

English

Svenska

Driver for Yaskawa JUSP-NS600

Manual

Contents

1	Introduction	3
2	Install and update driver.....	4
2.1	Installation of driver using Internet	4
2.2	Installation of driver from disk	4
3	Connecting the operator terminal to the servo system	5
3.1	Settings in the programming tool	5
3.2	Transfer the driver to the operator terminal.....	8
3.3	Connecting the servo system.....	9
4	Addressing.....	10
4.1	Digital and analog signals	10
4.2	Example project.....	15
4.3	Recipes	18
5	Efficient communication	19
6	Troubleshooting.....	20
7	Drawings	21

1 Introduction

This manual describes how to connect the Yaskawa Sigma II servo systems with the option module JUSP-NS600 to the operator terminals, and how to install and handle the driver Yaskawa JUSP-NS600.

The functions in the operator terminals and in the programming tool for the operator terminals are described in the manual for the operator terminals and the programming tool.

For information about the servo system we refer to the manual for the current system.

2 Install and update driver

When installing the programming tool the drivers available at the time of release are installed too. A new driver can be added into the programming tool either with the programming tool using an Internet connection or from disk. A driver can be updated to a newer version similarly.

2.1 Installation of driver using Internet

To update available drivers to the latest version or to install new drivers, the function **File/Update terminal drivers/from Internet** in the programming tool can be used. All projects must be closed prior to using this function, and the computer must be able to connect to Internet. A browser is not required. When the connection is established all drivers that can be downloaded to the computer from Internet are listed. The list states the version number of available and installed drivers. Select which drivers to install, and then click **Download**. The function **Mark Newer** will mark all drivers available in a newer version than the ones installed, and the drivers not installed. Each driver is approximately 500 kb, and ready to use when the download is finished.

2.2 Installation of driver from disk

To update available drivers to the latest version or to install new drivers, the function **File/Update terminal drivers/from Disk** in the programming tool can be used. All projects must be closed prior to using this function. Select the folder with the new driver and open the mpd-file. All drivers that can be installed are listed, and the version number of available drivers and the version number of installed drivers are stated. Select which drivers to install into the programming tool, and then select **Install**. The function **Mark Newer** will mark all drivers available in a newer version than the ones installed, and the drivers not installed.

3 Connecting the operator terminal to the servo system

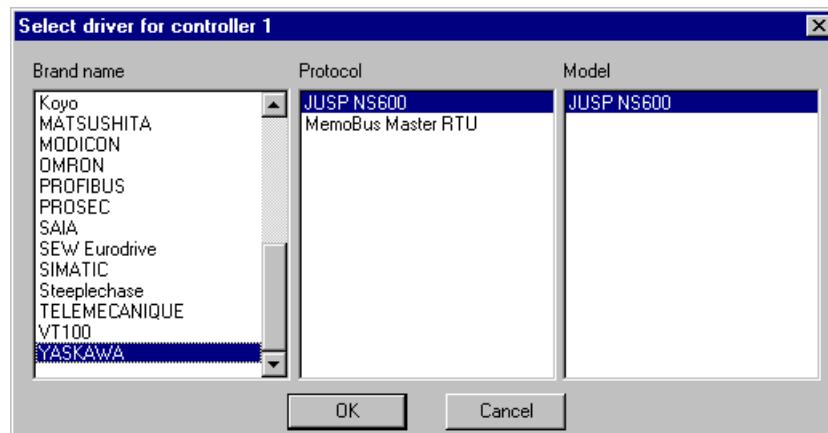
3.1 Settings in the programming tool

For communication with the Yaskawa Sigma II servo system via the JUSP-NS600 protocol the following settings must be made in the programming tool for the operator terminals.

Driver selection

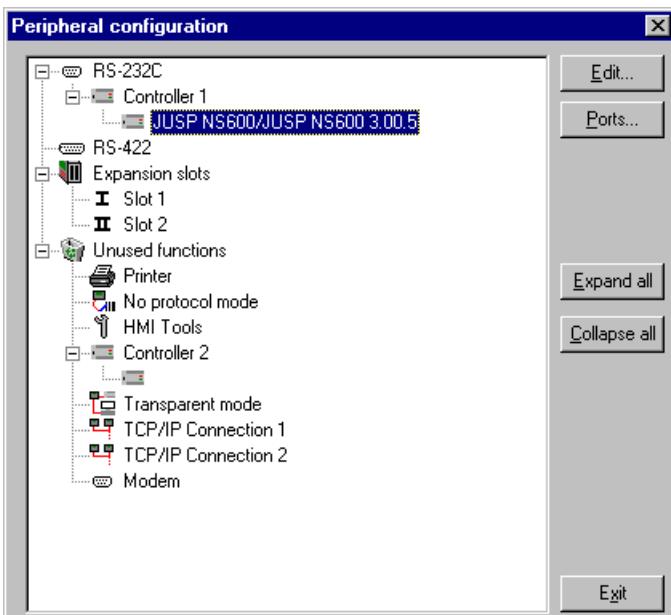
Choosing File/New creates a new project and displays the dialog **Project Settings**. In an existing project, the dialog is displayed by selecting **File/Project Settings**.

Click **Change...** under **Controller systems** to display available drivers. Choose **Brand name**, **Protocol** and **Model**, and click **OK**. Click **OK** once more to confirm the project settings.



Communication setup

The settings for the communication between the terminal and the servo system are made under **Setup/Peripherals**. To change the port the servo system is connected to, mark **Controller 1** (or **Controller 2**) and hold left mouse button down and drag to move the servo system to another communication port. Mark the selected communication port and click **Edit** to change the settings.



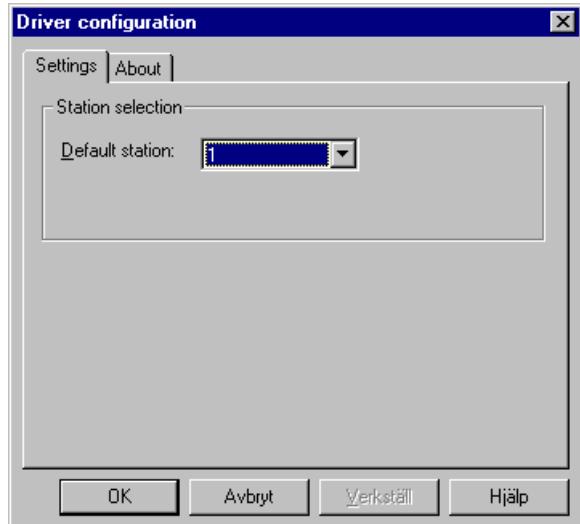
Correct settings:

Parameter	Setting
Port	RS232C or RS422
Baud rate	9 600, 19 200 or 38 400
Data bits	7
Stop bits	1
Parity	Even

Servo program requirement

No program is needed in the servo system, but parameter **Pn801** (baud rate) must be set according to the settings in the operator terminal. Since this is an offline parameter, the power to the servo system must be turned off and on again in order to enable a new setting. The parameter can be changed using the special unit JUSP-OP02A, with any terminal program, or with the operator terminal.

To make specific settings for the selected driver mark the driver name and click **Edit**.



Parameter	Description
Default station	At start up, the terminal checks communication towards the default station. It is also the station that will be addressed if no station number is stated in front of the I/O-device name.

Default station is stated under **Peripherals/Driver Configuration/Settings**. To communicate with other stations, the station number is stated as a prefix to the device.

Example:

F:SVON means servo system on in station 15 (F hex).

SVON means servo system on for the system stated as **Default station**.

16 stations can be addressed: 0-F (hex).

Commands

Commands can be stated on the command line under **Setup/System Signals** in the programming tool for the operator terminals. The commands are separated by a space.

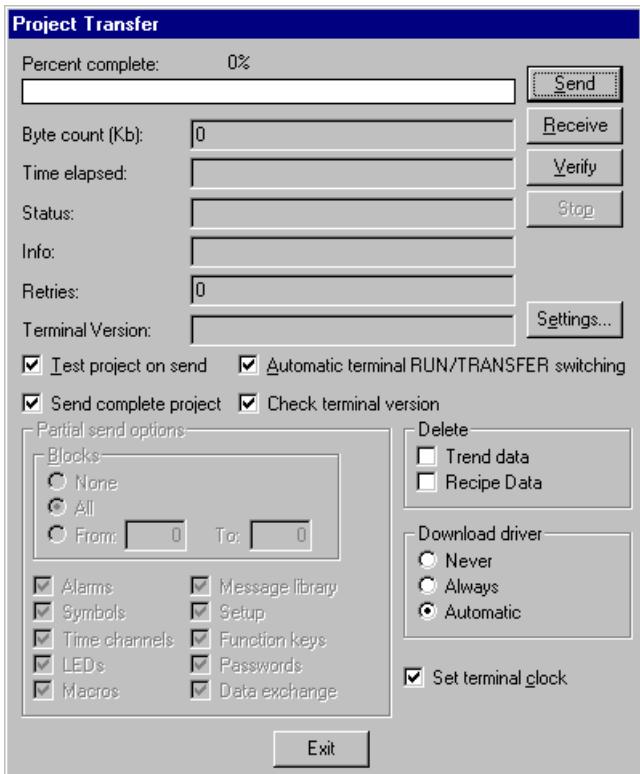
The command **Timeout (T)** is used to set the time within which the servo system has to answer before a communication error occurs. The default time-out is 2000 ms.

The command **DEBUG** is used to display instructions sent to the servo system. Create an ASCII device of 30 characters called **DEBUG** prior to using the command. Ignore the last characters displayed in the string.

Since there are a lot of devices that are only local variables in the terminal, you may have a block in your project that communicates only when you want to write something to the servo system. If communication problems occur in this case, they are not detected or shown until some device is to be written. To avoid this, the command **COMCHK** can be stated on the command line in order to always make the driver perform a communication check in every block. Beware that communication on other ports of the servo system may have some trouble, when **COMCHK** is stated on the command line.

3.2 Transfer the driver to the operator terminal

The selected driver is downloaded to the operator terminal when the project is transferred to the terminal. Select Transfer/Project.



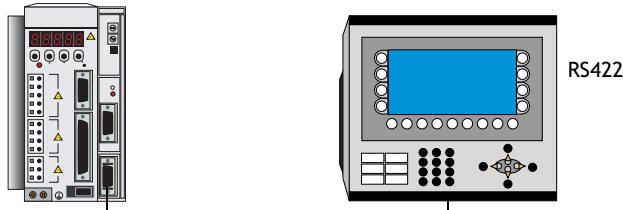
There are three alternatives when the driver is downloaded to the terminal.

Function	Description
Never	The driver is not downloaded and the existing driver in the terminal is used.
Always	The driver is downloaded every time the project is transferred.
Automatic	The driver is downloaded only if the driver in the terminal is not the same as the selected driver in the project.

3.3 Connecting the servo system

The connection can be of the type Point-to-Point via RS422 or via RS232.

Point-to-Point RS422 connection



A cable according to chapter Drawings is connected between the CN6 port on the NS600 option module on the servo system and the RS422 port on the operator terminal.

Point-to-Point RS232 connection

Two cables can be joined in order to connect the terminal and the servo system.

14 pin 3M Yaskawa NS600	9 pin D-sub female	9 pin D-sub male	9 pin D-sub female, operator terminal
2	2	2	3
4	3	3	2
14	5	5	5
	7		
	8		
	1		
	4		
	6		

Use the programming cable NS600, intended for communication between the servo system and a PC, and a cable described on the right-hand side above. Connect the two cables, and plug the 9 pin D-sub female into the RS232C port on the terminal.

Note!

The connection can also be of the type multidrop. For details see the Yaskawa manual.

Correct parameter settings in the servo system

Set the servo system parameters according to the table below.

Parameter	Parameter name	Setting	Description
Pn800	Serial Communication Protocol	1	RS422 + Echo back
Pn801	Baud rate	0	9 600
Pn802	Answer	1	OK = answer

Parameters **Pn800** and **Pn801** are offline parameters, which means that the power to the servo system must be turned off and on again in order to enable new settings. Parameter **Pn802** is an online parameter, which means that a new setting has an effect immediately.

The baud rate can be increased by changing the parameter **Pn801** and the terminal setting accordingly. Set **Pn801** to 1 for baud rate 19 200, or to 2 for baud rate 38 400.

For further information about servo system parameters and connections we refer to the manual for the servo system.

4 Addressing

4.1 Digital and analog signals

The operator terminal can handle digital signals (ON/OFF) and analog signals (values) of the types in the tables below.

See the Yaskawa manual for further information.

Types concerning the program table:

Device name	Yaskawa device	Type	Description
POSTCHOICE	(POST)	16-bit	When POST is triggered, a message is formed by the values in POSTCHOICE and POSTVALUE .
POSTVALUE	(POST)	32-bit	
POST	POST	Bit	
SPDT	SPDT	32-bit	
SPDTRIGG	(SPDT)	Bit	Sends what SPDT holds.
RDSTT	RDSTT	32-bit	
RSPDT	None	32-bit	
RSPDTRIGG	None	Bit	
POUTT	POUTT	16-bit/ASCII	Can be ASCII 5 characters long, but setting device POUTT_0 , POUTT_1 , POUTT_2 , POUTT_3 , POUTT_4 is recommended for multiple choice. See chapter 4.2.
EVENTTCHOICE	(EVENTTT)	16-bit	When EVENTTT is triggered, a message is formed by the values in EVENTCHOICE and EVENTTVALUE .
EVENTTVALUE	(EVENTTT)	32-bit	
EVENTTT	EVENTTT	Bit	
LOOPT	LOOP	32-bit	
ITNEXT	(NEXTTT)	32-bit	When this device is changed it is sent to the servo. See also POSEVENT . If ITNEXT is read as END then 0 is put in ITNEXT and ITNEXTDIG is changed to END .
ITNEXTDIG	NEXTTT	Bit	Recommended use: text0 = JMP text1 = END
ZONEPT	ZONEPT	32-bit	
ZONENT	ZONENT	32-bit	
JSPDT	JSPDT	32-bit	
POSEVENT	None	Bit	Does what both POST and EVENTTT does. Additionally, if ITNEXTDIG is set to END , then END is sent to NEXTTT in the index table to the row matching the address set on POSEVENT . Example: If ITNEXTDIG is set to JMP , nothing is sent to the NEXTTT in the index table row, since it already exists there (the value in ITNEXTTT).

Types not concerning the program table:

Device name	Yaskawa device	Type	Description
ASCIIS	None	ASCII (max 30)	Ignore station settings in project or station index. The string in ASCIIS is sent at ASCIIS-TRIGG.
ASCIISTRIGG	None	Bit	Example: 1SVON or FSVON. (SVON to station 1, SVON to station F). Mostly for debug purposes.
ASCIIR	None	ASCII (max 30)	What is last received from the servo system. Mostly for debug purposes.
ACC	ACC	32-bit	ACCTRIGG sends what ACC holds.
ACCTRIGG	(ACCA9	Bit	
DDEC	DEC	32-bit	DDECTRIGG sends what DDEC holds.
DDECTRIGG	(DEC)	Bit	
PRM	PRM	32-bit	Set PRMASCII to be 4 characters long. PRMASCII is only used for PRMO.
PRMASCII	PRM	ASCII	MULPRM is used when the value in PRM is to be used by an object with multiple choice. For example, Pn801: 9 600, 19 200, 38 400 can be shown instead of 0, 1, 2.
MULPRM	None	16-bit	
SVON	SVON	Bit	
SVOFF	SVOFF	Bit	
STARTN	STARTss	Bit	
STOP	STOP	Bit	
SKIP	SKIP	Bit	
MON	MON	32-bit	
JOGVALUE	(JOG)	32-bit	Set JOGP/JOGN momentary on function key. On press, JOGP/JOGN is sent with the value in JOGVALUE. On release, SKIP is sent. If JOGP/JOGN is set (not momentary), JOGP/JOGN is sent and the jog will not stop until the operator sends SKIP by setting the object SKIP.
JOGP	JOGP	Bit	
JOGN	JOGN	Bit	
RJOG			Not implemented
DBRMS	DBRMS	32-bit	
EVTIME	EVTIME	32-bit	
IN1	IN1	32-bit	Zero fill, bit addressing possible
IN2	IN2	32-bit	Zero fill, bit addressing possible
LOOP	LOOP	32-bit	
MTSIZE	MTSIZE	32-bit hex	
MTTYPE	MTTYPE	32-bit hex	
OUT1	OUT1	32-bit	Zero fill, bit addressing possible
OUT2	OUT2	32-bit	Zero fill, bit addressing possible
PGTYPE	PGTYPE	32-bit hex	
PGVER	PGVER	32-bit hex	
RGRMS	RGRMS	32-bit	
STIFFD	STIFFd	Bit	
SVTYPE	SVTYPE	32-bit hex	

Device name	Yaskawa device	Type	Description
SVVER	SVVER	32-bit hex	
SVYSPEC	SVYSPEC	32-bit	
TRMS	TRMS	32-bit	
TYPE	TYPE	32-bit hex	
VER	VER	32-bit hex	
YSPEC	YSPEC	32-bit	
INERTIA	INERTIA	32-bit	
RDST	RDST	32-bit	
LOOPT	LOOPT	32-bit	
TRM	TRM	32-bit	Set TRMASCII to be 4 characters long. TRMASCII is only used for TRM0.
TRMASCII	(TRM)	ASCII	
RES	RES	Bit	
ARES	ARES	Bit	
STNINDEX	None	16-bit hex	
INDEX	None	16-bit	
INTEST	IN2TEST	32-bit	Zero fill, bit addressing possible
OUTTEST	OUT2TEST	32-bit	Zero fill, bit addressing possible
ALMN	ALMN	ASCII/ 16-bit hex	If 16-bit hex is used, the alarm code or error code in the returned string is shown in this device. If there is no alarm code, 0 is shown.
ALMTRCCLR	ALMTRCCLR	Bit	
ERR	ERR	ASCII/ 16-bit hex	If 16-bit hex is used, the error code in the returned string is shown in this device. If there is no error code, 0 is shown.
POUT	POUT	16-bit/ ASCII	Can be ASCII 5 characters long but setting device POUT_0, POUT_1, POUT_2, POUT_3, POUT_4 is recommended for multiple choice. See explanation of POUTT in chapter 4.2.
PGMSTEP	PGMSTEP	ASCII	Setting PGMSTEP to be 3 characters long is recommended.
STIFF	STIFF	32-bit	
ABSPGRES	ABSPGRES	Bit	
MLTLIMSET	MLTLIMSET	Bit	
ALM	ALM	ASCII	
TUNESTORE	TUNESTORE	Bit	
CURZERO	CURZERO	Bit	
PGMINIT	PGINIT	Bit	
ZONEINIT	ZONEINIT	Bit	
JSPDINIT	JSPDINIT	Bit	
PGMSTORE	PGMSTORE	Bit	
ZONESTORE	ZONESTORE	Bit	
JSPDSTORE	JSPDSTORE	Bit	
START	START	Bit	
PGMRES	PGMRES	Bit	
SPD	SPD	32-bit	

Device name	Yaskawa device	Type	Description
RSPD	RSPD	32-bit	
POSVALUE	None	32-bit	See instructions using POSVALUE .
POSVALUE	None	32-bit	See instructions using POSVALUE .
STI	(STI)	Bit	Takes the value of POSVALUE and sends it with STI as 1STI00000100 (1 station).
ST	(ST)	Bit	Takes the value of POSVALUE and sends it with ST as 1ST00000100 (1 station).
ST2	(ST)	Bit	Sends the string 1ST to the servo system if station 1 is chosen. If the device looks like 2:ST2 , the string 2ST is sent.
RSI	RSI	Bit	Takes the value of POSVALUE and sends it with RSI as 1RSI00000100 (1 station).
RS	RS	Bit	Takes the value of POSVALUE and sends it with RS as 1ST00000100 (1 station).
POSI	(POSI)	Bit	Takes the value of POSVALUE and sends it with POSI as 1POSI00000100 (1 station).
POS	(POS)	Bit	Takes the value of POSVALUE and sends it with POS as 1POS00000100 (1 station).
HOLD	HOLD	Bit	
ZRN	ZRN	Bit	
PRMINIT	PRMINIT	Bit	
ZSETVALUE	ZSETVALUE	32-bit	
ZSET	ZSET	Bit	

The devices that are forced to 32-bit cannot be used for multiple choice, except for PRM that has a special device named MULPRM.

Note!

The formats of the devices must correspond to what is written in this document. If, for example, 16-bit is stated in the document, and 32-bit is chosen, errors could occur.

Special variables

Instructions that can be sent to, but not read from the servo system is managed by some internal variables in the operator terminal. Those devices do not necessarily reflect the status in the servo system.

Examples:

- When the digital device **SVON** (which can be written to, but not read from) is pressed, it is set to 1 and displays the text for the device that corresponds to 1 in the programming tool for the operator terminals. The servo system starts if everything is OK. If the power to the servo is turned off, the operator terminal will still show that the servo system is on, even though it is off. In this case **SVON** can be changed to 0 without problems (**SVOFF** can be sent to a servo system already turned off).
- If a servo system in operation is known to the operator terminal to be on, and power to the terminal is turned off, the terminal will show that the servo system is off. In this case **SVON** can be changed to 1 without problems (**SVON** can be sent to a servo system, already running).

Internal variables in the operator terminal

The devices ACC, ASCIIR, ASCIIS, DDEC, JOGN, JOGP, JOGVALUE, POSVALUE, POSIVALE, RSPD, SPD, STNINDEX, SVOFF, SVON and ZSETVALUE have internal variables.

Not all of the devices have one variable to each of the 16 possible stations (axis).

Example:

JOGP has one variable for each station, but only one JOGVALUE. The stations have to share the JOGVALUE. Two devices, 1:JOGP and 2:JOGP are used. When 1:JOGP is set to 1, the jog instruction with the value in JOGVALUE is sent to station 1. When 2:JOGP is set to 1, the jog instruction with the value in JOGVALUE is sent to station 2.

Only JOGN, JOGP, SVOFF and SVON have one variable for each station.

The following variables have one internal variable for each row in the index table:
POSTVALUE, POSTCHOICE, EVENTTVALUE, EVENTTCCHOICE,
ITNEXT.

Important information regarding block changes

When a block change is performed in the operator terminal, the values for the internal variables are read from the servo system.

Example:

A block change to a block that uses the variables POSTCHOICE0 and POSTVALUE0 is performed. 1POST0 is sent (axis 1).
1POST000=A+12345678[CR][LF] is received. POSTCHOICE0 is updated with the value 0, and the terminal displays A (if multiple choice is created according to chapter 4.2). POSTVALUE0 is updated with the value 12345678. If the value in POSTVALUE0 is changed, and Enter is pressed, then only the local variable in the terminal is changed. In order to write the value to the servo system, the variable POST has to be triggered.

Note!

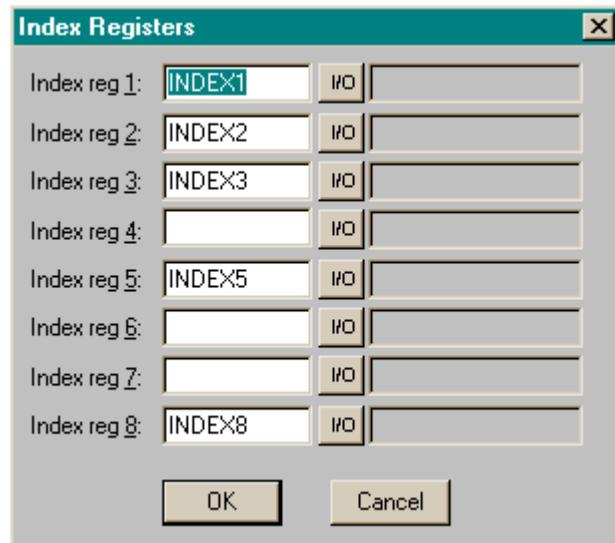
Recipe handling with this driver may have limited functionality. Recipe is only to be used with the index table, and nowhere else.

There are only 16 stations to address, and 0-127 rows in the index table.

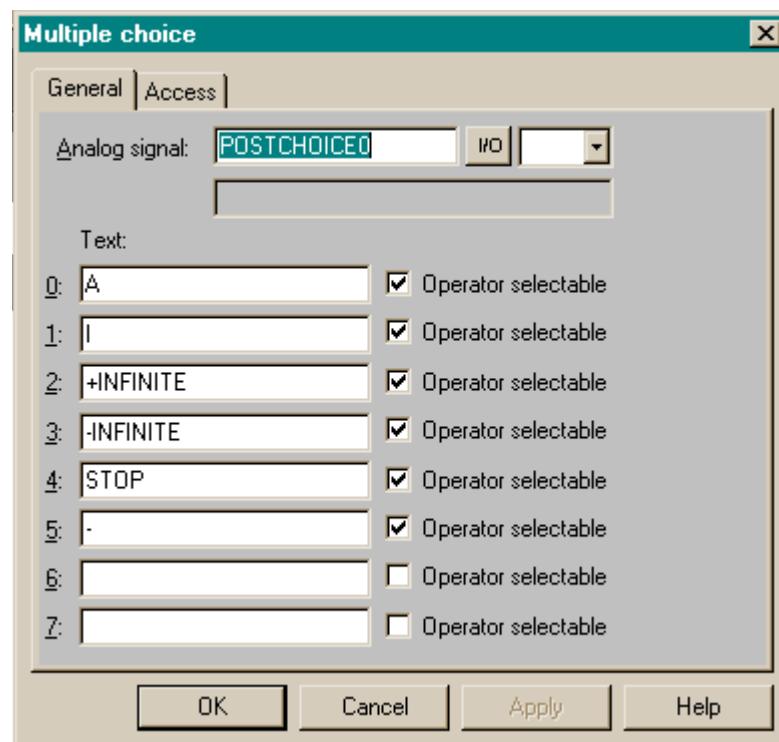
Use of arrays does not work with this driver.

4.2 Example project

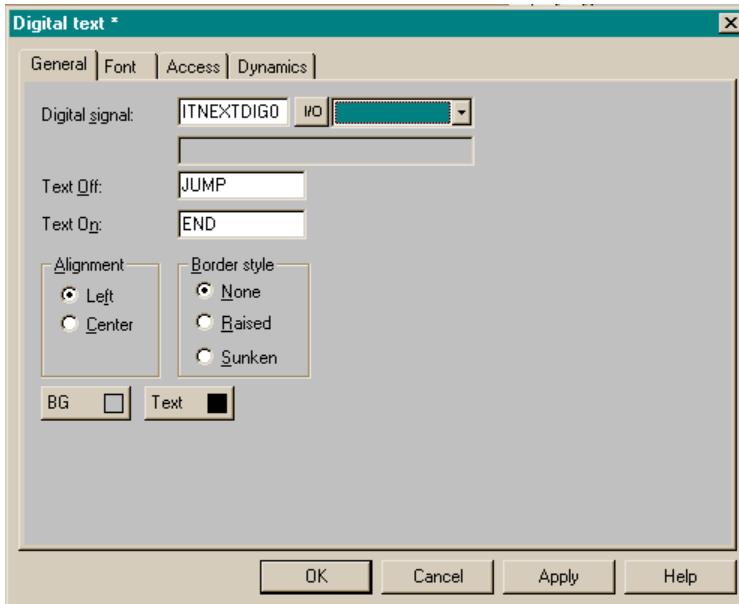
The INDEX device has eight variables, one for each index.



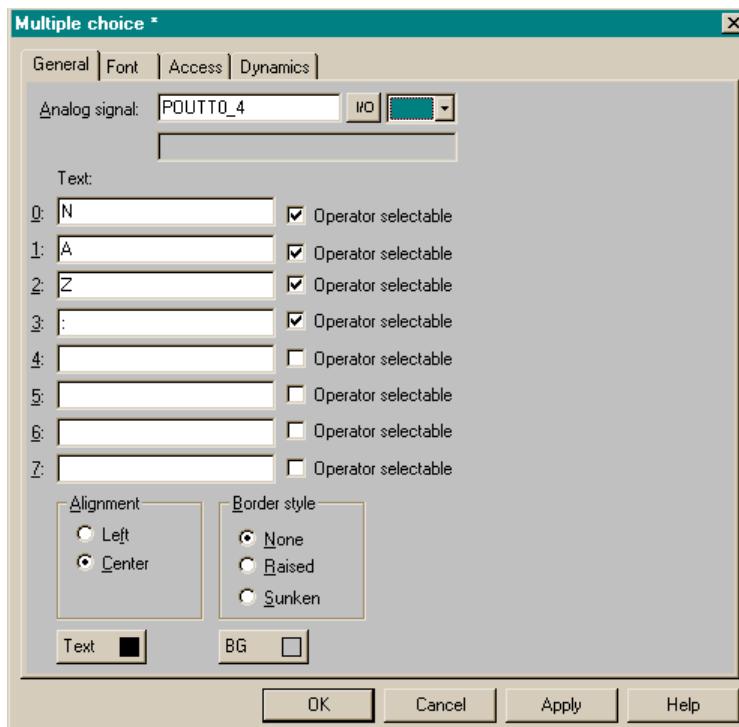
POSTCHOICE: The text has no influence on what is sent to the servo system.
Regard the texts below as suggestions.



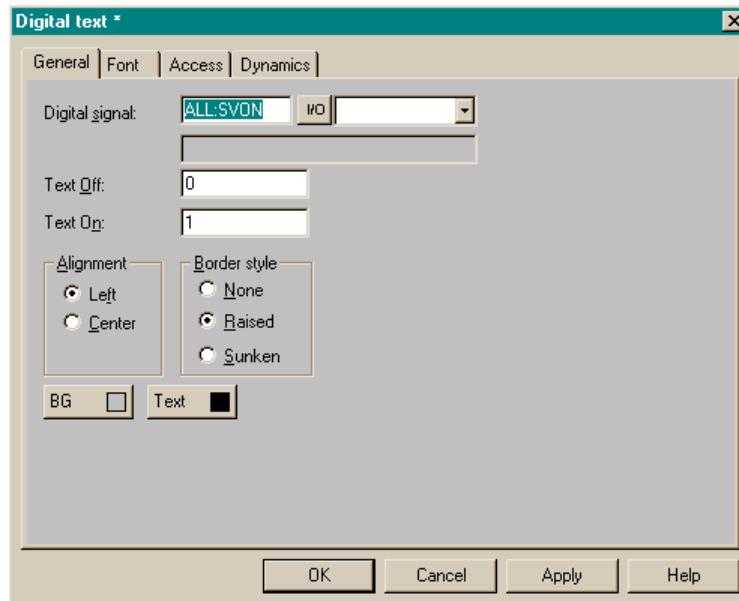
ITNEXTDIG: Text Off is used to show JMP or JUMP, and Text On to show END.



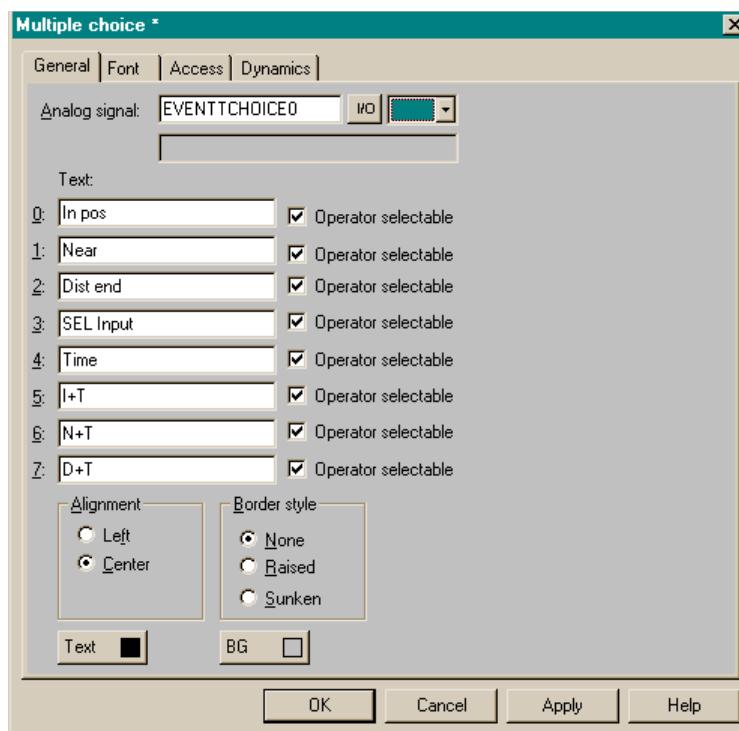
POUTT: The text has no influence on what is sent to the NS600. Regard the texts below as suggestions. The bits are addressed POUTT0_0 - POUTT0_4. POUTT0_0 is addressed for the least significant bit.



Example of a device addressed as ALL. In this case the string *SVON is sent.



EVENTCHOICE: The text has no influence on what is sent to the servo system.
Regard the texts below as suggestions.



4.3 Recipes

If POUTT is to be used in the index table and saved as a recipe, perform the instruction below:

If POUTT is used as POUTT0_0 - POUTT0_4 there must also be an object POUTT0 shown as ASCII (5 positions). This object does not have to be visible or changeable; it is just used for proper saving and loading of recipes with the POUTT device.

Recipe used in a text block

If recipe is used in a text block, the fact that a text block can be scrolled has to be considered. Under some circumstances this can be a problem:

Assume that a terminal that only displays two rows in a text block is used.

The first row displays POSTCHOICE0, the second POSTCHOICE1, and the third (that is not visible) POSTCHOICE2.

When jumping to this block, values for the displayed variables are read and updated (see section Important information regarding block changes in chapter 4.1). The value for POSTCHOICE2 is not yet read.

We now change the values for POSTCHOICE0 and POSTCHOICE1 and trigger both of them to be written to the servo system.

Then we save it as a recipe. This would be wrong, because the saved value for POSTCHOICE2 is random.

When loading this recipe, the random value is written to the internal variables, and when triggered, the random value is sent to the servo system, destroying the correct value in POST2 (the third row in the index table).

You could try changing the value in POSTCHOICE0 and then scroll down to POSTCHOICE2, so that POSTCHOICE0 disappears. Then scroll up to POSTCHOICE0 again to see that the value has been overwritten by a new reading from the servo system.

Correct mode of procedure:

When jumping to the text block, be sure to scroll to all rows to confirm that the values in the terminal match the values in the servo system.

Then change desired values, and make sure to trigger to write them to the servo system.

Save the recipe. When loading the recipe now, the correct values will be used.

As an alternative, change POSTCHOICE0 and POSTCHOICE1, and then scroll to POSTCHOICE2 and, if desired, change that too.

Then save the recipe. The value in POSTCHOICE0 is correct, because it is not destroyed until it is shown again on the display.

When reading the recipe, and triggering the values to the servo system, the recipe should be correct.

5 Efficient communication

The devices are asked for, one by one, and there is no way to pack signals. Only the baud rate can be changed to 38 400.

6 Troubleshooting

If a variable does not work as expected

If, for example, ZRN does not work as expected, perform the following steps:

Create a new block with ZRN (make it changeable) and the device DEBUG (ASCII 30 characters long).

Add the device ASCIIR (ASCII 30 characters long).

State the command DEBUG on the command line.

Send the project to the operator terminal.

Go to ZRN and press Enter. DEBUG now shows what was sent, and ASCIIR shows what was received. Maybe there is an error code that is described in the manual for the servo system.

Alternatively, a terminal program can be connected. Then check if the servo system behaves as expected.

7 Drawings

RS-232

Pin no	Name	Signal direction Terminal ↔ XXX
1	⁴⁾ +5V >200mA	↔
2	TxD	→
3	RxD	←
5	0V	
7	CTS	←
8	RTS	→
9		

4) Only for units with one serial port (RS-422) and 232-25-ADP (adapter). On all other terminals, not connected.

RS-422

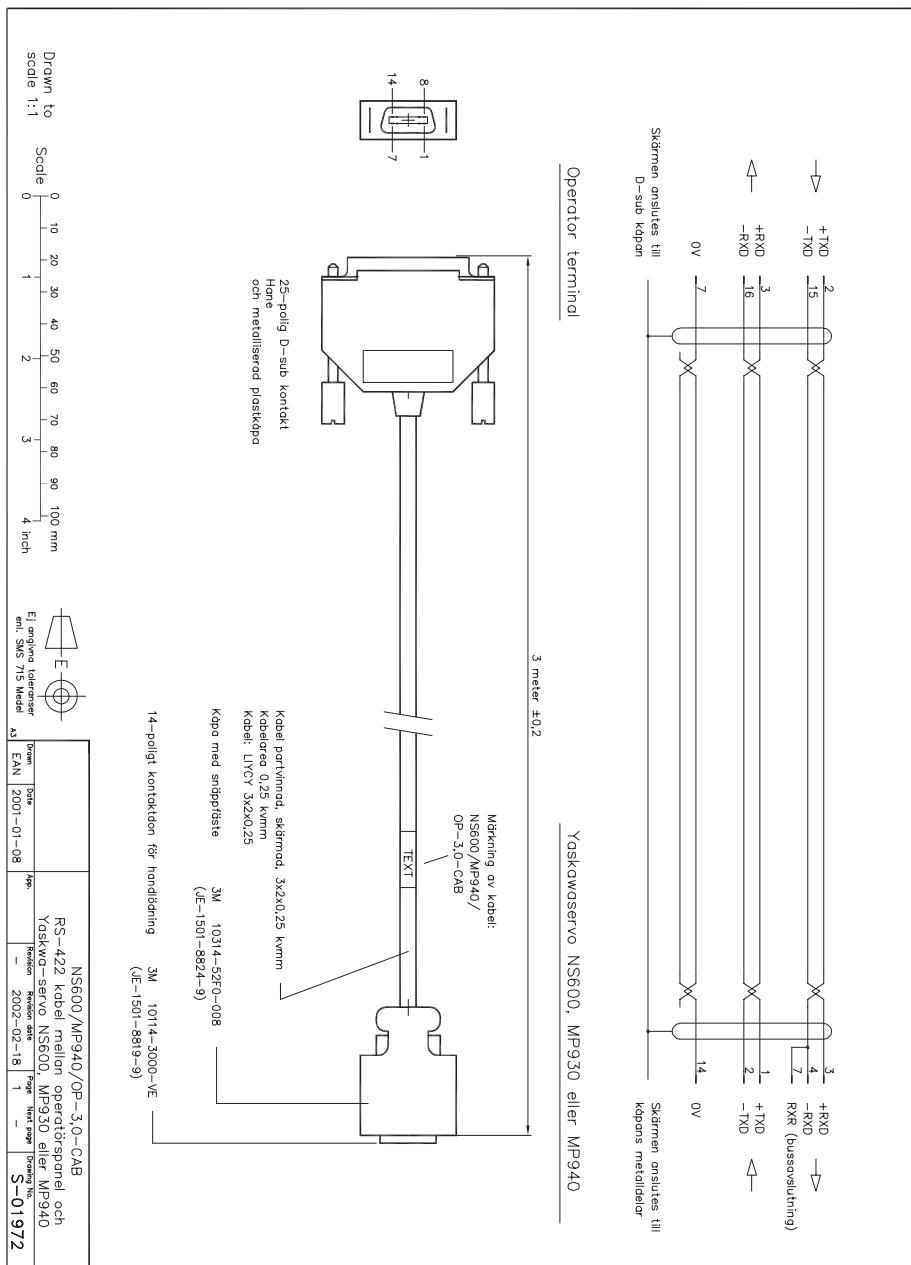
Pin no	Name	Signal direction Terminal ↔ XXX
2	+TxD	→
15	-TxD	→
3	+RxD	←
16	-RxD	←
4	+RTS	→
17	-RTS	→
5	+CTS	→
18	-CTS	→
20	1)	
21	1)	
7,8	0V	
14	+5V	→
	<500mA	
12,13	2) +5V 24,25 >200mA	→
9	³⁾ TxD	→
10	³⁾ RxD	←
22	³⁾ CTS	←
23	³⁾ RTS	→

Bld Revision 1.25 pg A3

RS-232/RS-422/RS485		
Form	4sp.	Revision
SLG	1998-12-01	Revised date
	—	Page
	2001-03-29	Next page
	1	Drawing No.
		S-02467

- 1) Pin no 20 connected to pin no 21 internal in the terminal
- 2) Only for units with 5VDC voltage feed
- 3) Only for units with one serial port (RS-422)

Cable between operator terminal and Yaskawa NS600



Innehåll

1	Introduktion.....	3
2	Installation och uppdatering av drivrutin	4
2.1	Installation av drivrutin via Internet.....	4
2.2	Installation av drivrutin från disk	4
3	Ansluta operatörsterminalen till servosystemet	5
3.1	Inställningar i programmeringsverktyget	5
3.2	Överföra drivrutinen till operatörsterminalen.....	8
3.3	Ansluta servosystemet.....	9
4	Adressering.....	10
4.1	Digitala and analoga signaler.....	10
4.2	Exempelprojekt.....	15
4.3	Recept.....	18
5	Effektiv kommunikation.....	19
6	Felsökning.....	20
7	Ritningar	21

1 Introduktion

Manualen beskriver hur Yaskawa Sigma II-servosystem med optionsmodul JUSP-NS600 ansluts till operatörsterminalerna, samt hur drivrutinen Yaskawa JUSP-NS600 installeras och hanteras.

Funktionerna i operatörsterminalerna och programmeringsverktyget för operatörsterminalerna beskrivs i manualen för operatörsterminalerna och programmeringsverktyget.

För information om servosystemet hänvisas till manualen för aktuellt system.

2 Installation och uppdatering av drivrutin

Tillgängliga drivrutiner installeras samtidigt som programmeringsverktyget installeras. En ny drivrutin kan läggas till i programmeringsverktyget antingen från Internet eller från disk. En drivrutin kan uppdateras till nyare version på samma sätt.

2.1 Installation av drivrutin via Internet

För att uppdatera befintliga drivrutiner till senaste version eller för att installera nya drivrutiner kan funktionen **Arkiv/Uppdatera drivrutiner/från Internet** användas. Alla projekt måste stängas innan funktionen används och datorn måste kunna ansluta till Internet. Någon webbläsare behövs inte. När anslutningen är etablerad visas en lista med alla drivrutiner som kan laddas ner till datorn via Internet. I listan visas versionsnummer på tillgängliga och på redan installerade drivrutiner. Markera önskade drivrutiner och välj därefter **Ladda ner**. Funktionen **Markera nyare** markerar alla drivrutiner som finns tillgängliga i en senare version, samt drivrutiner som inte är installerade. Varje drivrutin är ungefär 500 kb stor, och kan användas så snart nedladdningen avslutats.

2.2 Installation av drivrutin från disk

För att uppdatera tillgängliga drivrutiner till senaste version eller för att installera nya drivrutiner kan funktionen **Arkiv/Uppdatera drivrutin/från Disk** användas. Alla projekt måste stängas innan funktionen används. Välj den katalog som innehåller den nya drivrutinen och öppna mpd-filen. En lista visas med alla drivrutiner som kan installeras. I listan visas versionsnummer på tillgängliga drivrutiner och versionsnumret på installerade drivrutiner. Markera önskade drivrutiner och välj därefter **Ladda ner**. Funktionen **Markera nyare** markerar alla drivrutiner som finns tillgängliga i en senare version samt drivrutiner som inte är installerade.

3 Ansluta operatörsterminalen till servosystemet

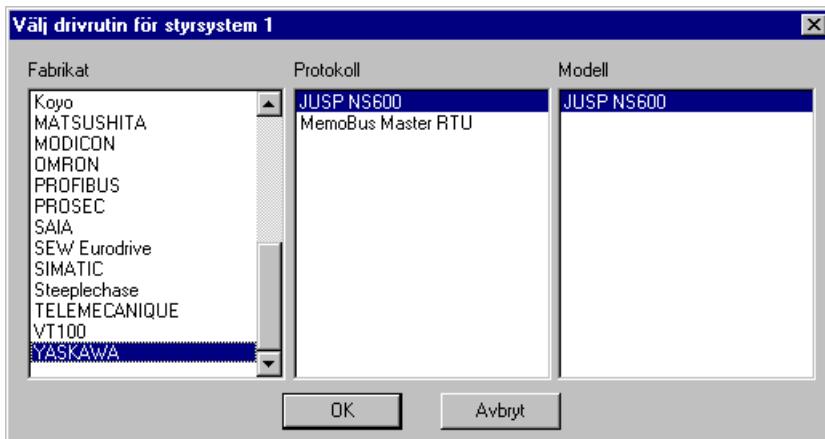
3.1 Inställningar i programmeringsverktyget

För kommunikation med servosystemet Yaskawa Sigma II via protokollet JUSP-NS600 måste följande inställningar göras i programmeringsverktyget.

Val av drivrutin

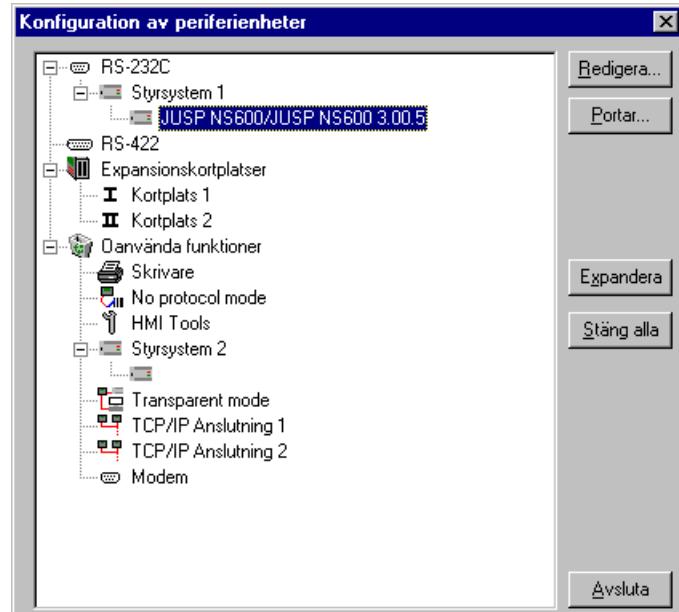
Välj Arkiv/Nytt för att skapa ett nytt projekt. Då visas dialogrutan **Projektinställningar**. I ett befintligt projekt visas dialogen genom **Arkiv/Projektinställningar**.

Klicka på **Ändra...** under **Styrsystem** och välj i listan över tillgängliga drivrutiner. Välj **Fabrikat**, **Protokoll** och **Modell**, och klicka **OK**. Klicka på **OK** igen för att bekräfta projektinställningarna.



Kommunikationsinställningar

Inställningarna för kommunikationen mellan terminalen och servosystemet anges under **Inställningar/Periferienheter**. För att ändra vilken port servosystemet är anslutet till, markera **Styrsystem 1** (eller **Styrsystem 2**) och håll vänster musknapp nere för att dra och släppa servosystemet på önskad kommunikationsport. Markera den valda kommunikationsporten och klicka på **Redigera** för att ändra inställningarna.



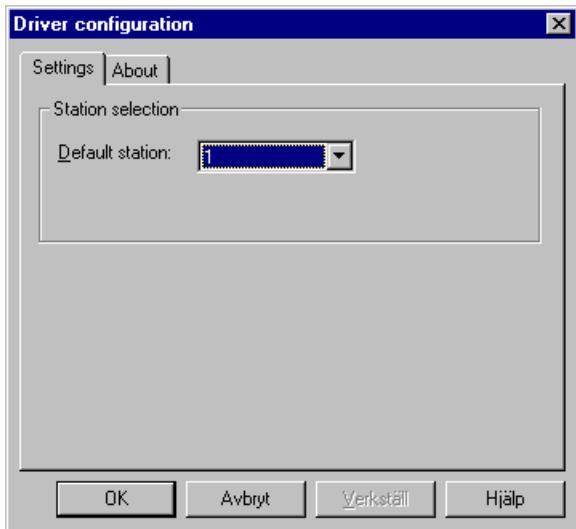
Korrekte inställningar

Parameter	Inställning
Port	RS232C eller RS422
Hastighet	9 600, 19 200 eller 38 400
Databitar	7
Stoppbitar	1
Paritet	Jämn

Krav på program i servosystemet

Det behövs inget program i servosystemet, men parameter **Pn801** (överföringshastighet) måste överensstämma med inställningarna i operatörsterminalen. Då detta är en "offline parameter", måste spänningen till servosystemet slås från och sedan till igen för att en ny inställning ska få genomslag. Parametern kan ändras genom att specialsignalen JUSP-OP02A används, med hjälp av något terminalprogram eller från operatörsterminalen.

Markera drivrutinens namn och klicka på **Redigera** för att göra inställningar för den valda drivrutinen.



Parameter	Beskrivning
Default station	Den station terminalen testar kommunikationen till vid uppstart. Det är även den station som adresseras då inget stationsnummer anges framför I/O-signalens namn vid adressering.

Under **Periferienheter/Driver Configuration/Settings** definieras parametern **Default station**. För att kommunicera med andra stationer anges stationsnumret som ett prefix till signalen.

Exempel:

F:SVON innebär servosystem on i station 15 (F hex).

SVON innebär servosystem on för systemet som angivits som **Default station**.

16 stationer kan adresseras: 0-F (hex).

Kommandon

Kommandon kan anges på kommandoraden under **Inställningar/Systemsignaler** i programmeringsverktyget för operatörsterminalerna. Kommandon åtskiljs av ett mellanslag.

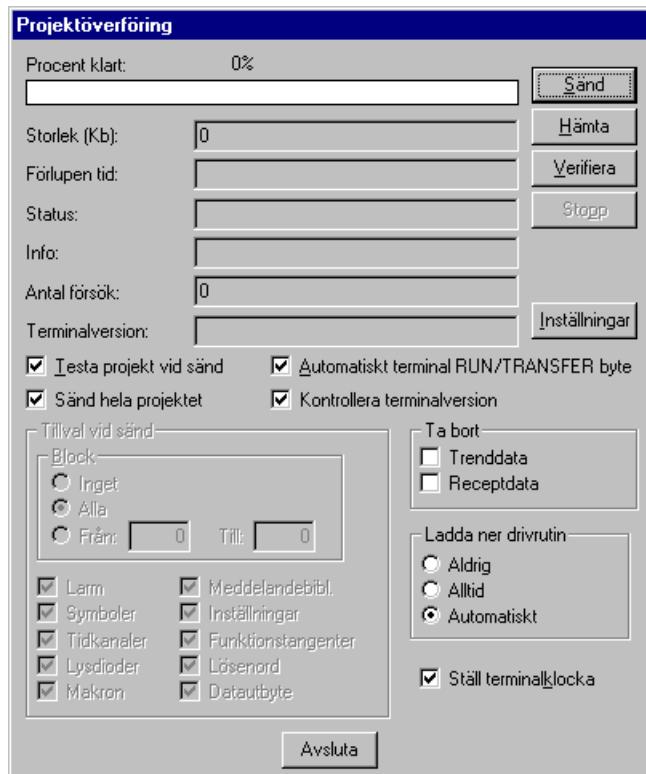
Kommandot **Timeout (T)** används för att ställa hur lång tid servosystemet har på sig att svara innan ett kommunikationsfel uppstår. Den förinställda tiden är 2000 ms.

Kommandot **DEBUG** används för att visa vilka instruktioner som skickats till servosystemet. Skapa ASCII-signalen **DEBUG** bestående av 30 tecken innan kommandot används. Bortse från de sista tecknen som visas i strängen.

Då det finns flera signaler som endast är lokala variabler i terminalen, kan det finnas block i projektet som bara kommunicerar då något ska skrivas till servosystemet. Om kommunikationsproblem uppstår i sådana fall, upptäcks eller visas detta inte förrän en signal ska skrivas. För att undvika detta kan kommandot **COMCHK** anges på kommandoraden, och då kommer drivrutinen alltid att utföra en kommunikationskontroll i varje block. Var uppmärksam på att kommunikationen mot andra portar på servosystemet kan störas när **COMCHK** har angetts på kommandoraden.

3.2 Överföra drivrutinen till operatörsterminalen

Drivrutinen laddas ner till terminalen när projektet överförs till terminalen. Välj Överför/Projekt.



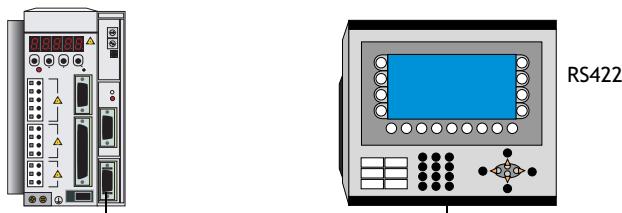
Det finns tre alternativ för hur drivrutinen ska överföras.

Funktion	Beskrivning
Aldrig	Drivrutinen laddas aldrig, befintlig drivrutin i terminalen används.
Alltid	Drivrutinen laddas varje gång hela projektet överföres.
Automatiskt	Drivrutinen laddas endast om drivrutinen i terminalen inte är samma som i projektet.

3.3 Ansluta servosystemet

Anslutningen kan vara av typen Point-to-Point via RS422 eller via RS232.

RS422-anslutning Point-to-Point



En kabel enligt kapitlet Ritningar ansluts mellan CN6-porten på servosystemets optionsmodul NS600 och RS422-porten på operatörsterminalen.

RS232-anslutning Point-to-Point

Två kablar kan kopplas ihop för att ansluta operatörsterminalen och servosystemet.

14-polig 3M Yaskawa NS600	9-polig D-sub hona	9-polig D-sub hane	9-polig D-sub hona, operatörsterminal
2	2	2	3
4	3	3	2
14	5	5	5
	7		
	8		
	1		
	4		
	6		

Använd programmeringskabeln NS600, avsedd för kommunikation mellan servosystem och persondator, och en kabel enligt ovan höger. Koppla ihop det två kablarna och anslut den 9-poliga honkontakten till RS232C-porten på terminalen.

Notera!

Multidrop-anslutning kan också göras. Se Yaskawa-manualen för mer information.

Korrektar parametrar i servosystemet

Ställ in servosystemets parametrar enligt nedanstående tabell.

Parameter	Parameternamn	Inställning	Beskrivning
Pn800	Serial Communication Protocol (Protokoll för seriell kommunikation)	1	RS422 + Echo back
Pn801	Baud rate (Överföringshastighet)	0	9 600
Pn802	Answer (Svar)	1	OK = svar

Parametrarna Pn800 och Pn801 är "offline parameters", vilket innebär att spänningen till servosystemet måste slås från och till igen för att nya inställningar ska få genomslag. Parametern Pn802 är en "online parameter", vilket innebär att en ny inställning slår igenom direkt.

Överföringshastigheten kan ökas genom att parametern Pn801 ändras motvarande terminalens inställning. Sätt Pn801 till 1 för hastighet 19 200 eller till 2 för 38 400.

För ytterligare information om parametrar och inkopplingar för servosystemet hänvisas till manualen för servosystemet.

4 Adressering

4.1 Digitala och analoga signaler

Operatösterminalen kan hantera digitala signaler (ON/OFF) och analoga signaler (värden) av typer enligt nedanstående tabell.

Se Yaskawa-manualen för ytterligare information.

Typer i programtablå:

Signalnamn	Yaskawa-signal	Typ	Beskrivning
POSTCHOICE	(POST)	16-bitars	
POSTVALUE	(POST)	32-bitars	
POST	POST	Bit	
SPDT	SPDT	32-bitars	
SPDTRIGG	(SPDT)	Bit	Skickar innehållet i SPDT.
RDSTT	RDSTT	32-bitars	
RSPDT	Ingen	32-bitars	
RSPDTRIGG	Ingen	Bit	
POUTT	POUTT	16-bitars/ASCII	Kan bestå av 5 ASCII-tecken, men att ange signalerna POUTT_0, POUTT_1, POUTT_2, POUTT_3, POUTT_4 rekommenderas för flerval. Se kapitel 4.2.
EVENTTCHOICE	(EVENTT)	16-bitars	
EVENTTVALUE	(EVENTT)	32-bitars	
EVENTT	EVENTT	Bit	
LOOPT	LOOP	32-bitars	
ITNEXT	(NEXTT)	32-bitars	När denna signal ändras skickas den till servosystemet. Se även POSEVENT. Om ITNEXT läses som END läggs 0 i ITNEXT och ITNEXTDIG ändras till END.
ITNEXTDIG	NEXTT	Bit	Rekommenderad användning: text0 = JMP text1 = END
ZONEPT	ZONEPT	32-bitars	
ZONENT	ZONENT	32-bitars	
JSPDT	JSPDT	32-bitars	
POSEVENT	Ingen	Bit	Utför det både POST och EVENTT gör. Dessutom, om ITNEXTDIG sätts till END, skickas END till NEXTT i indextabellen till den rad som motsvarar adressen som satts för POSEVENT. Exempel: Om ITNEXTDIG sätts till JMP, skickas inget till NEXTT på raden i indextabellen, då det redan finns där (värdet i ITNEXTT).

Typer utanför programtablån:

Signalnamn	Yaskawa-signal	Typ	Beskrivning
ASCIIS	Ingen	ASCII (max 30)	Bortse från projektets stationsinställningar och stationsindex. Strängen i ASCIIS skickas då ASCIISTRIGG är satt. Exempel: 1SVON eller FSVON . (SVON till station 1, SVON till station F). Används huvudsakligen för felsökning.
ASCIIR	Ingen	ASCII (max 30)	Det som senast mottogs från servosystemet. Används huvudsakligen för felsökning.
ACC	ACC	32-bitars	ACCTRIGG skickar innehållet i ACC .
ACCTRIGG	(ACCA9	Bit	
DDEC	DEC	32-bitars	DDECTRIGG skickar innehållet i DDEC .
DDECTRIGG	(DEC)	Bit	
PRM	PRM	32-bitars	Sätt PRMASCII till att bestå av 4 tecken. PRMASCII används endast för PRMO .
PRMASCII	PRM	ASCII	MULPRM används när värdet i PRM ska användas av ett objekt med flerval. Exempel med Pn801 : 9 600, 19 200, 38 400 kan visas i stället för 0, 1, 2.
MULPRM	Ingen	16-bitars	
SVON	SVON	Bit	
SVOFF	SVOFF	Bit	
STARTN	STARTss	Bit	
STOP	STOP	Bit	
SKIP	SKIP	Bit	
MON	MON	32-bitars	
JOGVALUE	(JOG)	32-bitars	Ange JOGP/JOGN momentant till en funktionstangent. När tangenten trycks ner skickas JOGP/JOGN med värdet i JOGVALUE . När den släpps skickas SKIP .
JOGP	JOGP	Bit	
JOGN	JOGN	Bit	Om JOGP/JOGN skickas (ej momentant), skickas JOGP/JOGN och jog kommer att pågå ända tills operatören skickar SKIP genom att objektet SKIP sätts.
RJOG			Ej implementerad
DBRMS	DBRMS	32-bitars	
EVTIME	EVTIME	32-bitars	
IN1	IN1	32-bitars	Nollutfyllning, bitadressering är möjlig
IN2	IN2	32-bitars	Nollutfyllning, bitadressering är möjlig
LOOP	LOOP	32-bitars	
MTSIZE	MTSIZE	32-bitars hex	
MTTYPE	MTTYPE	32-bitars hex	
OUT1	OUT1	32-bitars	Nollutfyllning, bitadressering är möjlig
OUT2	OUT2	32-bitars	Nollutfyllning, bitadressering är möjlig
PGTYPE	PGTYPE	32-bitars hex	
PGVER	PGVER	32-bitars hex	
RGRMS	RGRMS	32-bit	
STIFFD	STIFFd	Bit	
SVTYPE	SVTYPE	32-bitars hex	

Signalnamn	Yaskawa-signal	Typ	Beskrivning
SVVER	SVVER	32-bitars hex	
SVYSPEC	SVYSPEC	32-bitars	
TRMS	TRMS	32-bitars	
TYPE	TYPE	32-bitars hex	
VER	VER	32-bitars hex	
YSPEC	YSPEC	32-bitars	
INERTIA	INERTIA	32-bitars	
RDST	RDST	32-bitars	
LOOPT	LOOPT	32-bitars	
TRM	TRM	32-bitars	Sätt TRMASCII till att bestå av 4 tecken. TRMASCII används endast för TRMO.
TRMASCII	(TRM)	ASCII	
RES	RES	Bit	
ARES	ARES	Bit	
STNINDEX	Ingen	16-bitars hex	
INDEX	Ingen	16-bitars	
INTEST	IN2TEST	32-bitars	Nollutfyllning, bitadressering är möjlig
OUTTEST	OUT2TEST	32-bitars	Nollutfyllning, bitadressering är möjlig
ALMN	ALMN	ASCII/ 16-bitars hex	Om 16-bitars hex används visas larm- eller felkoden i den returnerade strängen här. Om det inte finns någon larmkod visas 0.
ALMTRCCLR	ALMTRCCLR	Bit	
ERR	ERR	ASCII/ 16-bitars hex	Om 16-bitars hex används visas felkoden i den returnerade strängen här. Om det inte finns någon felkod visas 0.
POUT	POUT	16-bitars/ ASCII	Kan bestå av 5 ASCII-tecken, men att ange POUT_0, POUT_1, POUT_2, POUT_3, POUT_4 rekommenderas för flerval. Se beskrivning av POUTT i kapitel 4.2.
PGMSTEP	PGMSTEP	ASCII	Att ställa PGMSTEP till att bestå av 3 tecken rekommenderas.
STIFF	STIFF	32-bitars	
ABSPGRES	ABSPGRES	Bit	
MLTLIMSET	MLTLIMSET	Bit	
ALM	ALM	ASCII	
TUNESTORE	TUNESTORE	Bit	
CURZERO	CURZERO	Bit	
PGMINIT	PGINIT	Bit	
ZONEINIT	ZONEINIT	Bit	
JSPDINIT	JSPDINIT	Bit	
PGMSTORE	PGMSTORE	Bit	
ZONESTORE	ZONESTORE	Bit	
JSPDSTORE	JSPDSTORE	Bit	
START	START	Bit	
PGMRES	PGMRES	Bit	
SPD	SPD	32-bitars	
RSPD	RSPD	32-bitars	

Signalnamn	Yaskawa-signal	Typ	Beskrivning
POSVALUE	Ingen	32-bitars	Se instruktioner för att använda POSVALUE .
POSVALUE	Ingen	32-bitars	Se instruktioner för att använda POSVALUE .
STI	(STI)	Bit	Tar värdet i POSVALUE och skickar det med STI som 1STI00000100 (1 station).
ST	(ST)	Bit	Tar värdet i POSVALUE och skickar det med ST som 1ST00000100 (1 station).
ST2	(ST)	Bit	Skickar strängen 1ST till servosystemet om station 1 valts. Vid 2: ST2 skickas 2ST .
RSI	RSI	Bit	Tar värdet i POSVALUE och skickar det med RSI som 1RSI00000100 (1 station).
RS	RS	Bit	Tar värdet i POSVALUE och skickar det med RS som 1ST00000100 (1 station).
POSI	(POSI)	Bit	Tar värdet i POSVALUE och skickar det med POSI som 1POSI00000100 (1 station).
POS	(POS)	Bit	Tar värdet i POSVALUE och skickar det med POS som 1POS00000100 (1 station).
HOLD	HOLD	Bit	
ZRN	ZRN	Bit	
PRMINIT	PRMINIT	Bit	
ZSETVALUE	ZSETVALUE	32-bitars	
ZSET	ZSET	Bit	

De signaler som tvingas till 32-bitars kan inte användas för flerval (multiple choice) utom PRM som har specialsignalen MULPRM.

Notera!

Signalernas format måste överensstämma med vad som framgår av detta dokument. Om exempelvis 16-bitars anges i dokumentet och 32-bitars används, kan fel uppstå.

Specialvariabler

Instruktioner som kan skickas till, men inte läsas från servosystemet hanteras av några interna variabler i operatörsterminalen. De signalerna avspeglar inte nödvändigtvis servosystemets status.

Exempel:

- När den digitala signalen **SVON** (som kan skrivas till men inte läsas från) trycks ner, sätts den till 1 och visar texten från den signal som motsvarar 1 i programmeringsverktyget för operatörsterminalerna. Servosystemet startar då om allt är i sin ordning. Om spänningen till servosystemet slås från, kommer operatörsterminalen fortfarande att visa att servosystemet är igång, fast det inte är det. I detta fall kan **SVON** ändras till 0 utan problem (**SVOFF** kan ju skickas till ett servosystem som redan är avstängt).
- Om en operatörsterminal känner till att ett servosystem är igång, och spänningen till terminalen bryts, kommer terminalen att visa att servosystemet är avstängt. Då kan **SVON** ändras till 1 utan problem (**SVON** kan ju skickas till ett servosystem som redan är igång).

Interna variabler i operatörsterminalen

Signalerna ACC, ASCIIR, ASCIIS, DDEC, JOGN, JOGP, JOGVALUE, POSVALUE, POSIVALUE, RSPD, SPD, STNINDEX, SVOFF, SVON och ZSETVALUE har interna variabler.

Dock har inte alla signaler en variabel till var och en av de 16 möjliga stationerna (axel).

Exempel:

JOGP har en variabel åt varje station, men bara ett JOGVALUE. Stationerna måste då dela JOGVALUE. Två signaler, 1:JOGP och 2:JOGP används. När 1:JOGP sätts till 1, skickas jog-instruktionen med värdet i JOGVALUE till station 1. När 2:JOGP sätts till 1, skickas jog-instruktionen med värdet i JOGVALUE till station 2.

Endast JOGN, JOGP, SVOFF och SVON har en variabel åt varje station.

Följande variabler har en intern variabel för varje rad i indextabellen: POSTVALUE, POSTCHOICE, EVENTTVALUE, EVENTTCCHOICE, ITNEXT.

Viktig information beträffande blockbyten

När ett blockbyte utförs i operatörsterminalen, läses värdena för de interna variablerna från servosystemet.

Exempel:

Ett blockbyte till ett block som använder variablerna POSTCHOICE0 och POSTVALUE0 utförs. 1POST0 skickas (axel 1).

1POST000=A+12345678[CR][LF] mottages. POSTCHOICE0 uppdateras med värdet 0, och terminalen visar A (om flerval har skapats enligt kapitel 4.2). POSTVALUE0 uppdateras med värdet 12345678. Om värdet i POSTVALUE0 ändras och **Enter** trycks ner, ändras bara den lokala variabeln i terminalen. För att värdet ska skrivas till servosystemet, måste variabeln POST triggas.

Notera!

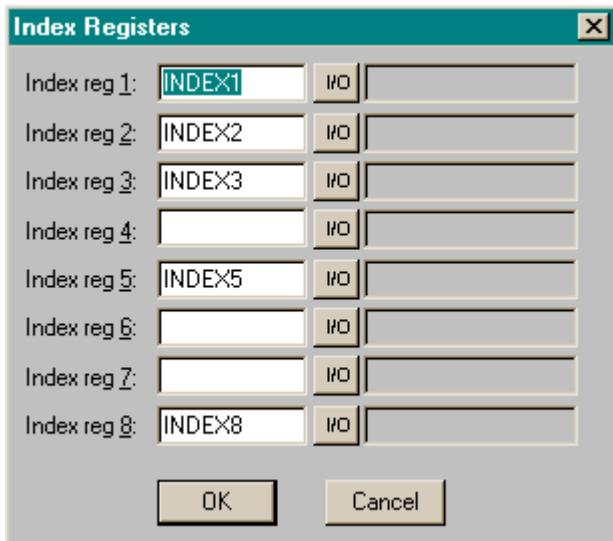
Recepthantering med denna drivrutin har begränsningar i funktionalitet. Recept bör endast användas med indextabellen, och ingen annanstans.

Det finns endast 16 stationer att adressera, och 0-127 rader i indextabellen.

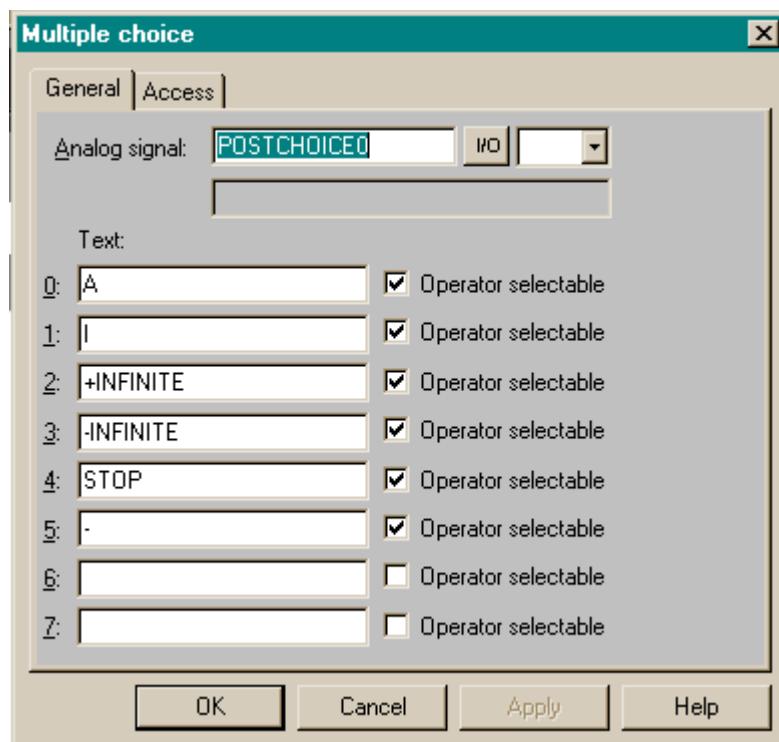
Att använda vektorer fungerar inte med denna drivrutin.

4.2 Exempelprojekt

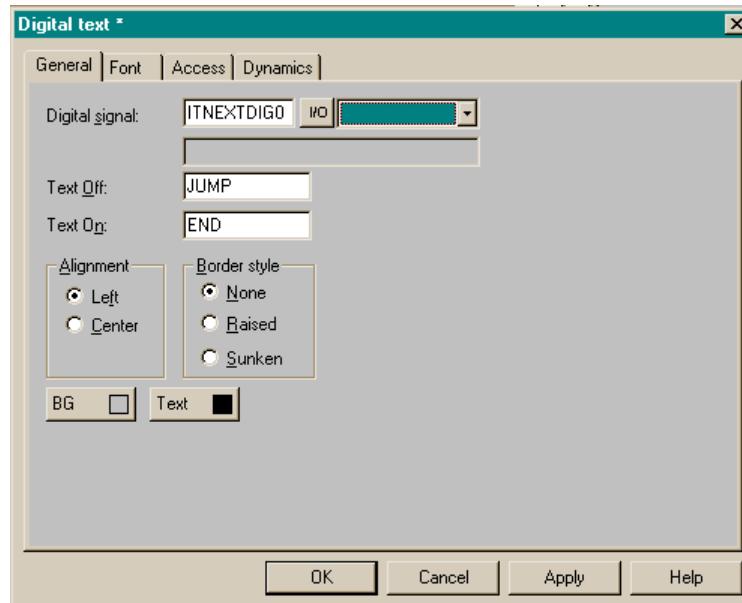
signalen INDEX har åtta variabler, en för varje index.



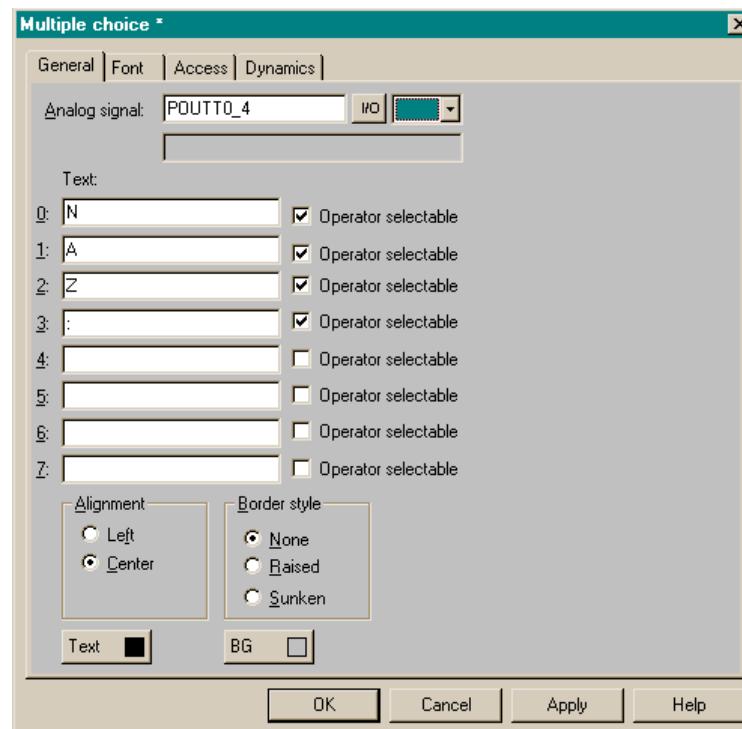
POSTCHOICE: Texten har ingen inverkan på vad som skickas till servosystemet.
Se texterna nedan som förslag.



ITNEXTDIG: Text Off används för att visa JMP eller JUMP, och Text On för att visa END.

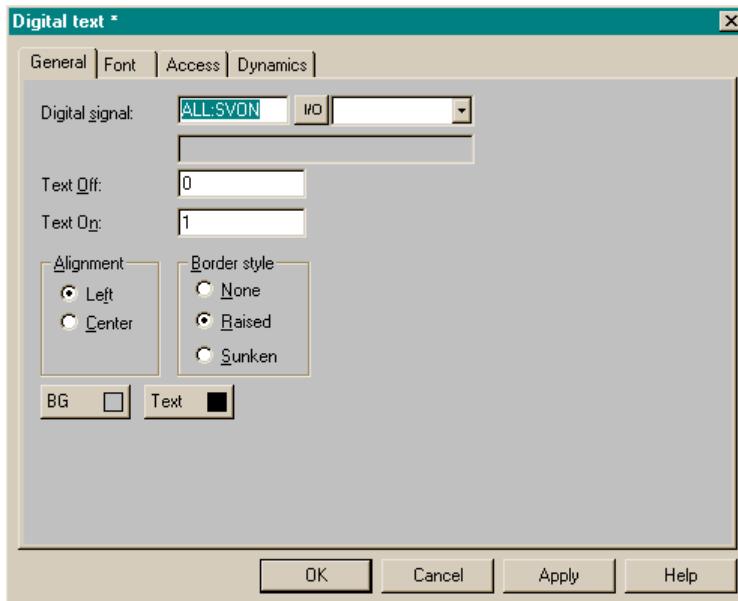


POUTT: Texten har ingen inverkan på vad som skickas till servosystemet.
Se texterna nedan som förslag. Bitarna adresseras POUTT0_0 - POUTT0_4.
POUTT0_0 adresseras för den minst signifika biten.

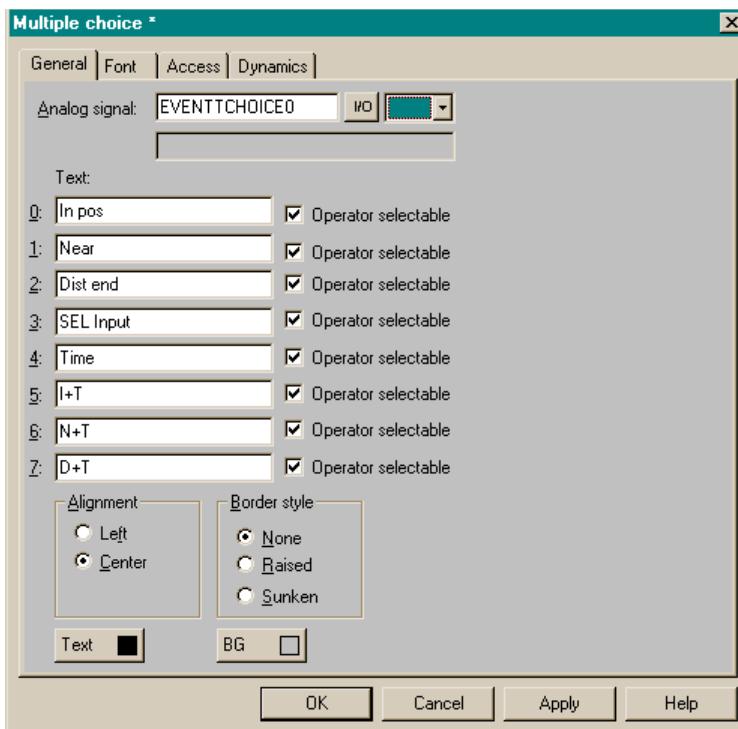


Adressering

Exempel på en signal som adresseras som **ALL**. I detta fall skickas strängen ***SVON**.



EVENTCHOICE: Texten har ingen inverkan på vad som skickas till servosystemet.
Se texterna nedan som förslag.



4.3 Recept

Utför instruktionen nedan om POUTT ska användas i indextabellen och sparas som ett recept:

Om POUTT används som POUTT0_0 - POUTT0_4 måste det också finnas ett objekt POUTT0 som visas som ASCII (5 positioner). Objektet behöver inte vara synligt eller kunna ändras; det används endast för att spara och ladda recept ska fungera ordentligt.

Recept som används i textblock

Om ett recept används i ett textblock, måste det faktum att det går att rulla (scrolla) textblock iakttagas. Under vissa omständigheter kan problem uppstå:

Anta att en terminal som endast visar två rader i ett textblock används.

Den första raden visar POSTCHOICE0, den andra POSTCHOICE1, och den tredje (som inte är synlig) POSTCHOICE2.

Vid hopp till detta block, läses och uppdateras värdena för de variabler som visas (se avsnittet Viktig information beträffande blockbyten i kapitel 4.1). Värdet för POSTCHOICE2 är inte läst ännu.

Vi byter nu värden för POSTCHOICE0 och POSTCHOICE1 och triggar dem båda till att skrivas till servosystemet.

Sedan sparas detta som ett recept. Detta vore fel, då sparad värde för POSTCHOICE2 är slumpmässigt valt.

När receptet laddas skrivs det slumpmässiga värdet till de interna variablerna, och när det triggas skickas det slumpmässiga värdet till servosystemet, och förstör det riktiga värdet i POST2 (den tredje raden i indextabellen).

Prova gärna att ändra värdet i POSTCHOICE0, och rulla ner till POSTCHOICE2, så att POSTCHOICE0 försvinner. Rulla sedan upp till POSTCHOICE0 igen, och se att värdet har skrivits över av en ny läsning från servosystemet.

Rätt tillvägagångssätt:

Rulla alla rader vid hopp till textblocket för att bekräfta att värdena i terminalen stämmer överens med värdena i servosystemet.

Ändra sedan önskade värden, och var noga med att triga skrivning av dem till servosystemet.

Spara receptet. När receptet sedan laddas kommer rätt värden att användas.

Alternativt kan POSTCHOICE0 och POSTCHOICE1 ändras, och rulla sedan ner till POSTCHOICE2 och ändra eventuellt även det värdet.

Spara sedan receptet. Värdet i POSTCHOICE0 är riktigt, då det inte förstörs förrän det visas igen på terminalens display.

När receptet nu läses och värdena triggas till servosystemet, borde receptet vara rätt.

5 Effektiv kommunikation

Signalerna avfrågas, en och en, och kan inte packas. Dock kan överföringshastigheten kan ändras till 38 400.

6 Felsökning

Om en variabel inte fungerar som förväntat

Utför följande steg om exempelvis ZRN inte fungerar som förväntat:

Skapa ett nytt block med ZRN (gör det ändringsbart) och signalen DEBUG (bestående av 30 ASCII-tecken).

Lägg till signalen ASCIIR (bestående av 30 ASCII-tecken).

Ange kommandot DEBUG på kommandoraden.

Skicka projektet till operatörsterminalen.

Gå till ZRN och tryck på Enter. DEBUG visar nu vad som skickats, och ASCIIR visar vad som tagits emot. Kanske finns det en felkod som beskrivs i manualen för servosystemet.

Alternativt kan ett terminalprogram anslutas. Kontrollera på så vis om servosystemet uppför sig som förväntat.

7 Ritningar

RS-232

Pin no	Name	Signal direction
		Terminal
1	+5V >200mA	
2	TxD	
3	RxD	
5	0V	
7	CTS	
8	RTS	
9		

4) Only for units with one serial port (RS-422) and
232-25-Arp (adapter). On all other terminals,
not connected.

RS-422

Pin no	Name	Signal direction
		Terminal
2	+TxD	
15	-TxD	
3	+RxD	
16	-RxD	
4	+RTS	
5	-RTS	
17	+CTS	
18	-CTS	
20	1)	
21	1)	
7,8	0V	
14	+5V	
	<500mA	
12,13	2)	
24,25	+5V >200mA	
9	3) TxD	
10	3) RxD	
22	3) CTS	
23	3) RTS	

- 1) Pin no 20 connected to pin no 21 internal in the terminal
2) Only for units with sync voltage feed
3) Only for units with one serial port (RS-422)

RS-485

Pin no	Name	Signal direction
		Terminal
1	Tx/Rx+	
2	Tx/Rx-	
3	0V	
4		

- 1) Pin no 20 connected to pin no 21 internal in the terminal
2) Only for units with sync voltage feed
3) Only for units with one serial port (RS-422)

RS-232/RS-422/RS485

Date	Page	Rev.	Page	Next page	Forming No.
SLG 1998-12-01	A3	-	2001-03-29	1	S-02467

Kabel mellan operatörsterminal och Yaskawa NS600

