Variable Speed Motor Modbus Register Specification

CHECKED _____ APPROVED _____

Quality Assurance Engineer Date Project Manager Date

Table of Contents

1 Revision History

Revision to this document may require approval with a Document Control Order. All revisions are listed in the following table.

Rev	Date	DCO	Author	Description
X1 – X16	December 2015 – January 2019	_____	_____	See version X16 for complete revision history from X1 to X16
X17	May 2020	_____	TB	Removed seldom used registers to improve readability

2 Introduction

2.1 Purpose

This document defines the Modbus register set used to interact and control a Nidec variable speed motor.

2.2 Audience

The specification assumes a basic understanding of the device as described in the Product Specification. It is written in a technical manner to provide detailed information to the customer, Nidec design engineers, quality assurance engineers, test engineers, and software testers.

2.3 Related Documents

- Modbus Application Protocol Specification V1.1b3
- Modbus over Serial Line Specification and Implementation Guide V1.02

3 Definitions

3.1 Glossary of Terms

Holding Register..... Register data that can be read and, in most cases, written by an application program
Input Register ..... Read-only register data provided by the system for application program use
{ } ..... Curly brackets denote a variable or a register name

4 Register Value Types

The Modbus specification does not place any restrictions on what is stored in a Holding register (see section **Error! Reference source not found.**) or an Input register (see section 7). For either register definition it is a simple 16-bit value and the data stored in those 16-bits is application defined. To provide more structure to the definitions of each register that are detailed in the following sections, each has been assigned to a more descriptive "type". The following table contains definitions for each of these types – this list is unique to this application and is not in any way connected to the ***Modbus Application Protocol Specification V1.1b3*** specification.

Register Type	Abbreviation	Variable Type	Units
Acceleration	Accel	Unsigned	RPM/sec
Address	Address	Unsigned	Valid Modbus address value
Binary	Binary	Unsigned	0=Off, False 1=On, True
Bit Image	Bit Image	Unsigned	None
Bus Voltage	V_{BUS}	Signed	DC Bus Voltage
Line voltage	V_{RMS}	Unsigned	Line-to-line voltage
List	List	Unsigned	None
Percent	Percent	Unsigned	1%
Percent/RPM	% / RPM	Signed	If system units are set to flow this will be a percent. If set to RPM this will be a speed.
Phase current	I_{RMS}	Unsigned	RMS phase current
Power	Power	Unsigned	Watts
Scale Factor	Scale	Unsigned	Scaling factor
Speed	Speed	Signed	RPM
Temperature(°C)	TC	Signed	Temperature in °C
Time(m)	Time(m)	Unsigned	Minutes
Time(s)	Time(s)	Unsigned	Seconds
Time(20u)	Time(20u)	Unsigned	20 Microseconds
Torque	Torque	Unsigned	Scaled torque in N•m
Version	Version	Unsigned	None

Some registers contain values that need to be scaled by an accompanying scale factor. Until the value is properly scaled, the register value will not reflect the actual value. Registers that require modification by a scaling factor are marked in the table(s) with (sc). For more information on how to obtain the various scaling factors, refer to the extended documentation.

5 Modbus Register Number vs. Address

The tables in the sections that follow show two numeric references for each register. By convention Modbus "holding" register numbers start at 40001 and "input" register numbers at 30001. This "register number" is just a convenient way to specify a specific location in a human readable form. The communications protocol does not directly use this number. When communicating between devices a request is made to access a resister by specifying the register "group" (which is either holding or input) and then the offset into that register table So the value in the "Addr" column below is the offset into that particular table and the "Reg." column shown the more human readable form for reference.

6 Holding Registers

Holding registers are 16-bit values that the application or external system can read from and write to. These registers can be read at any time but can only be written when the motor is stopped. There are a few exceptions that are not constrained by this rule and they are noted in the table which follows.

Reg.	Addr	Name	Type	Value	Range	Description
40001	0	Motor On/Off ***(writable even when motor is running)***	Binary		0,1	If the motor is currently off, writing a 1 here will turn it on using the manual speed rpm from register 40002 and acceleration from register 40004. If the motor is currently running, writing a zero here will turn it off.
40002	1	Manual speed RPM ***(writable even when motor is running)***	Speed		(see notes to the right)	Writing a new value here will set the RPM value for the motor. If the motor is currently running it will accelerate or decelerate to this new speed. If the motor is currently off then this speed will be used when register 40001 is set to 1. If the value written here is less than register 40034 {Min Output RPM} then it will be ignored and if the value written here is greater than register 40035 {Max Output RPM} then it will be ignored. The value written into this register will be translated into an equivalent percentage value and placed into register 40003 (overwriting the previous value) so that both registers are in sync. (Note: anytime a new flow % value is written to 40003 the equivalent RPM value is calculated and then stored here overwriting the previous value.)

† - This register can hold a percentage flow from 0 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

‡ - This register can hold a percentage flow from 1 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

Reg.	Addr	Name	Type	Value	Range	Description
40003	2	Manual flow % ***(writable even when motor is running)***	Percent		1-100	Writing a new value here will set the manual flow rate % for the motor. Any value written to this register that is outside the range of 1 to 100 will be ignored. The value written into this register will be translated into motor RPM and placed into register 40002 (overwriting the previous value) so that it is in sync with this register. If the motor is currently running it will accelerate or decelerate to the new speed in 40002. If the motor is currently off then the speed in 40002 will be used when register 40001 is set to 1.
40004	3	Acceleration rate ***(writable even when motor is running)***	Accel		1-500	This value sets the motor acceleration rate in RPM/sec. It can be updated at any time but will not be used until the next time the motor speed is changed via register 40002 or 40003.
40005 to 40026	——					See extended documentation for information on these registers
40027	26	Flow Percent or RPM	Binary		0,1	Limited use with Modbus – If 0 (zero) selected registers contain flow percent. If 1 (one) those registers contain an RPM value.
40028 to 40031	——					See extended documentation for information on these registers
40032	31	Prime Percent or RPM	%/RPM		†	Percent level or RPM for the motor to run at during the priming time. Often set to 100% or max RPM.
40033	32	Prime time	Time(m)	0 allowed to disable	0-9	Number of minutes for the priming cycle to run. If the value of this register is 0 then the priming step is skipped.

† - This register can hold a percentage flow from 0 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

‡ - This register can hold a percentage flow from 1 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

Reg.	Addr	Name	Type	Value	Range	Description
40034	33	Min Output RPM	Speed	Must be ≥ {Lower Speed Limit} and ≤ { Max Output RPM } - 99	(see notes to the right)	This register is similar to {Lower Speed Limit} but unlike {Lower Speed Limit} this register can be written to a larger value to increase the lowest speed the motor can run. When a flow is specified in percent this value is used in the calculation to convert percent flow to motor RPM. This register must be less than or equal to {Max Output RPM} -99. Any attempt to set it larger than that will be ignored. If this register is changed and the value in {Manual speed RPM} is less than the new setting for this register then {Manual speed RPM} is adjusted up to the new minimum speed.
40035	34	Max Output RPM	Speed	Must be ≥ {Min Output RPM} + 99 and ≤ {Upper Speed Limit}	(see notes to the right)	This register is similar to {Upper Speed Limit} but unlike {Upper Speed Limit} this register can be written to a smaller value to limit the fastest speed the motor can run. When a flow is specified in percent this value is used in the calculation to convert percent flow to motor RPM. This register must be greater than or equal to {Min Output RPM} +99. Any attempt to set it smaller than that will be ignored. If this register is changed and the value in {Manual speed RPM} is larger than the new setting for this register then {Manual speed RPM} is adjusted down to the new maximum speed.
40036 to 40039	——					See extended documentation for information on these registers
40040	39	Reverse	Binary		0,1	Sets the direction the motor turns in. Looking at the shaft end of the motor: 0 = Counter Clockwise Rotation 1 = Clockwise Rotation
40041 to 40047	——					See extended documentation for information on these registers
40048	47	Modbus Baud Rate	List	0=9600,1=19200	0,1	This value sets the Modbus baud rate

† - This register can hold a percentage flow from 0 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

‡ - This register can hold a percentage flow from 1 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

Reg.	Addr	Name	Type	Value	Range	Description
40049	48	Parity	List	0=None 1=Even 2=Odd	0-2	Selects the Modbus parity. If “None” is selected then 2 stop bits should also be selected in register 40050 in order to adhere to the Modbus specification. If strict adherence is not required 1 stop bit can be used. If parity is “Even” or “Odd” then only one stop bit should be used.
40050	49	Stop	List	0=1 Stop Bit 1=2 Stop Bits	0,1	Number of stop bits used on the Modbus serial line.
40051	50	Slave Id	Address	Factory default=240	1-247	Each slave on the RS-485 line must have a unique address in the range of 1 to 247. Units are shipped from the factory with this ID set to 240.
40052	51	1.5 Character Timeout	Time(20u)		1-65535	This should be the amount of time it takes for 1-1/2 characters to be transmitted on the serial line. Determining the proper setting for this register can be difficult so use of the default value is strongly recommended. More information regarding this register can be found in the extended register documentation. The value in this register is only read once when Modbus is first activated.
40053	52	3.5 Character Timeout	Time(20u)		1-65535	This register functions similarly to {1.5 Character Timeout} except the time delay is longer. Refer to the Description for {1.5 Character Timeout} for more information.
40054	53	Reset	Binary		0,1	Writing a ‘1’ to this register will cause a system reset to occur. There is a slight delay in order to allow the slave to respond back to the master before resetting.
40055	54	Save settings	Binary		0,1	Writing a ‘1’ to this register will force all holding registers with settings into NVM. Settings in the holding registers will not persist through a power cycle unless they are stored into NVM. After the master has updated any critical settings that need to be saved, this register should be set to ‘1’ to persist the values. The value in this register will automatically return to ‘0’ after the save operation completes.

† - This register can hold a percentage flow from 0 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

‡ - This register can hold a percentage flow from 1 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

Reg.	Addr	Name	Type	Value	Range	Description
40056	55	Communication timeout	Time(s)		1-65535	The motor is a slave device and expects to see some form of communication from the master periodically. If nothing is received from the master for a set number of seconds, the motor will be commanded off. This register sets the timeout in seconds. The communication timeout option must be enabled for this to be used, see the {Communication timeout scheme} register for details on timeout schemes.
40057	56	Communication timeout scheme	List	0=Direct address 1=Any broadcast 2=Any communication 3=Disable checking	0-3	The communication timeout timer (see the {Communication timeout} register) can be reset by different methods. If this setting is 0, then the slave must receive a valid message directed at the slave specific ID. In other words, the master must speak directly to this slave. If 1 then any valid broadcast message on the bus will reset the timeout as will any valid direct message. If 2, any valid communication detected on the bus will reset the timeout even if it's is directed to another slave. For this case, valid broadcasts or valid direct messages will reset the timeout too. If 3, then the timeout is disabled. "Valid" means the packet CRC is good and the packet conforms to Modbus Serial Line PDU format.
40058 and 40059	——					See extended documentation for information on these registers
40060	59	Max Acceleration	Accel	Must be > 0 and ≤ {Upper Accel Limit}	(see notes to the right)	By default this register will be equal to {Upper Accel Limit} but unlike {Upper Accel Limit} this register can be written to a smaller value to limit the motor acceleration. This register must be greater than zero and less than or equal to {Upper Accel Limit}. Values outside of that range will be ignored.
40061 to 40088	——					See extended documentation for information on these registers

† - This register can hold a percentage flow from 0 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

‡ - This register can hold a percentage flow from 1 to 100 or an RPM value depending on the setting of register **Error! Reference s ource not found.**

7 Input Registers

Input registers are 16-bit values that the application or external system read from (but not write to). These registers can be read at any time and return either fixed values (such as the speed limits) or variable input data to the system (such as the current motor speed).

Reg.	Addr	Name	Type	Value	Range	Description
30001	0	Current Speed	Speed		0-max*	Current motor speed in RPM. In normal use the upper end of the range for this register is {Max Output RPM} (40035). *max – maximum value varies with motor
30002	1	Current Flow %	Percent		1-100	Current motor speed from {Current Speed} register expressed as a flow percent
30003	___					See extended documentation for information on this register
30004	3	Motor Power	Power		0-max*	Output power from the motor control/input power to the motor in watts. *max – maximum value varies with motor
30005	___					See extended documentation for information on this register
30006	5	Motor Fault Status	List	0=Initializing 1=Set Volts 2=Motor Request 3=Motor Delay 4=Idle 5=Logging 6=Waiting 7=Stopped	0 – 7	Fault processing status value
30007	6	Motor Fault Code	List		0 – 65535	When a fault occurs, {Motor Fault Status} will typically change from 4 to either 6 or 7 depending on the fault. When that happens, this register will be set to the code that caused the fault. In the case of self-clearing faults, this register may revert to zero if the fault clears and the value in {Motor Fault Status} goes back to 4.
30008 to 30030	___					See extended documentation for information on these registers
30031	30	Interface Fault State	List	0=Initializing 4=Idle 5=Logging 6=Waiting 7=Stopped	0-7	Present state of the interface fault processing state machine. States 1,2 and 3 are reserved and not used in the current design.

Reg.	Addr	Name	Type	Value	Range	Description
30032	31	Interface Fault Code	Bit Image	Bit 0: Unused Bit 1: Modbus comm timeout Bit 2: Motor comm timeout Bit 3: UI comm timeout Bit 4: Reset DI to factory setting Bit 5: DI bad limit table Bit 6: DI bad options Bit 7: DI bad power loss Bit 8: DI bad template Bit 9 – Bit 15: Unused	0-65535	When a fault occurs, {Interface Fault State} will typically change from 4 to either 6 or 7 depending on the fault. When that happens, this register will be set to the code that caused the fault. When this value is non-zero the value in {Interface Fault State} must be checked as well. If it is 5 or 6 the fault is processing and may clear itself. Continue to monitor both {Interface Fault Code} and {Interface Fault State}. When {Interface Fault State} returns to 4 and {Interface Fault Code} changes to 0 then the fault is cleared. If {Interface Fault State} changes to 7 the fault is fatal and will not clear itself. Many faults will try indefinitely to clear themselves (i.e. communication faults) so the interface may appear to be hung at state 6 for a prolonged period of time. The master controller should manage this case by enforcing a timeout of its own.
30033 to 30113	——					See extended documentation for information on these registers
30114	113	Stopped State	Bit Image			Stopped status 0. Motor is not stopped 1. Motor is stopped
30115	114	V_{AC} input line volts	V_{RMS}			Measured V_{AC} input line voltage (V_{RMS} line-to-line)
30116	115	Control's input power	Power			Control's estimated input power (system power) in watts
30117	116	Motor Fault Status	Bit Image			**Bit Fault** 0 - power up reset 1 - watchdog reset 2 - motor stalled 3 - over current (HW) 4 - max power 5 - communication timeout 6 - under voltage 7 - over voltage 8 - heatsink backdown 9 - heatsink too hot 10 - over current (SW) 11 - open phase 12 - open and shorts test failed 13 - PFC too hot/cold 14 - eeprom CRC failed 15 - PFC backdown
30118	117	Inverter Temperature (sc)	Scaled TC			Inverter temperature in °C scaled by temperature scale **[signed]**
30119	118	Estimated shaft speed	Speed			Estimated shaft speed (looking at motor drive shaft side positive = CCW, negative = CW direction) **[signed]**
30120	119	Target shaft speed	Speed			Target shaft speed **[signed]**

Reg.	Addr	Name	Type	Value	Range	Description
30121	120	Motor RMS phase current (sc)	Scaled I_{RMS}			Motor RMS phase current scaled by phase current scale
30122	121	Motor RMS line-to-line voltage	V_{RMS}			Motor RMS line to line voltage (no scaling)
30123	122	Motor input power in watts	Power			Motor input power in watts (no scaling) **[signed:+]**
30124	123	Torque in N•m (sc)	Torque			Torque in N•m scaled by Torque scale
30125	124	Horsepower (sc)	Power			HP scaled by HP scale
30126	125	On-board NTC temperature (sc)	Scaled TC			More accurate at low temperatures than {Inverter Temperature}. Uses same units, °C scaled by temperature scale **[signed]**
30127	126	Measured V_{dc} bus	V_{BUS}			DC bus voltage (no scale) **[signed:+]**
30128	127	MCU State				MCU state 0. Reset state 1. Pre-charge delay state 2. Stop state 3. Fault state 4. Run state

[signed] – value in register is signed, could be positive or negative
[signed:+] – value in register is signed, typically positive

8 Diagnostics

The Modbus specification defines special diagnostic counters and procedures for monitoring and testing the Modbus communication sub-system. Technically these are not "registers" in the same sense of holding registers and input registers but they function in a way that closely parallels the standard register groups.

The Modbus interface for Nidec variable speed motors implements a subset of the Diagnostics outlined in section 6.8 of the Modbus Application Protocol Specification V1.1b3 specification. The table below indicates which features are supported:

Sub-function code		Name	Implemented
Hex	Dec		
00	00	Return Query Data	✓
01	01	Restart Communications Option	✓ *
02	02	Return Diagnostic Register	‡
03	03	Change ASCII Input Delimiter	‡
04	04	Force Listen Only Mode	✓
	05 … 09	RESERVED	—
0A	10	Clear Counters and Diagnostic Register	✓
0B	11	Return Bus Message Count	✓
0C	12	Return Bus Communication Error Count	✓
0D	13	Return Bus Exception Error Count	✓
0E	14	Return Server Message Count	✓
0F	15	Return Server No Response Count	†
10	16	Return Server NAK Count	†
11	17	Return Server Busy Count	†
12	18	Return Bus Character Overrun Count	✓
13	19	RESERVED	—
14	20	Clear Overrun Counter and Flag	‡
N/A	21 … 65535	RESERVED	—

✓ – Supported
* – Ignores contents of data field, treats data field as 00 00 (see Modbus spec for details)
† – Not supported, always returns zero
‡ – Returns Illegal Function exception response when read