MAC motors and Device Net control

Content

Description and test: April 2005, Allan Rex Pedersen, Automatik Partner ApS
(Rev 17-06-2005, changes in zerosearch function)

Note: SLC does not support the variable "Long", which is also known as a doubbleword integer. The devicenet interface is byte based and values are supposed to be "Integers" and "Long integers". To compensate the functions are build to work with 2 word, but You cannot trust the values shown in monitor.

In this example Mac drive number 1, has node adr. #5 and Mac drive number 2, has node adr. #6
For IO is used a Phoenix in-line IL DN BK3 node adr. #2 (1 byte out / 1 byte in).

List of project files

AB-JVL.dnt Devicenet project for rockwell network 1 motor.
AB-JVL2.dnt Devicenet project for rockwell network 2 motors and 1 phoenix I/O
DEVAB-JVL.rss PLC program for using 1 motor at device net.

PLC Project files are valid for CPU 5/03 to 5/05 and Micrologix

List of software tools for this note

Controlled by: Allen-Bradley device net master module
PLC program is made in RSLogix 500 veresion 6.00
Device net scanner software version version 5.11.00, from Rockwell
RS_Linx 2.41 and 2.43 lite
MacTalk version 1.34 beta2, from JVL
MacRegIO version 1.14

List of equipment for making this note

Allen-Bradley PLC: CPU5-03, 1747 DeviceNet scanner module, digital input and digital output, and Phoenix Contacts In-line I/O block
2 pcs. Mac motor equipped with each a device net interface: MAC140 + MAC00-FD4

Short info about communication to mac motor

All communication is done via 8 bytes or 4 word, having a layout like a telegram. It is possible to set / get information to and from servo drive.
The communication is basically like setting specified values in the 4 word (see figure 1), and set the bit "LoadData", and wait for response at the input bit "LoadComplete" (see figure 2).
As the devicenet telegram layout is in byte and plc program in word, there will be a great job converting this information.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	-	Hard Stop	Smooth Stop	Direction (V. Mode)	Incremental	-	Load Data
1	Command Data 1							
2	Command Axis Number			Command Message Type				
3	Command Data 2							
4	Command Data 3							
5	Command Data 4							
6	Command Data 5							
7	Command Data 6							

Figure 1 Telegram layout for device net setting control bit and word. Startadr.: O:1.1

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Enable	-	-	-	General Fault	On Target Position	-	Profile in progressa
1	Response Data 1							
2	Load Complete	-	-	-	-	Rev Limit	Fwd Limit	-
3	Response Axis Number			Response Message Type				
4	Response Data 2							
5	Response Data 3							
6	Response Data 4							
7	Response Data 5							

Figure 2 Telegram layout for device net getting status as bit and word. Startadr.: I:1.1

How to use...

The idea about this note and program is to have an easy to use program, that may give You a possibility for fast setup and test of Your hardware. Hopefully You may save some development time using the subroutines in Your programming.

Following step is necessary:

1. set switches like described below
2. Run "RSNetworx for devicenet", create network and download to Your devicenet scanner (master). NB. Scanner should have lowest node number.
3. If needed change addresses, and correct parameter "Node addr." In Subroutine calls.
4. set enable flag for Your drive
5. set flag Enable (B3:11/8) and also MOVE (B3:5/0 or B3:5/1), to see Your drive positioning.
6. Make modifications and finish Your job...

Good luck !

Configuration and setup of scanner

Using the devicenet configurator software (RSNetworx for devicenet), You may put the wanted nodes in Your devicenet, and download to network.
Check and correct switch setting on Your nodes. Nodenumber shall match the configurator software.
Also the baudrate setting shall be the same for all units.
Resistors also called terminators should be either mounted or activated by switch, in each physical end of the devicenet.

Note: It is only possible to download devicenet configuration to the scanner when CPU is in "program" mode

Configurator setup:

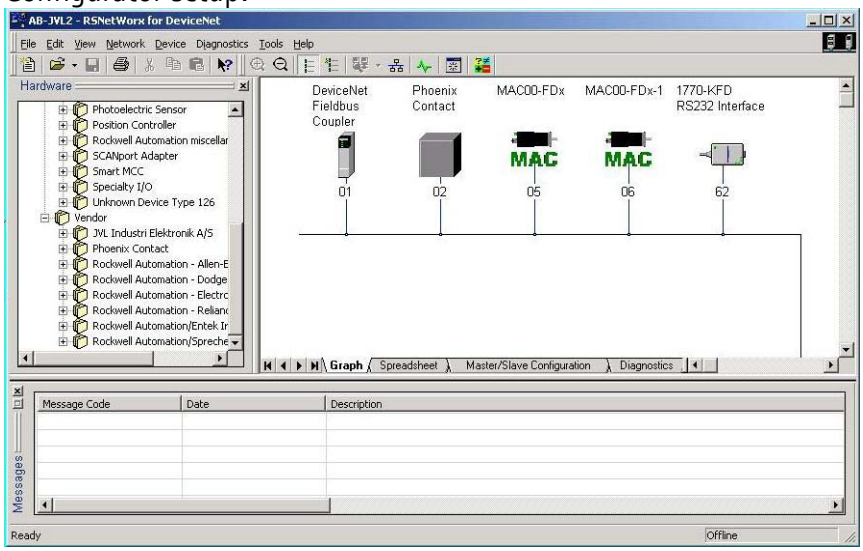

fig. 3 Device net configured only with 2 drives and I/O. Filename: AB-JVL2.dnt

Allen-Bradley PLC setup

CPU SLC5-03 OS302

Device net Scanner: 1747-SDN mounted in first slot, 1. Version 7.006
Nodenumber: 63 (factory set), changed to 01 in this test.
Baud rate: 125 kbit

PLC Status and control memory area:
O:1.0 Reserved data area for IO communication of device net (control, write)
I:1.0 Reserved data area for IO communication of device net (status, read)

NB address area depends on mounting position in rack. If You choose to change position You also have to choose the addresses in this example.
NB to make the device net running the LSB of the first word in scanner must be on. (see last line in Main-test section).

Device net I/O addresser (1.st slot device net):

Input adr.	Output adr
I:1.0 Scanner status	O:1.0 dig. Output
I:1.1 Mac node #5 status	O:1.1 Mac node #5 command
I:1.2 Mac node #5	O:1.2 Mac node #5
I:1.3 Mac node #5	O:1.3 Mac node #5
I:1.4 Mac node #5	O:1.4 Mac node #5
I:1.5 Mac node #6 status	O:1.5 Mac node #6 command
I:1.6 Mac node #6	O:1.6 Mac node #6

I:1.7 Mac node #6	O:1.7 Mac node #6
I:1.8 Mac node #6	O:1.8 Mac node #6
I:1.9 Phoenix input byte #2	O:1.9 Phoenix Output byte #2

Mac setup

Node number should be set at DIP switch.
Selection of termination for devicenet Switch 2 (if the drive is the last device physically in the devicenet wiring then ON, else OFF)
Selection of Baudrate for the devicenet

To set node: #5, and Baudrate: 125k

Switch no.	position
1	on
2	off
3	on
4	off
5	off
6	off
7	off
8	off
9	off
10	off

PLC program description: Easy-Mac

The PLC program is split in several sections to make it more simple to understand.
The program is constructed using subroutines for driver operation, The user should apply new parameters and call the routines.
For simple test of program, You may use the monitorlists "Custom data monitor" in the project.
Common for the project is the word B3:3, collecting error flag from each part of the program.

Section Main-test

Examples of how to call the different function to control Mac drives. The sections necessarily can easily be copied into other User applications with the needed subroutines.
In this example the following control flags are used: enable, hardstop, incremental and load data.

Section UseMove

Move function is an "Easy to start" function for a simple positioning. This section shows how to set parameters.
The section includes 2 call. One for position out at x100000, and one for position back to 0. The 2 call has different positioning profiles. To position X using slow acceleration but high speed, and return to position 0 using slow speed but fast acceleration and deceleration. See also Move below.
The only thing to do to position at 100.000 is to set flag (B3:5/0), the rest is turning out automatically. To return to 0, simply set (B3:5/1).

Calls DoCommand 3 times for parameter transferring to driver sending first velocity, then acceleration and target position at last. Thereafter the actual position will be requested every 2. second until drive is "In-position" again.

Section Device Net Common
Get status flags from drive and set command flags to drives, and check whether the call are.
Using Direct addressing, and shall be changed when device net addresses are changed.
Also containing status registers of the device net scanner, for simple trouble shooting.
Status flags: general fault, rev limit, fwd limit, in position, enabled, load completed.

Section Fault
Common section for indicating error and fault running device net, drives and communication functions.
The different timeout flags and other error indicators are collected into one flag (B3:7/0) which will indicate if an error should appear. Also an output will flash in the example (O:3.0/0).

Section DoCommand, and Section DoParameter (SUB)
Basic commands (functions) are DoCommand and DoParameter doing transfers to and from drives using relativ adressing. The functions can handle one operation per call. Each call should be finished (correct) before another call can be done.
Each call will activate a small communication sequence for handshaking parameter to drives in a safe way.
The functions are relative addressing drives, which means that a drive is addressed using the first address for the drive in the devicenet, as the parameter NodeAddress (Interval 1..63).
DoParameter is also capable of handling DoPositionControllerAttribute objects, depending the value of parameter Messagetype (DoParameter messagetype hex 1f) (DoPositionControllerAttribute hex 1b).

Section MoveSub (SUB)
Move function is an "Easy to start", kind of high level instruction. Using this function You only have to do one call in your part of the application to transfer a position profile to the drive and monitoring actual position until operation finished and motor is "In-position".
This function is based upon several calls to basic command "DoCommand".
See a call example in section UseMove. Set parameters and activate MOVE function

Section ZeroSearch (SUB) rev. 17-06-2005
The function will at call, Reset enable of the drive, sets the parameters for zerosearching, and at last set enable of drive. The handling of Enable is caused in the fact that it is the positive transition of Enable which activates the transferred parameters.
Basically this function work as a 6 step sequence calling DoParameter 3 times one for each parameter using messagetype (hex 1b).
The parameter "Zerosearch velocity", act now as a double word, using signed notation. Positive values start zeroseaching CW, and negative values start zeroseaching CCW.

Negative values are 2's complement and require MSB set in N7:45.

Section Initialize
Sets the most important parameters for drive control.
Basically this function work as a 6 step sequence calling DoCommand 3 times one for each parameter.

Description of Functions / Subroutines:

DoCommand
Ressources: word B3:1; Sub DO_COMMAND; call sub flag: B3:0/1; Error B3:3/0
Parameter layout

Address for parameter	Calling parameter	Returning parameter	Comment
N7:10 (and N7:11)	DINT		Parameter value
N7:12	INT		Message type / command no.
N7:13	INT		Node address
N7:14	INT		Response Message type / command
N7:17	INT		Responding message type
N7:18 (and N7:19)		DINT	Response value

Example using DoCommand:
The function with example parameters will:
With node #5 (address 1.1 to 1.4), set target position= 10.000, Messagetype=1 means "move to", and Response message type = 1 means return actual position.

To activate example Set bit B3:0/0
DoCommand(node: 1, Value: 10000, CmdMsg: 1, ResponsMsg: 1),
Return value = actual position and response msg. = 1.

Word scheme for PLC addresses writing DoCommand function at devicenet

O:1.1	Block number	Control flags
O:1.2	Response Axis no. and Message type	Control Axis no. and Message type
O:1.3	Parameter value low order word	
O:1.4	Parameter value high order word	

Word scheme for PLC addresses reading DoCommand respons at devicenet

I:1.1	Block number	Status flags
I:1.2	Response Axis no. and Message type	Status flags
I:1.3	Parameter value low order word	
I:1.4	Parameter value high order word	

Possible message types for command

Message number (hex)	Description	Comments
0	No operation	JVL standard
1	Target position	
2	Target velocity	
3	Acceleration	
5	Torque	
1B	PositionControllerAttribute	
1F	Parameter	

Possible message types for response

Message number	Description	Comments
0	No operation	
1	Actual position	
3	Actual velocity	
5	Torque	
14	Error code	N7:18 Lo order: general error code, and Hi order additional error code
1B	PositionControllerAttribute	
1F	Parameter	

NB some commands will only be valid when drive is enabled.

DoParameter
Resources: word B3:2; Sub DO_PARAM; call sub B3:0/12; Error B3:3/1
Message type: 1Fhex
Parameter layout

Address for parameter	Calling parameter	Returning parameter	Comment
N7:20 (and N7:21)	DINT		Parameter to set value
N7:22	INT		Messagetype: 1F hex
N7:23	INT		Node address
N7:24	INT		Attribute to set
N7:25	INT		Attribute to get
N7:30 (and N7:31)		DINT	Value of read parameter
N7:32		INT	Attribute no. Read

NB: For normal position / velocity mode it's not necessary to use this function. Please notice that the units of the parameters is different from the devicenet units.

Example using DoParameter:
The function with example parameters will:
With node #5 (address 1.1 to 1.4), set value= 50, Messagetype=1f means parameter, Set attribute=6 (V_SOLL) and Get attribute=14 (GEARF1)

To activate example Set bit set B3:0/10,
DoParameter(value: 50, Node: 1, Messagetype: #1F, Attribute set: #6, Attribute get: #14),
return value: 256, get attribute: 14 (means GEARF1 = 256)

Word scheme for PLC addresses writing DoParameter function at devicenet

I:1.1	Parm Get Attribute	Control flags
I:1.2	Parm Set Attribute	Axis no. and Message type
I:1.3	Parameter value low order word	
I:1.4	Parameter value high order word	

Word scheme for PLC addresses reading DoParameter function at devicenet

I:1.1	Parameter read	Status flags
I:1.2	Messagetype	Status flags
I:1.3	Parameter value low order word	
I:1.4	Parameter value high order word	

Some parameters are only possible to change when drive is enabled.

For further detail see "Technical Manual" for the motor available from JVL.

DoPositionControllerAttribute
Ressources: word B3:2; call DoParameter; call sub B3:0/11; Error B3:3/1
Messagetype: 1Bhex
This function is accomplished using call to DoParameter, but with parameter Messagetype = 1Bhex.

Address for parameter	**Calling parameter**	**Returning parameter**	**Comment**
N7:20 (and N7:21)	DINT		Parameter to set value
N7:22	INT		Messagetype: 1B hex
N7:23	INT		Node address
N7:24	INT		Attribute to set
N7:25	INT		Attribute to get
N7:30 (and N7:31)		DINT	Value of read parameter
N7:32		INT	Attribute no. read

Example using DoPositionControllerAttribute:
The function with example parameters will:
With node #5 (address 1.1 to 1.4), set value= 50, Messagetype=#1B means messagetype, Set attribute=6 (homing torque) and Get attribute=102 (homing velocity).

Torque limit for zerosearch.
Notice: Torque zerosearch first activates at next rising enable. And is not visible.

To activate example Set bit set B3:0/13,
DoPositionControllerAttr.(value: 50, Addressing Node: 1, Messagetype: #1B, Attribute set: 103, Attribute get: 102),
return value: 105, get attribute: 102 (value may be different)

ZeroSearch
Ressources: word B3:8; Sub ZEROSEARCH; call DoParameter; Error: B3:3/4
Messagetype: 1Bhex (basically DoPositionControllerAttribute)

N7:40	INT	Type of zerosearch
N7:41	Not in use	Reserved,
N7:42	INT	Zerosearch Torque limit
N7:44 (and N7:45)	DINT	Zerosearch velocity
N7:43	INT	Node address

This function begin reset and after parameter transfer set of the "Enable flag" for the drive. The function call will bee active while searching and until the drive is "in-position".

Example using zerosearch: set torque limit.
NB: this example will turn motor axis CW until torque limit is reached, then motor will change direction and turn on for 0,5 second. Finally an offset position will be done.
Speed 3413 = 50 RPM,
Torque limit at the value = 50 means that it is possible to stop motor using fingers on motor axis.

The function with example parameters will:
With node #5 (address 1.1 to 1.4), set Type of zerosearch=12, set Zerosearch velocity=3413, and set torque limit=50.

To activate example Set bit B3:0/13
Zerosearch (Type: 12, Velocity: 3413, Torque limit: 50)
No return value

An example to turn CCW during could be: N7:44 = f2ab(h) and N7:45 = ffff(h) equals –3413.
Or using integer values: N7:44 = -3414 and N7:45 = -1 equals –3413.

Move
Ressources: word B3:5; Sub MoveSub; call sub B3:5/3; Error B3:3/2
Parameter layout

Address for parameter	**Calling parameter**	**Returning parameter**	**Comment**
N7:60 (and N7:61)	DINT		Target position
N7:62 (and N7:63)	DINT		Velocity
N7:64 (and N7:65)	DINT		Acceleration
N7:66	INT		Node address
N7:70 (and N7:71)		DINT	Response act. position

This function will only work if drive is not set into relative positioning.

Example using Use_Move:
The function with example parameters will:
To activate example Set B3:5/0 and B3:5/1,

Move1(Target pos.= 100000, Velocity = 50000 , Acc.= 10000)
Move2(Target pos.= 0, Velocity = 5000 , Acc.= 100000)

While the function is active it will continously return actual position of the drive.

Activation of MOVESUB function by setting B3:5/3, the flag will be reset by the sub when drive becomes InPosition.

Parameter information:
Velocity: 5000 is slow, 100.000 is fast
Acceleration: 10.000 is slow, 100.000 is quicker (counts/secondˆ2)

Name	Text	Devicenet	Skala	Value interval
Target velocity	Velocity during positioning	DoCommand Mess.type 2	(ingen)	0..273.000 (4000 RPM)
Acceleration	Acc. And Dec. during positioning	DoCommand Mess.type 3		1..27.306.666 counts / sekˆ2
Target position	Target position	DoCommand Mess.type 1	(ingen)	-67.000.000 .. +67.000.000

Figure 11. Table for values to apply

Initialize
Ressources: word B3:9; Sub call DoParameter; call sub B3:0/9; Error B3:3/3

Parameter layout

Address for parameter	Calling parameter	Returning parameter	Comment
N7:80	DINT		Maximum velocity
N7:82	DINT		Acceleration
N7:84	DINT		Torque limit
N7:86	INT		Node address

Sets the most important parameters for drive operation.
NB this function will only be valid when drive is enabled prior to call.

Example using Use_Move:
The function with example parameters will:
With node #5 (address 1.1 to 1.4),set maximum velocity = 20000, Acceleration = 10000, torque limit = 50,

To activate example Set B3:0/9,

See table in figure 11, for further details.