	overspeed, encoder frequency > 4 MHz
0x0000 8000	commutation not found
0x0002 0000	i²*t has come

TOP

Boolean Control Word Operation: 0x2840

Index	Sub	Bits	CMD	Unit	Description
2840	01	10	RW	logic	bit controlled *and* operation with the actual control word
2840	02	10	RW	logic	bit controlled *or* operation with the actual control word

TOP

Bus Parameter Object: 0x2F80, 0x2F90-2

Configuration of interfaces RS485 and CANopen.

	Index	Sub	Bits	CMD	Unit	Description
	2F80	00	08	RW	unsigned	ID-address addition 0 - 127 to DIP-switch value
	2F81/82	00	08	RW	unsigned	Baud Rate objects for CAN bus 1 MBit/s (40m): 2F81 = > *0x00*, 2F82 = > *0x14* 500 kBit/s (130m): 2F81 = > *0x00*, 2F82 = > *0x1C* 250 kBit/s (270m): 2F81 = > *0x01*, 2F82 = > *0x1C* 125 kBit/s (530m): 2F81 = > *0x03*, 2F82 = > *0x1C* 50 kBit/s (1.3km): 2F81 = > *0x47*, 2F82 = > *0x2F* 20 kBit/s (3.3km): 2F81 = > *0x53*, 2F82 = > *0x2F*
	2F90	00	08	RW	unsigned	type of interface 0: Standard RS485 1: JETTER RS485
	2F91	00	08	RW	unsigned	Baudrate 0x3F: 9.6 kBaud 0x1F: 19.2 kBaud 0x0F: 38.4 kBaud
	2F92	00	08	RW	unsigned	IndexHigh 0x60: for Jetter RS485 to declare objects at high byte 0x60XX address

TOP

Device Info + Software Version object 0x2FE0

	Index	Sub	Bits	CMD	Unit	Description
	2FE0	01	20	RO	unsigned	*software version* e.g. 29
	2FE0	03	20	RO	unsigned	*date* of modification 0x0315490 means 20010128, which says 1/28/2001

Appendix F - Glossary

At this chapter we will have a brief view on terms we often use but they might be not known generally.

Mapping

The value of an object at address Subindex-Index-Bitlength is another Objekt address. This pointer concept is known at many high level programming languages. For example the whole address of the actual master position is 2509-06-20. If you want to compare the position with some specified values you "map" it at the address of the comparator target 2180-01-20. This procedure is called mapping and generates a very powerful concept. Objects capable to be mapped are signed with "M" in the object catalog.

Statemachine

The system motor plus drive we define as a system with different discrete states. From the mechanical engineering we know the state space describtion where (s(t), v(t), t) determines the whole system at any time. This is true also for our drive - motor - system, but we want to know also whether the power is turned on, whether there is an error or whether the nominal target is reached and some more things. In CANopen we've therefore a status word. Its bitcode determines a special drive status. By use of the control word one can change the states in those directions defining the statemachine. We want to describe now the following important tasks, where we could observe these statemachine:

- Switch on of the drive
- Start homing
- Making a positioning in absolute mode

Notwendig sind die folgenden Objekte:

	Object	Meaning	Address
	Status word	consisting of important flags: error, commutation, Ready, target reached, reference found, motor switched on/ off	60410010
	Control word	determines states: power on motor, Enable, Reset, Start motion, motion absolut or relative	60400010
	Operation mode	e.g. 1 Positioning , 6 Homing	60600008
	Homing	e.g. 32 means "at next index puls"	60980008
	Profile velocity	Velocity at v(t)-trapezium, unit conversion below	60810020
	Acceleration	positive slope in v(t)	60830020

	Deceleration	negative slope in v(t)	60840020
	Target position	defined in increments	607A0020

Therefore we have the following course:

	Action / Course	**Control word**	**Status word**	**Status**
	Logic on!	0x0006	0x0031	ready to get power
	Power on!	0x000f		
			0x4437	Commutation found, no error, power on
	Power off!	0x0006		
			0x4031	as above without power

We choose the homing method : Number 32 (60980008 => 0x20) and control homing.

	Action / Course	**Control word**	**Status word**	**Operation mode**	**Status**
		0x000F	0x4437	0x01	Motor on power, controller on
	Homing 32			0x06	
	Start!	0x001F			
			0xD437		Reference found
	Ready to make a positioning absolute/ relative	0x000F / 0x004F		0x01	
			0xC437		"Set point/ Start" fifth flag in control word appears as 0x1000 in the status word. Important, because the controller acts just on 0x000F => 0x001F puls

To start a motion we initiate first the value for the deceleration and the acceleration and then:

	Action	**Control-word**	**Status word**	**Velocity**	**Target position**	**Status**
		0x000F	0xC437	100 1/min = 0x000B 71B0	8000 inc	Init
	Start!	0x001F				

			0xD037			Moving
			0xD437			Target reached

With these small operation we can manage 80% of the communication. To make a RESET one puts 0x80 to the control word, checks whether the status word says 0x0031 and then we go on like at the first step.

TOP

Address

An address is specified in the following way: first the index (consisting of four hexadecimal numbers), further the subindex (consisting of two hexadecimal numbers) and at the end a two hexadecimal code for the bit length of the data. We take the status word as an example:

(Index, Subindex, Bitlength) = (6041,00,10) or 60410010

Bitcode

Bitcode means we reproduce several event states by a code (1000 1110), that could be 8 bit, 16 bit or 32 bit and convert them into hexadecimal numbers. An example is the distribution of the states high / low at the 8 digital inputs. We would code DIN6 to 8 high active and all the others low by the number (1110 0000) which is 0xE0.

TOP

Velocity internal scaling

value in rpm is 60 • internal value [inc/64 s] / 64 / encoder resolution

Analog target value

skaling the analog port:
[-10V ..10V] - > factor [internal unit] / conversion factor [internal unit / Volt] / 2^shift / 512
Example to determine the factor at constant shift:
Maximum velocity is 1440 U/min => 1440/ [60/ 8000/ 64] / 2^3 / 512 = 3000

TOP

Scaling acceleration

value in rad/s² is equal to internal value[16 inc/s²] • 16 • 2 • pi / encoder resolution, 1 g at a belt of 100 mm/rev with encoder 8000 inc is equal to 9.81 / (0.1/ 2 • pi) • 8000 / (16 • 2 • pi) = 4969 [16 inc/s²] in the drive

Digital current

digital current also called Idac is measured in units from 0 - 2047. This could be converted to Ampere by: value • maximum drive current [A] / 2047. The ECOSTEP200 could reach 12 A peak and the ECOSTEP100 roughly 7 A.

TOP

Digital friction current

digital friction current [Idac] : could be measured indirectly by averaging the current objekt 6078,00 at slow speed. This value is multiplied by 1.2 and could be used as first approximation for the limit value of the integral parameter in the velocity loop vc_kilim.

Automatic reverse mode

automatic reverse in position mode: You keep the motor in an uncritical position to have at least 45° freedom in both directions. Your installations is already done correctly. Then you switch on the the logik and fill in the following values into the right side of the Direct Object Editor (Hex-values):

Index	Subindex	Hex value
2040	01	1
2040	02	60400010
2040	03	3F
2040	04	21400B10
2040	05	8041
2041	01	1
2041	02	607A0020
2041	03	0
2041	04	21400110
2041	05	8042
2042	01	1
2042	02	607A0020
2042	03	200
2042	04	21400110
2042	05	8041
2118	0	40
switch off:		
2041	05	8000

TOP

Data storage into the drive flash

You can store the data through hsio in: (Main Menü\Device Profile DS301 \Save/Init):

```
Save / Init

Save Parameter - all
Save Parameter - Communication
Save Parameter - Drive Config
Save Parameter - Offline Config

Init Parameter - all
Init Parameter - Communication
Init Parameter - Drive Config
Init Parameter - Offline Config
```

TOP

Table of errors in hsio

You find the error in the following menu: *Main Menu\Device Configuration\Error Flags*

Error window latest version 2001

```
Error Flags

FAULT_H8SWD_BIT          0  enable
FAULT_REGLERWD_BIT       0  enable
FAULT_ENC_ERROR_BIT      0  enable
FAULT_MOTENCCAP_BIT      0  enable
FAULT_MAENCCAP_BIT       0  enable
FAULT_OVERTEMP_BIT       0  enable
FAULT_UVMESS_BIT         0  enable
FAULT_OV_ERROR_BIT       0  enable
FAULT_UV_ERROR_BIT       0  enable
FAULT_A_ERROR_BIT        0  enable
FAULT_B_ERROR_BIT        0  enable
FAULT_OUT_DIAG_BIT       0  enable
FAULT_EX_ENABLE_BIT      0  enable
FAULT_FOLLOWINGERR_BIT   1  enable
FAULT_OVERSPEEDERR_BIT   0  enable
FAULT_COMMUFINDERR_BIT   0  enable
FAULT_ABORT_CONN_BIT     0  enable
FAULT_IxIxT_BIT          0  enable
```