FUTUREDESIGN CONTROLS

nCompass CM

User Communications Reference Manual

nCompass CM User Communications Reference Manual Rev B
July 2019 (V2.0)
Supersedes: August 2014

Safety Information in this Manual

Notes, cautions and warnings appear throughout this book to draw your attention to important operational and safety information.

A "**NOTE**" marks a short message to alert you to an important detail.

A "**CAUTION**" safety alert appears with information that is important for protecting your equipment and performance.

A "**WARNING**" safety alert appears with information that is important for protecting you, others and equipment from damage. Pay very close attention to all warnings that apply to your application.

This symbol (an exclamation point in a triangle) precedes a general CAUTION or WARNING statement.

This symbol (a lightning bolt in a lightning bolt in a triangle) precedes an electric shock hazard CAUTION or WARNING safety statement.

Technical Assistance

If you encounter a problem with your nCompass controller, review all of your configuration information to verify that your selections are consistent with your application: inputs; outputs; alarms; limits; etc. If the problem persists after checking the above, you can get technical assistance by dialing +1 (866) 342-5332 or by faxing your request to +1 (866) 332-8014, Monday thru Friday, 8:00 a.m. to 5:00 p.m. Eastern Standard Time. You can also email your request to support@futuredesigncontrols.com.

An applications engineer will discuss your application with you.

Please have the following information available:

- Complete Model #'s and/or Serial #'s for Component(s) in Question
- Complete Software Version #'s
- All Configuration Information
- All User Manuals

Warranty and return information is on the back cover of this manual.

Your Comments

Your comments or suggestions on this manual are welcome. Please send them to:
Future Design Controls, P.O. Box 1196, Bridgeview, Illinois, 60455
Telephone: +1 (888) 751-5444; fax: +1 (888) 307-8014
csr@futuredesigncontrols.com

The nCompass CM User Communications Reference Manual is copyrighted by Future Design Controls, Inc., © 2014, all rights reserved (http://www.futuredesigncontrols.com/nCompass.htm).

1 What is the nCompass?

The nCompass system combines all of the features of typical loop controllers, video/chart recorders and datalogging systems into a single/intuitive device. Email, SMS (text messaging), FTP (file transfer protocol for automated data backup) and remote view/control (Web server/VNC server) are standard with nCompass and can be accessed via LAN/WAN using a PC, tablet or smart phone device.

Future Designs “nCompass” provides a 4.3”color touch screen interface with standard “Smart Device” user interface features for up to ten loop OEM control applications. All loop configuration and runtime user access is configurable at the device with no PC software required. OEM’s have the ability to configure runtime features (screens available, menus, etc...) to easily customize the system for their requirements. These configurations can be imported/exported to any nCompass device for setup (from scratch) within minutes.

In addition to a maximum of ten loops of control, nCompass can also provide up to an additional 15 inputs for process monitoring for a total of 25 process inputs. The system is provided with eight 24Vdc digital inputs, two 24Vdc outputs and 6 relay outputs standard. nCompass can be expanded to a total of 16 digital inputs and 32 digital outputs. nCompass also provides the capability of accepting analog inputs for remote set point control and analog outputs capable of retransmitting system variables (PV, SP or %Out) to other devices such as a chart recorder. The 0-10Vdc or 4-20mA user selectable signals are provided through the addition of optional analog expander cards.

Individual process controllers, one for each loop in the system, provide reliable, consistent and accurate control by distributing the process control requirements of the system among multiple processors. Each loop controller provides full auto tune functionality with high resolution, universal process inputs. When coupled with the built in ramping programmer of nCompass, it allows for automatic, timed control of all processes and outputs of the system.

1.1 Features

The nCompass digital inputs can be configured as alarm inputs with adjustable delay timers, as control inputs for controlling automatic program operation or for direct control of the system’s digital outputs.

The nCompass digital outputs can be used as direct outputs for controlling external equipment related to the application through software switches, called events, or be programmed to act as system alarm or status outputs. All outputs have adjustable delay times for on, off and cycle times.

nCompass can be operated in single set point or automatic program control mode. Program entry is made easy through the use of copy, paste and delete menu selections. Programs can be copied to the external ‘USB’ memory stick and then imported to another nCompass controller which eliminates the need to enter duplicate ramp/soak programs into multiple systems. When running in automatic program mode, the operator can place the system into hold and change any control parameter without modifying the original program. This gives the operator maximum flexibility over the controlled process.

Data file analysis tools (auto-trend) make looking at historical data a simple task. Any control variable saved to the nCompass SD memory can be plotted on the historical data trend, for any time frame within the data file’s total time range.

The built in Ethernet functionality includes a ‘Web Server’ to provides access to all nCompass data (view only), a VNC interface for remote control and monitoring and an NTS clock, all available via a local Intranet connection (wired or wireless), or the World Wide Web using standard software like Microsoft’s Internet Explorer.

nCompass provides a rich set of tools for control interaction and data analysis. Views include system overviews, trends, alarms, automatic programs as well as historical data, alarm history and audit trail views. The menu driven interface eliminates screen 'clutter' by providing an easy to use 'Smart Device' interface for interaction between the user and nCompass.

nCompass can store more than one year of data on its SD memory card. Data logging can be enabled manually or automatically during automatic program operation. Data backup is provided with the 'USB I-Stick' for plug and play transfer of files to any PC running Microsoft Windows XP operating systems and via the FTP back-up utility.

nCompass protects system access with 4 level security (user rights based), audit trails that document all user activity and ensures data integrity by digitally signing all data files and audit trails to meet regulatory requirements.

The nCompass controller includes the following features:

- Single/Dual loop controller models.
- Touch screen, "Smart Device" user interface (UI).
- Email, SMS, FTP, VNC and Web functionality standard.
- Remote View/Control using PC, Tablet or Smartphone.
- Detailed maintenance, alarm monitoring and alarm history.
- User configurable data logging and historical data viewer.
- 4 level security with digitally signed audit trails and data files.
- National time server connectivity with daylight savings.
- Multi-lingual user interface supports over 25 languages.
- 30,000 hour LED display

2 Optional User Communications Card Installation

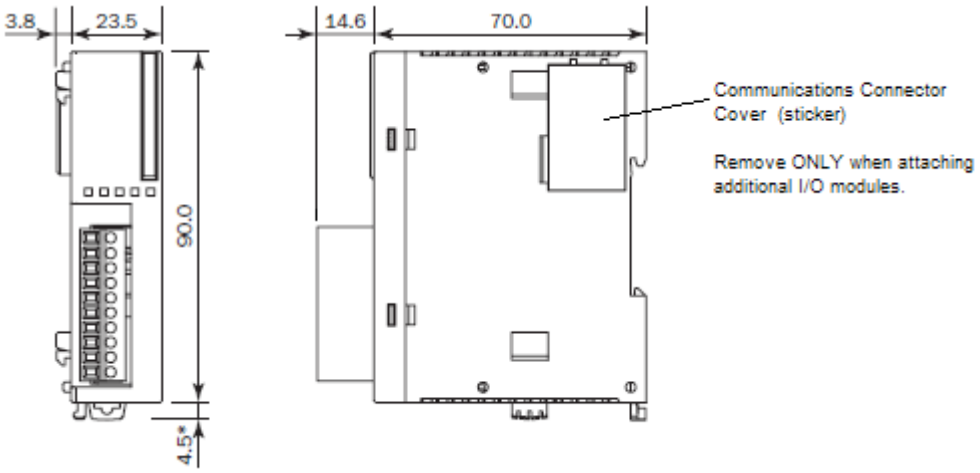
WARNING:

- To avoid potential electric shock and other hazards, all mounting and wiring for nCompass must conform to the National Electric Code (NEC) and other locally applicable codes.
- Special expertise is required to install, wire, configure and operate the nCompass controller. Personnel without such expertise should not install, wire or operate nCompass.

CAUTION:

- Prevent metal fragments and pieces of wire from dropping inside the housing of any nCompass component. If necessary, place a cover over the component during installation and wiring. Ingress of such fragments and chips may cause a fire hazard, damage or malfunction of the device.
- Locate the nCompass and all related control components away from AC power/motor wiring and sources of direct heat output such as transformers, heaters or large capacity resistors.

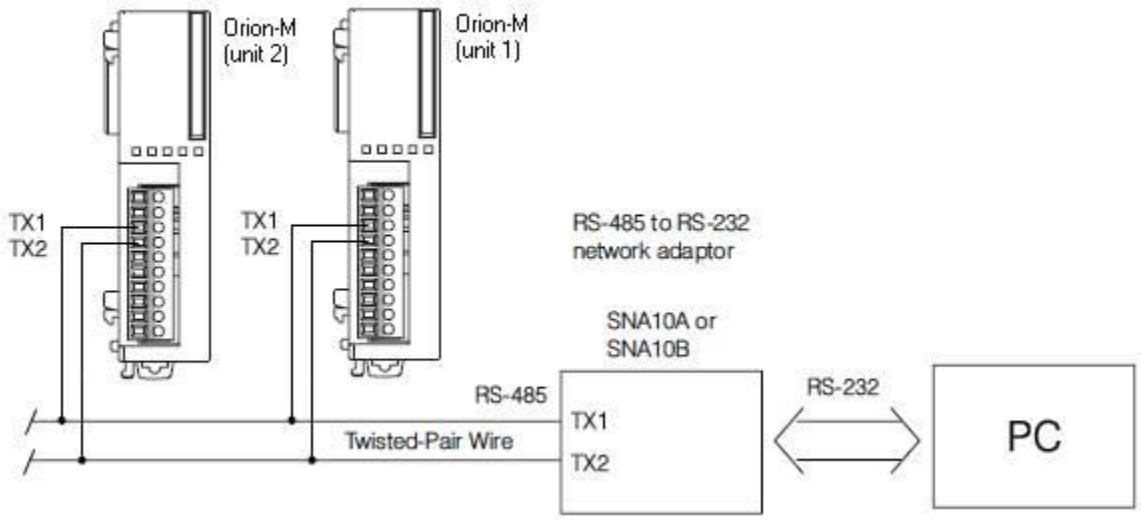
The nCompass user interface provides an RS232/422/485 interface as a standard feature. See section 3.3.2, User Serial Communications in the nCompass User Manual for information. In addition to the standard interface, a secondary, optional user communications interface, an RS-485 (FC5A-SIF4) communications card can be installed on the nCompass control module. This would allow two master devices to communicate with nCompass at the same time.



IMPORTANT: *If nCompass is also equipped with the optional RS-232 (FC4A-SIF2) communications interface for a barcode scanner, the RS-485 (FC5A-SIF4) user communications card must be the first communications card installed on the control module. If the RS-232 card is installed first, nCompass will not operate properly when power is applied and the red error (ERR) light on the control module will illuminate.*

2.1 Optional User Communications Card Wiring

The RS485 communications card (FC5A-SIF4) allows multiple nCompass controllers to be connected to a single communications link. The connection requires a single twisted-pair cable that is daisy-chained from one nCompass to the next. Use of this card requires an RS485 connection on the host device. Since most computers do not provide this type of interface, an RS232 to RS485 adapter is required. Future Design recommends the use of the SNA10A or SNA10B network adapter.



NOTE: *When using shielded twisted-pair cable, be sure to ground only when end of the cable, preferably at the RS232 to RS485 network adapter. Allowing any other portion of the cable shield to come in contact with ground, or grounding both ends, will cause ground loop currents to flow in that section of the cable which can cause communication errors.*

3 Communication Basics

The purpose of this document is to provide users interested in using data communications with the nCompass controller, the ability to set up and use a simple network of one or more nCompass controller(s) by providing a basic understanding of data communications using standard definitions, interfaces and protocols.

In this manual, numbers in the format '0x00' represent values in hexadecimal. Numbers in the format '0' represent values in decimal and finally, numbers in the format '00000000' represent values in binary unless otherwise stated.

3.1 Explanation of Terms

Machine-to-Machine Communication
In order for machines to communicate with each other, they need a code called a character format or character set. They require rules called protocol to govern their conversation and prevent confusion and errors. Computers need a connecting interface over which to communicate. They may use one pair of wires to send information in one direction and another pair to send in the opposite direction (full duplex), or they may use one pair to send data in both directions (half duplex).

Character Format
The code or character format for the nCompass data communications is shared by virtually everyone in the electronics industry. This code defines a stream of 1's and 0's that are created by varying a voltage signal in a regular manner. This code is the American Standard Code for Information Interchange, called ASCII.

Bits and Bytes
The word "bit" is simply the contraction of the words **bi**nary digi**t**. A bit is the basic unit in ASCII. It is either a "1" or a "0". A byte is a string of eight bits that a computer treats as a single character. ASCII can use a single byte to represent each letter of the alphabet, each digit and each punctuation mark we use.

ASCII
The ASCII code defines 128 separate characters, one for each letter, digit and punctuation mark. ASCII also includes control characters similar to those we find on computer keys, such as backspace, shift and return. It also has nine communications control characters for identification, enquiry (inquiry), start of text, end of text, end of transmission, acknowledge, negative acknowledge and escape. The ASCII code is sometimes written in a base 16 number system that is called hexadecimal or "hex" for short. The numbers 0 through 9 represents the first ten digits of this system, and the letters A through F represents the final six digits. The 128 ASCII character codes with the decimal, binary and hexadecimal equivalents are listed in the following table.

ASCII Control Codes
ASCII Control Codes are used to give instructions to the remote device and result in specific actions, such as a line feed instruction on a printer. ASCII Control Codes, the first 33 ASCII characters (non printable), are important for the operation of communicating equipment. They give instruction to remote devices that result in specific actions such as a line feed on a printer. Holding down the keyboard control key while pressing the appropriate keyboard key is what sends these values.

ASCII Character Chart

Char	Code	Decimal	Binary	Hex	Char	Code	Decimal	Binary	Hex
NUL	Ctrl @	0	00000000	00	**@**	Shift 2	64	01000000	40
SOH	Ctrl A	1	00000001	01	**A**	Shift A	65	01000001	41
STX	Ctrl B	2	00000010	02	**B**	Shift B	66	01000010	42
ETX	Ctrl C	3	00000011	03	**C**	Shift C	67	01000011	43
EOT	Ctrl D	4	00000100	04	**D**	Shift D	68	01000100	44
ENQ	Ctrl E	5	00000101	05	**E**	Shift E	69	01000101	45
ACK	Ctrl F	6	00000110	06	**F**	Shift F	70	01000110	46
BEL	Ctrl G	7	00000111	07	**G**	Shift G	71	01000111	47
BS	Ctrl H	8	00001000	08	**H**	Shift H	72	01001000	48
TAB	Ctrl I	9	00001001	09	**I**	Shift I	73	01001001	49
LF	Ctrl J	10	00001010	0A	**J**	Shift J	74	01001010	4A
VT	Ctrl K	11	00001011	0B	**K**	Shift K	75	01001011	4B
FF	Ctrl L	12	00001100	0C	**L**	Shift L	76	01001100	4C
CR	Ctrl M	13	00001101	0D	**M**	Shift M	77	01001101	4D
SO	Ctrl N	14	00001110	0E	**N**	Shift N	78	01001110	4E
SI	Ctrl O	15	00001111	0F	**O**	Shift O	79	01001111	4F
DLE	Ctrl P	16	00010000	10	**P**	Shift P	80	01010000	50
DC1	Ctrl Q	17	00010001	11	**Q**	Shift Q	81	01010001	51
DC2	Ctrl R	18	00010010	12	**R**	Shift R	82	01010010	52
DC3	Ctrl S	19	00010011	13	**S**	Shift S	83	01010011	53
DC4	Ctrl T	20	00010100	14	**T**	Shift T	84	01010100	54
NAK	Ctrl U	21	00010101	15	**U**	Shift U	85	01010101	55
SYN	Ctrl V	22	00010110	16	**V**	Shift V	86	01010110	56
ETB	Ctrl W	23	00010111	17	**W**	Shift W	87	01010111	57
CAN	Ctrl X	24	00011000	18	**X**	Shift X	88	01011000	58
EM	Ctrl Y	25	00011001	19	**Y**	Shift Y	89	01011001	59
SUB	Ctrl Z	26	00011010	1A	**Z**	Shift Z	90	01011010	5A
ESC	Ctrl [	27	00011011	1B	**[**	[	91	01011011	5B
FS	Ctrl \	28	00011100	1C	****	\	92	01011100	5C
GS	Ctrl]	29	00011101	1D	**]**	]	93	01011101	5D
RS	Ctrl ^	30	00011110	1E	**^**	Shift 6	94	01011110	5E
US	Ctrl _	31	00011111	1F	**_**	Shift -	95	01011111	5F
SP	SPACE	32	00100000	20	**`**	`	96	01100000	60
!	Shift 1	33	00100001	21	**a**	A	97	01100001	61
"	Shift ‘	34	00100010	22	**b**	B	98	01100010	62
#	Shift 3	35	00100011	23	**c**	C	99	01100011	63
$	Shift 4	36	00100100	24	**d**	D	100	01100100	64
%	Shift 5	37	00100101	25	**e**	E	101	01100101	65
&	Shift 7	38	00100110	26	**f**	F	102	01100110	66
‘	‘	39	00100111	27	**g**	G	103	01100111	67
(	Shift 9	40	00101000	28	**h**	H	104	01101000	68
)	Shift 0	41	00101001	29	**I**	I	105	01101001	69
*****	Shift 8	42	00101010	2A	**j**	J	106	01101010	6A
+	Shift =	43	00101011	2B	**k**	K	107	01101011	6B
,	,	44	00101100	2C	**l**	L	108	01101100	6C
-	-	45	00101101	2D	**m**	M	109	01101101	6D
.	.	46	00101110	2E	**n**	N	110	01101110	6E
/	/	47	00101111	2F	**o**	O	111	01101111	6F
0	0	48	00110000	30	**p**	P	112	01110000	70
1	1	49	00110001	31	**q**	Q	113	01110001	71
2	2	50	00110010	32	**r**	R	114	01110010	72
3	3	51	00110011	33	**s**	S	115	01110011	73
4	4	52	00110100	34	**t**	T	116	01110100	74
5	5	53	00110101	35	**u**	U	117	01110101	75
6	6	54	00110110	36	**v**	V	118	01110110	76
7	7	55	00110111	37	**w**	W	119	01110111	77
8	8	56	00111000	38	**x**	X	120	01111000	78
9	9	57	00111001	39	**y**	Y	121	01111001	79
:	Shift ;	58	00111010	3A	**z**	Z	122	01111010	7A
;	;	59	00111011	3B	**{**	Shift [	123	01111011	7B
<	Shift ,	60	00111100	3C	**|**	Shift \	124	01111100	7C
=	=	61	00111101	3D	**}**	Shift]	125	01111101	7D
>	Shift .	62	00111110	3E	**~**	Shift `	126	01111110	7E
?	Shift /	63	00111111	3F	**DEL**	Delete	127	01111111	7F

4 Serial Communication

The user communications interface for nCompass employs serial communication, which is the exchange of data in a one-bit-at-a-time, sequential manner on a single data line or channel. Serial contrasts with parallel communication, which sends several bits of information simultaneously over multiple lines or channels. Not only is serial data communication simpler than parallel, it is also less costly.

Baud Rate

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (BPS). However, if you connect two serial devices together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 9600 BPS, then the line is also changing states 9600 times per second.

Typical baud rates used for computers are 9600, 14400, 19200, 38400 and 57600 baud. As the baud rate increases, so does the transmission rate of data. Thus you get more information in a shorter period of time. However, the faster the transmission rate, the more susceptible it is to error due to the quality of the cable and sources of electrical “noise” in the environment. In order to balance throughput with reliability, nCompass uses a 9600 baud rate. *Thus a device used to communicate with the nCompass must have its serial port set for 9600 baud in order for data communications to work properly.*

Start and Stop Bits

The start bit informs the receiving device that a character is coming, and a stop bit tells it that a character is complete. The start bit is always a 0. The stop bit is always a 1. The human speech equivalent of these bits could be a clearing of the throat to get someone’s attention (start bit); and a pause at the end of a phrase (stop bit). Both help the listener understand the message.

A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit's duration. Stop bits can be 1, 1.5, or 2 bit periods in length. The nCompass uses the default – and most common – length of 1 period for the stop bit. *A device used to communicate with the nCompass must also have its serial port set to use a stop bit of 1 in order for data communications to work properly.*

Parity Bit

Besides the synchronization provided by the use of start and stop bits, an additional bit called a parity bit may optionally be transmitted along with the data. A parity bit affords a small amount of error checking, to help detect data corruption that might occur during transmission. There are several defined parity selections available for serial communications. They are even parity, odd parity, mark parity, space parity or none at all can be used. When even or odd parity is being used, the number of marks (logical 1 bits) in each data byte are counted, and a single bit is transmitted following the data bits to indicate whether the number of 1 bits just sent is even or odd.

For example, when even parity is chosen, the parity bit is transmitted with a value of 0 if the number of preceding marks is an even number. For the binary value of 0110 0011 the parity bit would be 0. If even parity were in effect and the binary number 1101 0110 were sent, then the parity bit would be 1. Odd parity is just the opposite, and the parity bit is 0 when the number of mark bits in the preceding word is an odd number. Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Since these two parity options serve no useful purpose whatsoever, they are almost never used. *The nCompass is set for even parity. Thus, a device used to communicate with the nCompass must also have its serial port set to use the same parity setting (even) in order for data communications to work properly.*

4.1 Interface Standards

An interface is a means for electronic systems to interact. It's a specific kind of electrical wiring configuration. It has nothing to do with how data is sent over that connection. The two most common interfaces used today are RS-232, which provides a simple 1 to 1 connection and RS-485, which provides a multi-drop connection where more than one device can be placed on the same line. The nCompass standard communications interface provides both options; however, only one can be wired and used at a time. The optional communications card interface provides an RS-485 connection only.

EIA-232 (Full Duplex)

An EIA-232 (formerly RS-232C) interface uses three wires: a single transmit wire; a single receive wire; and a common line. Only two devices can use an EIA-232 interface. A -3 to -24 volt signal indicates a 1 and a +3 to +24 volt signal indicates a 0. The EIA-232 signal is referenced to the common line rather than to a separate wire, as in EIA-485. Thus, an EIA-232 cable is limited to a maximum of 50 feet, due to noise susceptibility.

EIA-485 (Half Duplex)

An EIA-485 interface uses two wires: a T/R+, a T/R- line. A -5-volt signal is interpreted as a 1, a +5-volt signal as a 0. As many as 31 slave devices can be connected to a master on a multi-drop network up to 4000 feet long.

Wiring

Most PCs have a standard EIA-232 port (usually referred to as RS-232). In these instances, you must use an interface converter to connect to an EIA-485 multi-drop system. For this connection, the terminals on most converters marked "TX1/RX1 or T+/R+" connect to terminal "A" of the RS-485 card and the terminals marked "TX2/RX2 or T-/R-" connect to terminal "B" of the RS-485 card. The standards do not specify the wire size and type. Use of 24 AWG twisted pair provides excellent results. If shielded cable is used, terminate the shield at one end only. Always follow the manufacturer's instructions supplied with the interface converter. See Biasing of Buses next.

Biasing of Buses

The EIA-485 standard requires the bus to be biased for reliable communication. This requires termination resistors to be placed across the T/R+ and T/R- wires. One resistor is placed at the PC where it connects to the EIA-485 bus. The second resistor is placed at the last controller on the network. Do not place resistors at each controller. The impedance of the wires used for the bus determines the resistor value. For twisted pair, the value is typically 120 ohms. In addition, it may be necessary to have a pull-up and pull-down resistor between the power supply and ground of the interface adapter.

Check the documentation that came with your interface adapter. Biasing the bus reduces reflection of signals sent down the bus. These reflections are sometimes referred to as a standing wave. This condition is most notable when communicating at high baud rates over longer distances.

4.1.1 Interface Converters

The purpose of an interface converter is to allow two different buses to be connected together. Interface converters are required when connecting an EIA-232 port to an EIA-485 bus. The EIA-485 bus is a half-duplex bus. This means that it can only send or receive data at any given time. Some interface converters on the market provide the ability to have full duplex with the EIA-485 bus. This is accomplished by using two receivers and transmitters tied in tandem. This type of converter will not work with the nCompass controller. Be sure that the model you purchase is designed for half duplex.

Another consideration when selecting an interface converter is how the converter handles switching between transmit and receive. Typically it is accomplished via a handshake line from the PC. When data flows into the converter from the PC, a handshake line is placed high. When data flows out of the converter to the PC, the handshake line is placed low. In this way, the handshake line controls the direction of information. Another method of achieving this is to use a built-in timer. The converter switches to transmit when a character is sent to it from the PC. After a period of time when the PC has not transmitted, the converter switches to a receive mode.

It is important that you understand how your converter accomplishes this task. You are required to wire this feature or make settings on the converter to enable this function. The PC will not talk to the controller correctly without properly setting this. Your converter may also require settings through dip switches, to set up communications parameters like baud rate, data bits, start bits, stop bits and handshaking. The converter may also require a separate power supply. Some converters get their power from the handshake lines of the PC. If you rely on this method, you will need to wire these additional lines. In addition, your software must set these lines high. A more reliable method is to use an external power supply. This is especially necessary when using a laptop computer. See the documentation that is provided with your converter for more information.

Not all converters are equal in performance. If your chamber operates in a harsh, electrically noisy environment, this can cause less robust converters to work intermittently or not at all. The following converter has been tested and is compatible with the nCompass. The converter is equipped with automatic send data control circuits, driver control in the converter hardware, so you don't have to work with software at all. The circuit monitors data flow and enables the driver during transmission and automatically disables it when no data is being sent. There is no need to rework software or install new drivers.

Future Design Controls
7524 West 98th Place
Bridgeview, IL 60455
Phone: 888-751-5444
Fax: 888-307-8014
E-mail: csr@futuredesigncontrols.com
www.futuredesigncontrols.com

Part # **SNA10A** Smart Network Adapter
Part # **DB9M-DB9F-6ft** (Cable Accessory to connect SNA10A to PC)

4.2 Protocol

Protocol describes how to initiate an exchange. It also prevents two machines from attempting to send data at the same time. There are a number of different data communications protocols, just as there are different human cultural protocols that vary according to the situation.

The protocol portion of nCompass communications is very important, because it provides a quality of communication that others often don't have. Protocol-driven communications are more accurate, because they are less prone to both operator and noise errors. Protocol maintains system integrity by requiring a response to each message. It's like registered mail — you know that your letter has been received because the post office sends you a signed receipt.

In nCompass data communications, a dialog will continue successfully as long as the messages are in the correct form and responses are returned to the protocol leader. If the operator enters an incorrect message, or interference comes on to the data line, there will be no response. In that case the master must retransmit the message or go to a recovery procedure. If an operator continues to enter an incorrect message or interference continues on the data line, the system will halt until the problem is resolved. nCompass uses Modbus RTU as the protocol of choice. Modbus RTU enables a PC to read and write directly to registers containing the nCompass parameters. With it, you can read all of the controller's parameters with just a few read commands.

Modbus Remote Terminal Unit (RTU)
Gould Modicon, now called AEG Schneider, created this protocol for process control systems called "Modbus". It has the advantage over other protocols of being extremely reliable in exchanging information. This protocol works on the principle of packet exchanges. The packet contains the address of the controller to receive the information, a command field that says what is to be done with the information and several fields of data. The last item sent in the packet is a field to ensure the data is received intact. This is called a cyclic redundancy check-sum. See the following example for information on how to generate this value. All information is exchanged in hex numbers. nCompass only supports the binary version of Modbus, referenced as RTU. The ASCII version is less efficient and is not supported. Therefore, you must be certain to format all data in hexadecimal.

The CRC (Cyclical Redundancy Checksum) is calculated by the following steps:

1. Load a 16-bit register (called CRC register) with 0xFFFF

2. Exclusive OR the first 8-bit byte of the command message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.

3. Shift the CRC register one bit to the right with MSB zero filling. Extract and examine the LSB.

4. If the LSB of the CRC register is zero, repeat step 3, else Exclusive OR the CRC register with the polynomial value 0xA001.

5. Repeat steps 3 and 4 until eight shifts have been performed. When this is done, a complete 8-bit byte will have been processed.

6. Repeat steps 2 through 5 for the next 8-bit byte of the command message. Continue doing this until all bytes of the command message have been processed. The final contents of the CRC register is the CRC value.

When transmitting the CRC value in the message, the upper and lower bytes of the CRC value must be swapped, i.e. the lower order byte will be transmitted first.

Example Cyclical Redundancy Checksum (CRC) Algorithm

```
unsigned int calc_crc(unsigned char *start_of_packet, unsigned char *end_of_packet)
{
unsigned int crc;
unsigned char bit_count;
unsigned char *char_ptr;

/* Start at the beginning of the packet */
char_ptr = start_of_packet;
/* Initialize CRC */
crc = 0xFFFF;
/* Loop through the entire packet */
do{
     /* Exclusive-OR the byte with the CRC */
     crc ^= (unsigned int)*char_ptr;
     /* Loop through all 8 data bits */
     bit_count = 0;
     do{
          /* If the LSB is 1, shift the CRC and XOR the polynomial mask with the CRC */
          if(crc & 0x0001){
               crc >>= 1;
               crc ^= 0xA001;
               }
          /* If the LSB is 0, shift the CRC only */
          else{
          crc >>= 1;
          }
     } while(bit_count++ < 7);
} while(char_ptr++ < end_of_packet);
return(crc);
}
```

4.3 Creating your own Modbus Application

Listed below are a few of the more common software packages that claim to support the Modbus protocol. This list is provided as informational only. Contact the software manufacturer for more information on applying their software.

LabView by National Instruments
11500 N Mopac Expwy
Austin, TX 78759-3504
Phone 800-683-8411
http://www.natinst.com

Wonderware by Wonderware
26561 Rancho Pkwy. South
Lake Forest, CA 92630
Phone 949-727-3200
http://www.wonderware.com

SpecView by SpecView Corporation
13409 53rd Ave NW
Gig Harbor, WA 98332
Phone 253-853-3199
http://www.specview.com

For additional information and support, modbus.org (http://modbus.org/tech.php) provides an abundance of technical documents, standards and support applications. If you already have a software application that uses Modbus, you can simply skip to the nCompass parameter table in the Getting Started section for the information your program requires. The rest of this section provides information on writing a software application that uses Modbus.

1. You must code messages in eight-bit bytes, with even parity, one stop bit (8, even, 1). nCompass has its parity set to even as default from the factory.

2. Negative parameter values must be written in twos' complement format. Parameters are stored in two-byte registers accessed with read and write commands to a relative address.

3. Messages are sent in packets that must be delimited by a pause at least as long as the time it takes to send 28 bits (3.5 characters). To determine this time in seconds, divide 28 by the baud rate. In the case of nCompass communications at 9600 baud, this calculates to a minimum period of ~3ms.

4. Values containing decimal points such as process values and setpoints, have the decimal point implied, i.e., the data exchange can only be performed using whole numbers. Thus, the value must be scaled appropriately in order to exchange the data correctly. For example, a setpoint of 78.4 degrees must be sent as a value of 784 in order for the nCompass to be set correctly. Likewise, a process value read from nCompass with a value of 827 is actually 82.7 degrees. Consult the parameter table for the proper format and allowable range of each value.

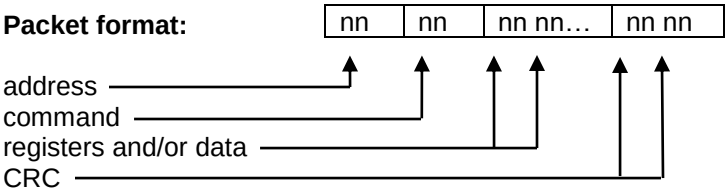
Handling Communication Errors
Messages with the wrong format or illegal values will receive an exception response. Messages with the wrong CRC or timing will receive no response. It is the user's responsibility to handle the error appropriately within their own software and determine whether to resend the message or halt for operator intervention.

User Responsibility
Refrain from altering prompts that do not appear on the nCompass front panel or are not included on the specific model. Refrain from reading or writing from/to a register that does not exist or is currently disabled. Writing values to unassigned registers could cause system instability, malfunction or failure. Care must also be taken in that the process cannot cause damage to property or injury to personnel if the wrong commands are sent due to operator error or equipment malfunction.

4.3.1 Packet Syntax

Each message packet begins with a one-byte controller address, from 0x01 to 0x1F. The second byte in the message packet identifies the message command: read (0x03); write single (0x06) or write multiple (0x10). The next "n" bytes of the message packet contain register addresses and/or data. The last two bytes in the message packet contain a two-byte Cyclical Redundancy Checksum (CRC) for error detection.

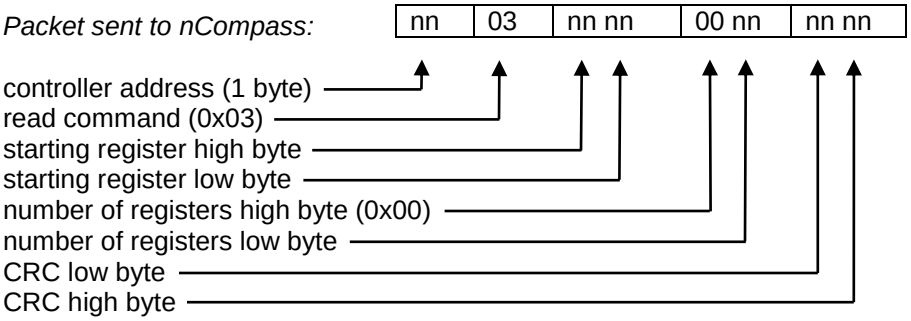
Packet format:



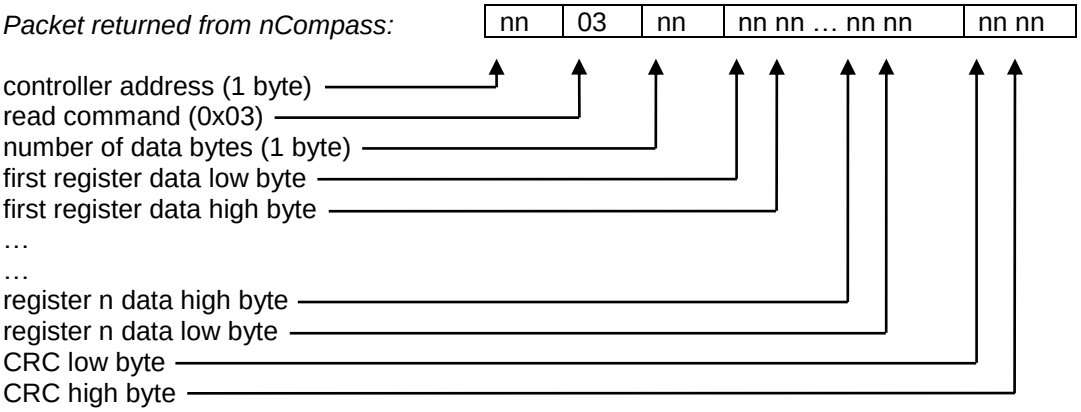
Read Register(s) Command (0x03)

This command returns from 1 to 64 registers. This command is used for reading one or more data locations from nCompass.

Packet sent to nCompass:



Packet returned from nCompass:



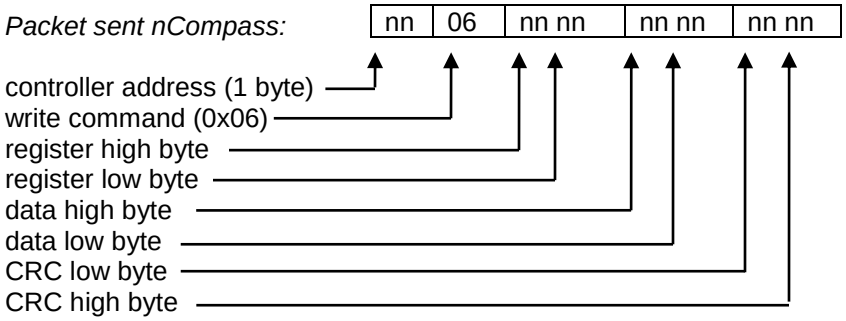
Example: Read registers 60 and 61 (loop 1 process variable and setpoint) of controller at address 1 configured for 1 decimal point.

Sent: 01 03 00 3C 00 02 04 07
Received: 01 03 04 **03 0D 01 F3** 2A 61

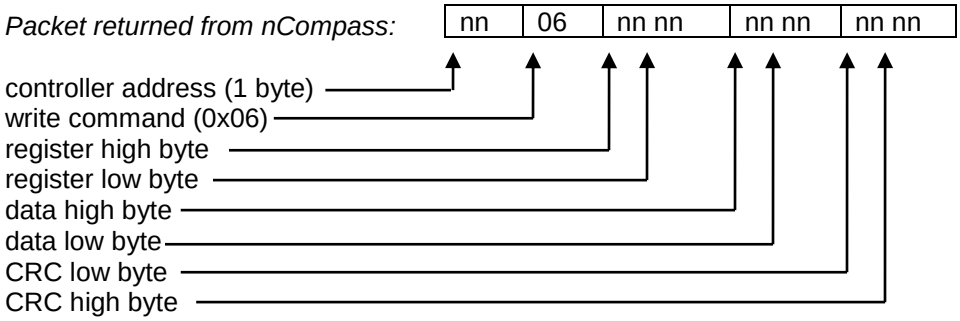
Message data: 781 (0x**030D**) = process variable of 78.1
499 (0x**01F3**) = setpoint of 49.9

Write Register Command (0x06)
This command writes a value to a single register. This command is used for setting a single control value in nCompass

Packet sent nCompass:



Packet returned from nCompass:



Example: Write register 67 (loop 3 setpoint) of controller at address one configured with no decimal point to 75 degrees (0x**004B**).

Sent: 01 06 00 43 **00 4B** 38 29
Received: 01 06 00 43 00 4B 38 29

Write Registers Command (0x10)

This command writes values to multiple registers in sequential order. This command can be used for setting multiple control values in nCompass using a single command.

IMPORTANT: *This command is for use with automatic ramp/soak program download only. It is used to transmit program data one step at a time to nCompass. See the Automatic Ramp/Soak Program Parameters section for the list of registers and their use. If this command is used to write to registers other than the correct program step registers, nCompass will respond with an acknowledge that the message was received; however, the command will not be executed.*

Packet sent to nCompass:

nn	10	nn nn	nn nn	nn	nn nn … nn nn	nn nn

controller address (1 byte)
write command (0x10)
starting register high byte
starting register low byte
number of registers to write high byte
number of registers to write low byte
number of data bytes
data high byte
data low byte
…
…
register n data high byte
register n data low byte
CRC low byte
CRC high byte

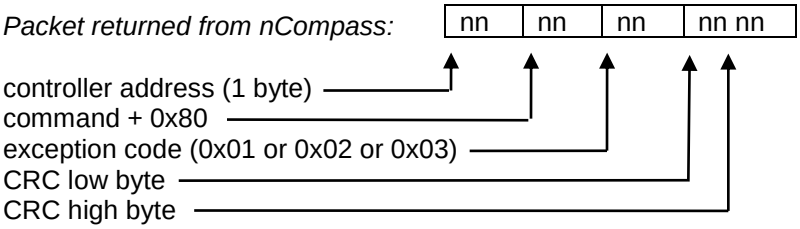
Packet returned from nCompass:

nn	10	nn nn	nn nn	nn nn

controller address (1 byte)
write command (0x10)
starting register high byte
starting register low byte
number of registers to write high byte
number of registers to write low byte
CRC low byte
CRC high byte

Exception Responses
When the NCompass cannot process a command, it returns an exception response and sets the high bit (0x80) of the command.

0x01 illegal command
0x02 illegal data address
0x03 illegal data value



4.3.2 Error Checking

In Modbus communications, every message sent from the master (your software) receives a response from the slave (nCompass), including write commands. Thus, after each command sent, you should read the controller response before sending the next message. This provides the method of error checking in order to verify that the message you sent was received correctly, and that the controller is operating accordingly. This allows you to then determine the appropriate recovery response in case the message was not received correctly by the controller, and what action is to be taken by an operator and/or the software itself.

The exception responses provide a basic form of error checking. When an exception response is received, the code provided in the response will tell you what the error was in the sent message. However, this is only valid if the controller receives the message you sent, and there was an out-of-range value or simple transmission error in the message. It does not validate incomplete or failed transmissions. To insure that the data you receive from a read command is correct, and that the controller properly received a write command, you must parse the controller's response and validate the return message to insure it is correct.

In order to validate that the message you received is correct, you must calculate the CRC for the received message and compare it with the CRC that the controller appended to the message. This verifies that the data you received was what nCompass sent. If the CRC's do not match, there was an error in the transmission and the entire message should be ignored. This could then be followed by an attempt to resend the failed command, or halt operation and alert an operator.

Example: Read registers 60 and 61 (loop 1 process variable and setpoint) of controller at address 1.

Command sent to nCompass 01 03 00 3C 00 02 04 07
Message received from nCompass: 01 03 04 03 0D 01 F3 2A 61

Calculated CRC: 2A61 (calculated from message 01 03 04 03 0D 01 F3)
Received CRC: 2A61

The calculated CRC matches the received CRC, the message is valid. Note that the last two bytes of the received message are not used to calculate the CRC. The last two bytes are the CRC that nCompass appended to the message. Do not include them when calculating the CRC.

4.3.3 Transmitting and Receiving Messages

In order to reliably communicate with the NCompass, it is important to develop an efficient means of transmitting and receiving messages. Modbus is a structured protocol and it must be properly followed. It is recommended, if possible, to locate an existing communication driver to incorporate into your software. Developing one from scratch can be challenging. However, if one is not available, or you choose to develop one yourself, the following guidelines may be of assistance.

Transmitting Messages

When sending a message to nCompass, it is important to remember that Modbus RTU protocol does not have start-of-transmission or end-of-transmission characters. All messages are "framed" using timeouts between characters. A timeout between characters is a pause of at least 1.5 characters in length, and a timeout between frames is a pause of at least 3.5 characters in length. If either of these periods are exceeded while a message is being sent to nCompass, it will discard the data it has received and wait for the first frame of the next valid communication.

At 9600 baud, the timeout between characters is a little over 1ms, and nCompass will take any characters after a delay of as little as 3ms, as the beginning of a new message. This is an important consideration, because in creating your message, there are several steps that must be executed in order to build the packet and format the data properly into hexadecimal to send out the serial port of your PC. If you write code in a manner that steps byte by byte through sending the message out the serial port, formatting each piece of data prior to sending it, there is a good possibility that two much time may pass between characters, thus causing a failed transmission.

Therefore, it is recommended that the entire message, including the CRC, be created and assembled prior to being sent to the serial port. By assembling the main body of the message first, you can then pass it to the CRC algorithm which can step sequentially through the message, generate the CRC and append it to the message body. Once the message is completely assembled, it can then be sent out the serial port as a completed packet. This will insure that the message reaches nCompass within the proper framing.

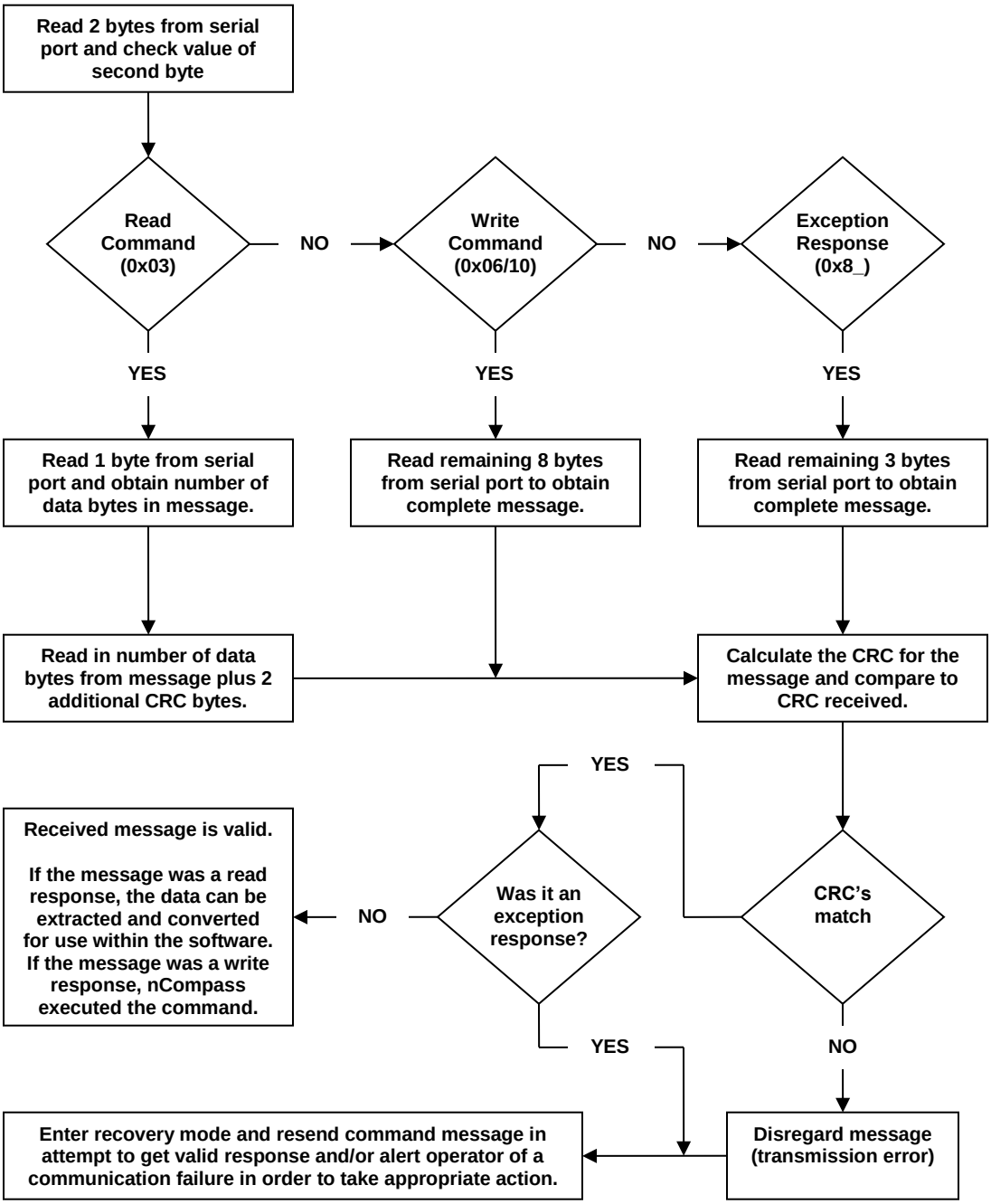
Receiving Messages

Due to the fact that Modbus RTU protocol does not have start-of-transmission or end-of-transmission characters, if the serial port driver you are using does not support an interval timeout setting allowing you to automatically terminate a read after a specified time passes between bytes (signaling the end of a message), you must know how long the message will be that you are receiving. That allows you to know how many bytes to read from your serial port and when you have received the entire message. If you rely on a maximum timeout period to terminate the read, depending upon the length of the received message, you will either loose a portion of the message or have to set the timeout period so high, that it will greatly affect the throughput of your code.

As can be seen from the previous examples for read and write commands in Section 4.3.1, the length of the returned message will vary based on the type of command, and for read commands, how many registers are being returned. Response messages can vary in length from as little as 5 bytes for an exception response to as many as 133 bytes for a read command. Therefore, in order to read in the message efficiently, you need to know what type of command it is in response to.

The response messages are always coded with the first two bytes of the message as the controller address and command type. When executing a read, read in only the first 2 bytes of data at the serial port. Examine the second byte and determine what the command is. If it is a write command (0x06 or 0x10), you know the response message is 8 bytes long. You can then read in the next 6 bytes of data from the serial port to complete the message. You can then calculate the CRC for the first 6 bytes of that message, and compare it to the last 2 bytes. If they match, then the communication completed successfully.

If the response is to a read command (0x03), you must then perform a single byte read from your serial port in order to get the next byte of the message. The third byte in a read response message is the number of data bytes in the message. By reading in this value, you then know how many data bytes follow. Note that this value does not include the 2 bytes for the CRC. Thus, when reading in the rest of the message, you will read in the number of data bytes plus an additional two, in order to get the CRC. You can then calculate the CRC for the message and compare it to the last two bytes. If they match, the data you received is valid.



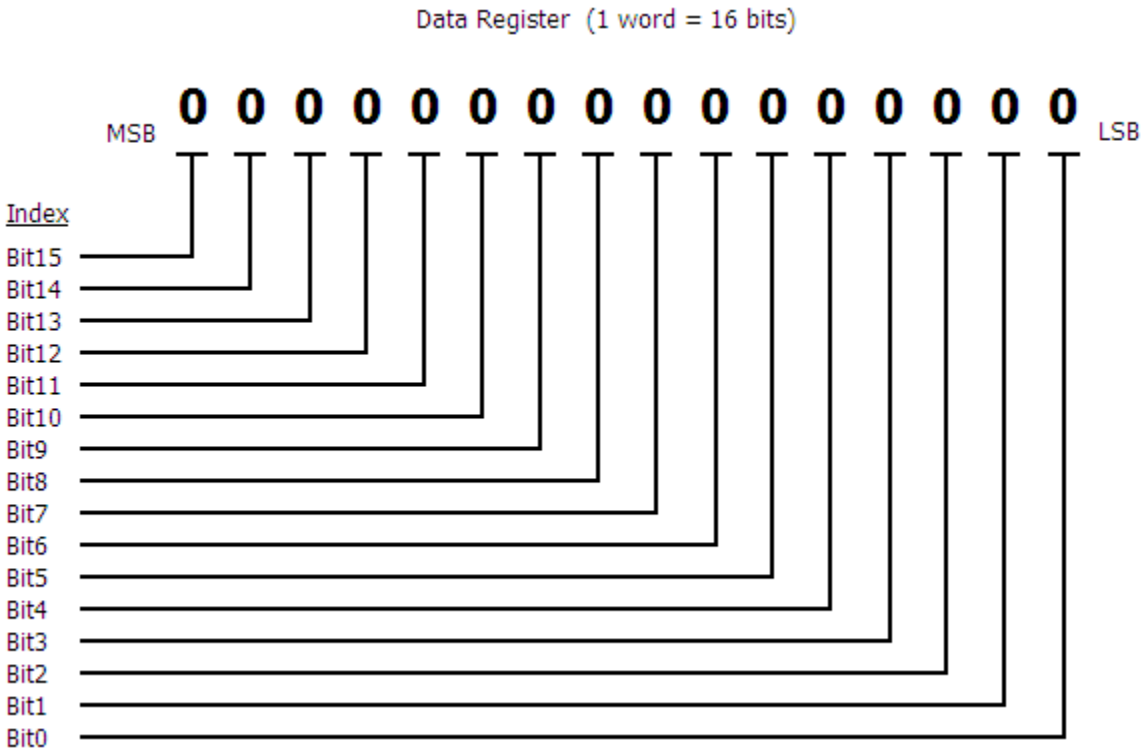
.

5 nCompass Data Registers

Some of the values contained in the nCompass register base contain bit oriented values. This means that each bit of the word indicates an on/off status for a specific setting or condition. In handling these values, it is recommended that the word be converted to its binary equivalent.

By converting the value to its binary equivalent, it produces a Boolean array of true [bit on (1)] and false [bit off (0)] values. This allows each bit to be examined individually. In the same manner, creating a Boolean array of 16 bits produces an equivalent hexadecimal value that can be sent to nCompass in order to set a control register.

For the purpose of this manual, parameters defined as bit oriented will have the function of each bit associated with the bit's index number in the data word. The index number is equal to that of a typical array function. Thus, an index number of zero, selects the first bit in the word (LSB). An index number of 1 selects the second bit in the word, and so on. This helps eliminate offset selection errors that may occur when coding software and using array functions to select which bit in the word that is required for examination.



Adhere to the following lists of registers and their allowable data ranges. DO NOT attempt to write to any other register number than those listed. DO NOT write to registers that are for options your controller does not have. Failure to adhere to this requirement can result in erratic control and/or damage to equipment.

5.1 Control Registers

Register #		Modbus Address	Parameter Description	Data *A Type	Range *B Low	Range *B High	*C Unit
0	**(0x0000)**	400001	System Mode Control	R/W	*B1	*B1	-
1	**(0x0001)**	400002	**RESERVED** – DO NOT WRITE				
2	**(0x0002)**	400003	Power Out Recovery Mode	R/W	*B2	*B2	-
3	**(0x0003)**	400004	Recovery Power Out Time	R/W	0	32767	seconds
4	**(0x0004)**	400005	Demand Defrost/Status	R/W	*B3	*B3	-
5	**(0x0005)**	400006	Defrost Interval	R/W	0	999	hours
6	**(0x0006)**	400007	Defrost Duration	R/W	0	999	minutes
7	**(0x0007)**	400008	Fan Delay (PreCool)	R/W	0	999	seconds
8	**(0x0008)**	400009					
9	**(0x0009)**	400010	Control Loop Auto/Manual Control	R/W	*B4	*B4	-
10	**(0x000A)**	400011	Control Loop Autotune Activation	R/W	*B5	*B5	-
11	**(0x000B)**	400012	**RESERVED** – DO NOT WRITE				
12	**(0x000C)**	400013	System Events 1-16	R/W	*B6	*B6	-
13	**(0x000D)**	400014	System Events 16-32	R/W	*B7	*B7	-
14	**(0x000E)**	400015	Program Start Step Number	W	1	99	-
15	**(0x000F)**	400016	Program Operating Status	R/W	*B8	*B8	-
16	**(0x0010)**	400017	Program Advance Previous/Next Step	W	*B9	*B9	-
17	**(0x0011)**	400018	Program Step Time Addition	W	0	32767	minutes
18	**(0x0012)**	400019	Program Name Characters 1 & 2	R	*B10	*B10	-
19	**(0x0013)**	400020	Program Name Characters 3 & 4	R	*B10	*B10	-
20	**(0x0014)**	400021	Program Name Characters 5 & 6	R	*B10	*B10	-
21	**(0x0015)**	400022	Program Name Characters 7 & 8	R	*B10	*B10	-
22	**(0x0016)**	400023	Program Name Characters 9 & 10	R	*B10	*B10	-
23	**(0x0017)**	400024	Year/Month Program Started	R	*B11	*B11	-
24	**(0x0018)**	400025	Day/DOW Program Started	R	*B12	*B12	-
25	**(0x0019)**	400026	Hour/Minute Program Started	R	*B13	*B13	-
26	**(0x001A)**	400027	Year/Month Estimated Program End	R	*B11	*B11	-
27	**(0x001B)**	400028	Day/DOW Estimated Program End	R	*B12	*B12	-
28	**(0x001C)**	400029	Hour/Minute Estimated Program End	R	*B13	*B13	-
29	**(0x001D)**	400030	Current Step of Program	R	1	99	-
30	**(0x001E)**	400031	Hours Left in Current Step	R	0	999	hours
31	**(0x001F)**	400032	Minutes/Seconds Left in Current Step	R	*B14	*B14	-
32	**(0x0020)**	400033	Program Wait Status	R	*B15	*B15	-
33	**(0x0021)**	400034	Waiting For Input (loop/monitor/digital)	R	*B16	*B16	-
34	**(0x0022)**	400035	Wait Setpoint	R	-3276.8	3276.7	*C1
35	**(0x0023)**	400036	Current Step Jump Step Number	R	1	99	-
36	**(0x0024)**	400037	Current Step Jumps Remaining	R	0	999	-

Register #	Modbus Address	Parameter Description	Data *A Type	Range *B		*C Unit
				Low	High	
37 **(0x0025)**	400038	Program Loop 1 Target Setpoint	R	-32768	32767	*C2
38 **(0x0026)**	400039	Program Loop 2 Target Setpoint	R	-32768	32767	*C2
39 **(0x0027)**	400040	Program Loop 3 Target Setpoint	R	-32768	32767	*C2
40 **(0x0028)**	400041	Program Loop 4 Target Setpoint	R	-32768	32767	*C2
41 **(0x0029)**	400042	Program Loop 5 Target Setpoint	R	-32768	32767	*C2
42 **(0x002A)**	400043	Program Loop 6 Target Setpoint	R	-32768	32767	*C2
43 **(0x002B)**	400044	Program Loop 7 Target Setpoint	R	-32768	32767	*C2
44 **(0x002C)**	400045	Program Loop 8 Target Setpoint	R	-32768	32767	*C2
45 **(0x002D)**	400046	Program Loop 9 Target Setpoint	R	-32768	32767	*C2
46 **(0x002E)**	400047	Program Loop 10 Target Setpoint	R	-32768	32767	*C2
47 **(0x002F)**	400048					
48 **(0x0030)**	400049					
49 **(0x0031)**	400050					
50 **(0x0032)**	400051					
51 **(0x0033)**	400052					
52 **(0x0034)**	400053	Last Program Jump Made from Step	R	1	99	-
53 **(0x0035)**	400054	Last Program Jump Made to Step	R	1	99	-
54 **(0x0036)**	400055	Total Program Jumps Made	R	0	32767	-
55 **(0x0037)**	400056	Loops Under Program Control	R	*B17	*B17	-
56 **(0x0038)**	400057					
57 **(0x0039)**	400058					
58 **(0x003A)**	400059					
59 **(0x003B)**	400060					
60 **(0x003C)**	400061	Loop 1 Process Variable (PV)	R	-32768	32767	*C2
61 **(0x003D)**	400062	Loop 1 Setpoint (SP)	R/W	-32768	32767	*C2
62 **(0x003E)**	400063	Loop 1 Percent Output (%Out)	R/W	-100.00	100.00	%
63 **(0x003F)**	400064	Loop 2 Process Variable (PV)	R	-32768	32767	*C2
64 **(0x0040)**	400065	Loop 2 Setpoint (SP)	R/W	-32768	32767	*C2
65 **(0x0041)**	400066	Loop 2 Percent Output (%Out)	R/W	-100.00	100.00	%
66 **(0x0042)**	400067	Loop 3 Process Variable (PV)	R	-32768	32767	*C2
67 **(0x0043)**	400068	Loop 3 Setpoint (SP)	R/W	-32768	32767	*C2
68 **(0x0044)**	400069	Loop 3 Percent Output (%Out)	R/W	-100.00	100.00	%
69 **(0x0045)**	400070	Loop 4 Process Variable (PV)	R	-32768	32767	*C2
70 **(0x0046)**	400071	Loop 4 Setpoint (SP)	R/W	-32768	32767	*C2
71 **(0x0047)**	400072	Loop 4 Percent Output (%Out)	R/W	-100.00	100.00	%
72 **(0x0048)**	400073	Loop 5 Process Variable (PV)	R	-32768	32767	*C2
73 **(0x0049)**	400074	Loop 5 Setpoint (SP)	R/W	-32768	32767	*C2
74 **(0x004A)**	400075	Loop 5 Percent Output (%Out)	R/W	-100.00	100.00	%

Register #	Modbus Address	Parameter Description	Data *A Type	Range *B Low	Range *B High	*C Unit
75 **(0x004B)**	400076	Loop 6 Process Variable (PV)	R	-32768	32767	*C2
76 **(0x004C)**	400077	Loop 6 Setpoint (SP)	R/W	-32768	32767	*C2
77 **(0x004D)**	400078	Loop 6 Percent Output (%Out)	R/W	-100.00	100.00	%
78 **(0x004E)**	400079	Loop 7 Process Variable (PV)	R	-32768	32767	*C2
79 **(0x004F)**	400080	Loop 7 Setpoint (SP)	R/W	-32768	32767	*C2
80 **(0x0050)**	400081	Loop 7 Percent Output (%Out)	R/W	-100.00	100.00	%
81 **(0x0051)**	400082	Loop 8 Process Variable (PV)	R	-32768	32767	*C2
82 **(0x0052)**	400083	Loop 8 Setpoint (SP)	R/W	-32768	32767	*C2
83 **(0x0053)**	400084	Loop 8 Percent Output (%Out)	R/W	-100.00	100.00	%
84 **(0x0054)**	400085	Loop 9 Process Variable (PV)	R	-32768	32767	*C2
85 **(0x0055)**	400086	Loop 9 Setpoint (SP)	R/W	-32768	32767	*C2
86 **(0x0056)**	400087	Loop 9 Percent Output (%Out)	R/W	-100.00	100.00	%
87 **(0x0057)**	400088	Loop 10 Process Variable (PV)	R	-32768	32767	*C2
88 **(0x0058)**	400089	Loop 10 Setpoint (SP)	R/W	-32768	32767	*C2
89 **(0x0059)**	400090	Loop 10 Percent Output (%Out)	R/W	-100.00	100.00	%
90 **(0x005A)**	400091					
91 **(0x005B)**	400092					
92 **(0x005C)**	400093					
93 **(0x005D)**	400094					
94 **(0x005E)**	400095					
95 **(0x005F)**	400096					
96 **(0x0060)**	400097					
97 **(0x0061)**	400098					
98 **(0x0062)**	400099					
99 **(0x0063)**	400100					
100 **(0x0064)**	400101					
101 **(0x0065)**	400102					
102 **(0x0066)**	400103					
103 **(0x0067)**	400104					
104 **(0x0068)**	400105					
105 **(0x0069)**	400106	Monitor 1 Process Variable	R	-32768	32767	*C2
106 **(0x006A)**	400107	Monitor 2 Process Variable	R	-32768	32767	*C2
107 **(0x006B)**	400108	Monitor 3 Process Variable	R	-32768	32767	*C2
108 **(0x006C)**	400109	Monitor 4 Process Variable	R	-32768	32767	*C2
109 **(0x006D)**	400110	Monitor 5 Process Variable	R	-32768	32767	*C2
110 **(0x006E)**	400111	Monitor 6 Process Variable	R	-32768	32767	*C2
111 **(0x006F)**	400112	Monitor 7 Process Variable	R	-32768	32767	*C2
112 **(0x0070)**	400113	Monitor 8 Process Variable	R	-32768	32767	*C2

Register #	Modbus Address	Parameter Description	Data *A Type	Range *B Low	Range *B High	*C Unit
113 **(0x0071)**	400114	Monitor 9 Process Variable	R	-32768	32767	*C2
114 **(0x0072)**	400115	Monitor 10 Process Variable	R	-32768	32767	*C2
115 **(0x0073)**	400116	Monitor 11 Process Variable	R	-32768	32767	*C2
116 **(0x0074)**	400117	Monitor 12 Process Variable	R	-32768	32767	*C2
117 **(0x0075)**	400118	Monitor 13 Process Variable	R	-32768	32767	*C2
118 **(0x0076)**	400119	Monitor 14 Process Variable	R	-32768	32767	*C2
119 **(0x0077)**	400120	Monitor 15 Process Variable	R	-32768	32767	*C2
120 **(0x0078)**	400121	**RESERVED** – DO NOT WRITE				
121 **(0x0079)**	400122	**RESERVED** – DO NOT WRITE				
122 **(0x007A)**	400123	**RESERVED** – DO NOT WRITE				
123 **(0x007B)**	400124	**RESERVED** – DO NOT WRITE				
124 **(0x007C)**	400125	**RESERVED** – DO NOT WRITE				
125 **(0x007D)**	400126	**RESERVED** – DO NOT WRITE				
126 **(0x007E)**	400127	**RESERVED** – DO NOT WRITE				
127 **(0x007F)**	400128	**RESERVED** – DO NOT WRITE				
128 **(0x0080)**	400129	**RESERVED** – DO NOT WRITE				
129 **(0x0081)**	400130	**RESERVED** – DO NOT WRITE				
130 **(0x0082)**	400131	**RESERVED** – DO NOT WRITE				
131 **(0x0083)**	400132	**RESERVED** – DO NOT WRITE				
132 **(0x0084)**	400133	**RESERVED** – DO NOT WRITE				
133 **(0x0085)**	400134	**RESERVED** – DO NOT WRITE				
134 **(0x0086)**	400135					
135 **(0x0087)**	400136					
136 **(0x0088)**	400137	Alarm Acknowledge	W	0	1	-
137 **(0x0089)**	400138	Loop Communication Fault Alarms	R	*B18	*B18	-
138 **(0x008A)**	400139	**RESERVED** – DO NOT WRITE				
139 **(0x008B)**	400140	Loop Input Alarms	R	*B19	*B19	-
140 **(0x008C)**	400141	Monitor Input Alarms	R	*B20	*B20	-
141 **(0x008D)**	400142	Loop/Monitor Alarms	R	*B21	*B21	-
142 **(0x008E)**	400143	Loop/Monitor Alarms (w/Service Alert)	R	*B22	*B22	-
143 **(0x008F)**	400144	Digital Input Alarms	R	*B23	*B23	-
144 **(0x0090)**	400145					
145 **(0x0091)**	400146					
146 **(0x0092)**	400147	**RESERVED** – DO NOT WRITE				
147 **(0x0093)**	400148	**RESERVED** – DO NOT WRITE				
148 **(0x0094)**	400149	**RESERVED** – DO NOT WRITE				
149 **(0x0095)**	400150	**RESERVED** – DO NOT WRITE				
150 **(0x0096)**	400151	**RESERVED** – DO NOT WRITE				

Register #		Modbus Address	Parameter Description	Data *A	Range *B		*C
				Type	Low	High	Unit
151	**(0x0097)**	400152	**RESERVED** – DO NOT WRITE				
152	**(0x0098)**	400153	**RESERVED** – DO NOT WRITE				
153	**(0x0099)**	400154	**RESERVED** – DO NOT WRITE				
154	**(0x009A)**	400155					
155	**(0x009B)**	400156					
156	**(0x009C)**	400157					
157	**(0x009D)**	400158					
158	**(0x009E)**	400159					
159	**(0x009F)**	400160	Redundancy Primary System/Status	R/W	*B24	*B24	-
160	**(0x00A0)**	400161	Alternating Run Time	R/W	0	32767	minutes
161	**(0x00A1)**	400162	Alternating Time of Day (HH)	R/W	0	23	hours
162	**(0x00A2)**	400163	Alternating Time of Day (MM)	R/W	0	59	minutes
163	**(0x00A3)**	400164	Product Load Timer Demand/Status	R/W	*B25	*B25	-
164	**(0x00A4)**	400165	Concurrent Minimum Run Time	R/W	0	32767	minutes
165	**(0x00A5)**	400166	**RESERVED** – DO NOT WRITE				
166	**(0x00A6)**	400167	**RESERVED** – DO NOT WRITE				
167	**(0x00A7)**	400168	**RESERVED** – DO NOT WRITE				
168	**(0x00A8)**	400169	**RESERVED** – DO NOT WRITE				
169	**(0x00A9)**	400170	**RESERVED** – DO NOT WRITE				
170	**(0x00AA)**	400171	**RESERVED** – DO NOT WRITE				
171	**(0x00AB)**	400172	**RESERVED** – DO NOT WRITE				
172	**(0x00AC)**	400173	**RESERVED** – DO NOT WRITE				
173	**(0x00AD)**	400174	**RESERVED** – DO NOT WRITE				
174	**(0x00AE)**	400175					
175	**(0x00AF)**	400176					
176	**(0x00B0)**	400177	Control Module Input Status	R	*B26	*B26	-
177	**(0x00B1)**	400178	Auxiliary Input Status	R	*B27	*B27	-
178	**(0x00B2)**	400179	Control Module/Auxiliary Output Status	R	*B28	*B28	-
179	**(0x00B3)**	400180	Auxiliary Output Status	R	*B29	*B29	-

Notes:

*A R/W Specifies readable / writable data, R specifies read only data and W specifies a write only control value.

*B The range of certain parameters are dependent upon system options. Consult the following range tables for information regarding the use of these parameters.

Reading bit oriented parameters

The value contained in these parameters is dependent upon the combination of "on" bits (1). Therefore, only the individual status of each bit has meaning, not the value of the parameter.

Setting bit oriented parameters

The value that must be written to these parameters is dependent upon the combination of "on" bits. Therefore, it is necessary to know the current value of the parameter before setting it so that only the bit status you want to update is changed. Otherwise, sending a value derived from only the bit you wish to set, will turn off all other functions related to the other bits in the parameter.

*B1

Parameter Value	Description
Bit0	nCompass Online
Bit1 - Bit15	Not Assigned

DO NOT alter the state of this register. Bit0 is the system online bit and is set by the nCompass HMI when the unit is ready for operation. Turning off this bit will turn off the system.

The status of this register should be used for information only, as a means of determining if the system is ready for operation.

*B2

Parameter Value	Description
Bit0	Off
Bit1	Hold
Bit2	Continue
Bit3	Restart
Bit4	Resume
Bit5 – Bit15	Not Assigned

*B3

Parameter Value	Description
Bit0	Demand Defrost
Bit1-7	Not Assigned
Bit8	Defrost Off
Bit9	Defrost Cycle Active
Bit10	Fan Delay (PreCool) Active
Bit11-15	Not Assigned

Note: Bit0 (demand defrost) will automatically reset after being written to activate defrost. Bits 8-10 are status bits only and will indicate the current defrost status.

*B4

Parameter Value	Description
Bit0	Loop 1 in Manual
Bit1	Loop 2 in Manual
Bit2	Loop 3 in Manual
Bit3	Loop 4 in Manual
Bit4	Loop 5 in Manual
Bit5	Loop 6 in Manual
Bit6	Loop 7 in Manual
Bit7	Loop 8 in Manual
Bit8	Loop 9 in Manual
Bit9	Loop 10 in Manual
Bit10-15	Not Assigned

Note: Manual operation may not be available on certain loop controls depending upon their configuration. If manual operation is not available, the bit for the loop will automatically turn off after being set.

When in manual mode, writing to the loop %Out register will adjust the output percentage of the control loop.

*B5

Parameter Value	Description
Bit0	Loop 1 in Autotune
Bit1	Loop 2 in Autotune
Bit2	Loop 3 in Autotune
Bit3	Loop 4 in Autotune
Bit4	Loop 5 in Autotune
Bit5	Loop 6 in Autotune
Bit6	Loop 7 in Autotune
Bit7	Loop 8 in Autotune
Bit8	Loop 9 in Autotune
Bit9	Loop 10 in Autotune
Bit10-15	Not Assigned

Note: Autotune operation is not supported on all loop controls compatible with the nCompass or may be unavailable depending upon their configuration. If autotune operation is not available, the bit for the loop will automatically turn off after being set.

When autotune completes normally, the bit for the loop will automatically turn off indicating that tune is complete. To terminate an autotune in progress, turn off the bit for the desired loop.

*B6

Parameter Value	Description
Bit0	System Event 1
Bit1	System Event 2
Bit2	System Event 3
Bit3	System Event 4
Bit4	System Event 5
Bit5	System Event 6
Bit6	System Event 7
Bit7	System Event 8
Bit8	System Event 9
Bit9	System Event 10
Bit10	System Event 11
Bit11	System Event 12
Bit12	System Event 13
Bit13	System Event 14
Bit14	System Event 15
Bit15	System Event 16

Note: Not all system events may be available on your system. Event names and functions are defined by system configuration. Consult your system documentation or contact your OEM for information on event use.

*B7

Parameter Value	Description
Bit0	System Event 17
Bit1	System Event 18
Bit2	System Event 19
Bit3	System Event 20
Bit4	System Event 21
Bit5	System Event 22
Bit6	System Event 23
Bit7	System Event 24
Bit8	System Event 25
Bit9	System Event 26
Bit10	System Event 27
Bit11	System Event 28
Bit12	System Event 29
Bit13	System Event 30
Bit14	System Event 31
Bit15	System Event 32

Note: Not all system events may be available on your system. Event names and functions are defined by system configuration. Consult your system documentation or contact your OEM for information on event use.

*B8

Parameter Value	Description
0	Program Not Running
1	Stop Program
2	Stop Program (All Off)
4	Hold Program
8	Run/Resume Program
16	Program in Autostart**
32	Program in Wait **
64	Program in Ramp**
128	Program in Soak**
256	Program in Guaranteed Soak**

**These values are set by the nCompass to indicate the operating status of the profile and cannot be set directly.

*B9

Parameter Value	Description
1	Program Advance to Previous Step
2	Program Advance to Next Step

This parameter only performs its function when the profile is in hold. Once the set function is executed, this parameter automatically resets to zero (0).

*B10

Parameter Value	High Order Byte	Low Order Byte	Description
Range Low	32	32	Program Name Character (ASCII Table)
Range High	126	126	Program Name Character (ASCII Table)

See the ASCII character chart in Section 3.1 for the character representation of these values.

Example

Read command of registers 18 to 22 from the nCompass returns the following values:

Register Values:	0x74 53	0x72 6F	0x20 65	0x65 54	0x74 73
ASCII Equivalent:	t S	r o	e	e T	t s

Assemble the ASCII characters in order from low to high byte starting with register 18 in order to assemble the Program name: “Store Test”. Note that null characters are not used in the Program name. A space (0x20) will be used is used in place of a null character to maintain the 10 character name length if the Program name is not ten characters long.

*B11

Parameter Value	Range Low	Range High	Description
High Byte	0	99	Year
Low Byte	1	12	Month

*B12

Parameter Value	Range Low	Range High	Description
High Byte	1	31	Day
Low Byte	0	6	Day of Week**

**The days of the week are represented as numbers:
0=Sun, 1=Mon, 2=Tue, 3=Wed, 4=Thu, 5=Fri, 6=Sat

*B13

Parameter Value	Range Low	Range High	Description
High Byte	0	23	Hour
Low Byte	0	59	Minute

Example

Read command of registers 23 to 25 for program start time or 26 to 28 for estimated program stop time from the nCompass returns the following values:

Register Values:	0x0A 0B	0x04 04	0x0A 1D
Decimal Equivalent:	10 11	4 4	10 29

Translating the values into an actual date and time provides a date and time of Thursday November 4, 2010 at 10:29am.

*B14

Parameter Value	Range Low	Range High	Description
High Byte	0	59	Minutes
Low Byte	0	59	Seconds

*B15

Parameter Value	Description
Bit0	Not Waiting
Bit1	Wait For Loop
Bit2	Wait For Monitor
Bit3	Wait For Digital Input
Bit4 - Bit15	Not Assigned

Note: Multiple wait for conditions can be active at once, i.e., the profile could be waiting for a combination of loops, monitors and/or digital inputs at the same time.

*B16

Parameter Value	Description
Bit0	Loop/Monitor/Digital Input 1
Bit1	Loop/Monitor/Digital Input 2
Bit2	Loop/Monitor/Digital Input 3
Bit3	Loop/Monitor/Digital Input 4
Bit4	Loop/Monitor/Digital Input 5
Bit5	Loop/Monitor/Digital Input 6
Bit6	Loop/Monitor/Digital Input 7
Bit7	Loop/Monitor/Digital Input 8
Bit8	Loop/Monitor/Digital Input 9
Bit9	Loop/Monitor/Digital Input 10
Bit10	Monitor/Digital Input 11
Bit11	Monitor/Digital Input 12
Bit12	Monitor/Digital Input 13
Bit13	Monitor/Digital Input 14
Bit14	Monitor/Digital Input 15
Bit15	Digital Input 16

Note: Each bit in the word can represent a wait for condition for more than one input, i.e., Bit0 can be on to indicate it is waiting for loop 1, monitor input 1 or digital input 1.

*B17

Parameter Value	Description
Bit0	Loop 1 Under Program Control
Bit1	Loop 2 Under Program Control
Bit2	Loop 3 Under Program Control
Bit3	Loop 4 Under Program Control
Bit4	Loop 5 Under Program Control
Bit5	Loop 6 Under Program Control
Bit6	Loop 7 Under Program Control
Bit7	Loop 8 Under Program Control
Bit8	Loop 9 Under Program Control
Bit9	Loop 10 Under Program Control
Bit10-15	Digital Input 16

*B18

Parameter Value	Description
Bit0	Loop 1 Communications Fault
Bit1	Loop 2 Communications Fault
Bit2	Loop 3 Communications Fault
Bit3	Loop 4 Communications Fault
Bit4	Loop 5 Communications Fault
Bit5	Loop 6 Communications Fault
Bit6	Loop 7 Communications Fault
Bit7	Loop 8 Communications Fault
Bit8	Loop 9 Communications Fault
Bit9	Loop 10 Communications Fault
Bit10-14	Not Assigned
Bit15	Monitor Communications Fault

*B19

Parameter Value	Description
Bit0	Loop 1 Sensor Break
Bit1	Loop 2 Sensor Break
Bit2	Loop 3 Sensor Break
Bit3	Loop 4 Sensor Break
Bit4	Loop 5 Sensor Break
Bit5	Loop 6 Sensor Break
Bit6	Loop 7 Sensor Break
Bit7	Loop 8 Sensor Break
Bit8	Loop 9 Sensor Break
Bit9	Loop 10 Sensor Break
Bit10-15	Not Assigned

*B20

Parameter Value	Description
Bit0	Monitor 1 Sensor Break
Bit1	Monitor 2 Sensor Break
Bit2	Monitor 3 Sensor Break
Bit3	Monitor 4 Sensor Break
Bit4	Monitor 5 Sensor Break
Bit5	Monitor 6 Sensor Break
Bit6	Monitor 7 Sensor Break
Bit7	Monitor 8 Sensor Break
Bit8	Monitor 9 Sensor Break
Bit9	Monitor 10 Sensor Break
Bit10	Monitor 11 Sensor Break
Bit11	Monitor 12 Sensor Break
Bit12	Monitor 13 Sensor Break
Bit13	Monitor 14 Sensor Break
Bit14	Monitor 15 Sensor Break
Bit15	Not Assigned

*B21

Parameter Value	Description
Bit0	Loop/Monitor Alarm 1
Bit1	Loop/Monitor Alarm 2
Bit2	Loop/Monitor Alarm 3
Bit3	Loop/Monitor Alarm 4
Bit4	Loop/Monitor Alarm 5
Bit5	Loop/Monitor Alarm 6
Bit6	Loop/Monitor Alarm 7
Bit7	Loop/Monitor Alarm 8
Bit8	Loop/Monitor Alarm 9
Bit9	Loop/Monitor Alarm 10
Bit10	Loop/Monitor Alarm 11
Bit11	Loop/Monitor Alarm 12
Bit12	Loop/Monitor Alarm 13
Bit13	Loop/Monitor Alarm 14
Bit14	Loop/Monitor Alarm 15
Bit15	Loop/Monitor Alarm 16

*B22

Parameter Value	Description
Bit0	Loop/Monitor Alarm 17
Bit1	Loop/Monitor Alarm 18
Bit2	Loop/Monitor Alarm 19
Bit3	Loop/Monitor Alarm 20
Bit4	Loop/Monitor Alarm 21
Bit5	Loop/Monitor Alarm 22
Bit6	Loop/Monitor Alarm 23
Bit7	Loop/Monitor Alarm 24
Bit8	Loop/Monitor Alarm 25
Bit9	Loop/Monitor Alarm 26
Bit10	Loop/Monitor Alarm 27
Bit11	Loop/Monitor Alarm 28
Bit12	Loop/Monitor Alarm 29
Bit13	Loop/Monitor Alarm 30
Bit14	Service Alert
Bit15	RESERVED

*B23

Parameter Value	Description
Bit0	Digital Input 0 Alarm
Bit1	Digital Input 1 Alarm
Bit2	Digital Input 2 Alarm
Bit3	Digital Input 3 Alarm
Bit4	Digital Input 4 Alarm
Bit5	Digital Input 5 Alarm
Bit6	Digital Input 6 Alarm
Bit7	Digital Input 7 Alarm
Bit8	Digital Input 8 Alarm
Bit9	Digital Input 9 Alarm
Bit10	Digital Input 10 Alarm
Bit11	Digital Input 11 Alarm
Bit12	Digital Input 12 Alarm
Bit13	Digital Input 13 Alarm
Bit14	Digital Input 14 Alarm
Bit15	Digital Input 15 Alarm

*B24

Parameter Value	Description
Bit0	Redundancy in Manual
Bit1	Redundancy in Auto
Bit2	Redundancy Reset
Bit3	Not Assigned
Bit4	System A Primary Selection
Bit5	System B Primary Selection
Bit6-7	Not Assigned
Bit8	Running System A in Auto
Bit9	Running System A in Manual
Bit10	Running System B in Auto
Bit11	Running System B in Manual
Bit12	Running System A in Fail Mode
Bit13	Running System B in Fail Mode
Bit14	Running System A/B Concurrent
Bit15	Fail Mode System A and B

Note: When setting manual mode (Bit0=on, Bit1=off), be sure to also set the current primary system (Bit4 or Bit5). To do this, read the register to get the current status first, set the lower two bits of the word to the desired mode, and then write the result back to the nCompass. If neither bit is set for system A or system B, the nCompass will default system A as the primary.

Bits8-15 are status only bits and will indicate the current operating condition. If none of the bits are on, redundancy is inactive and neither system A nor system B outputs are on.

*B25

Parameter Value	Description
Bit0	Demand Inhibit
Bit1-7	Not Assigned
Bit8	Product Load Timer Off
Bit9	Product Load Timer On
Bit10-15	Not Assigned

Note: Bit0 (demand inhibit) will automatically reset after being written to activate the product load timer. Bit8 and Bit9 are status bits only and will indicate the product load timer status.

*B26

Parameter Value	Description
Bit0	Digital Input 0 On
Bit1	Digital Input 1 On
Bit2	Digital Input 2 On
Bit3	Digital Input 3 On
Bit4	Digital Input 4 On
Bit5	Digital Input 5 On
Bit6	Digital Input 6 On
Bit7	Digital Input 7 On
Bit8 – Bit15	Not Assigned

*B27

Parameter Value	Description
Bit0	Digital Input 8 On
Bit1	Digital Input 9 On
Bit2	Digital Input 10 On
Bit3	Digital Input 11 On
Bit4	Digital Input 12 On
Bit5	Digital Input 13 On
Bit6	Digital Input 14 On
Bit7	Digital Input 15 On
Bit8 – Bit15	Not Assigned

*B28

Parameter Value	Description
Bit0	Digital Output 0 On
Bit1	Digital Output 1 On
Bit2	Digital Output 2 On
Bit3	Digital Output 3 On
Bit4	Digital Output 4 On
Bit5	Digital Output 5 On
Bit6	Digital Output 6 On
Bit7	Digital Output 7 On
Bit8	Digital Output 8 On
Bit9	Digital Output 9 On
Bit10	Digital Output 10 On
Bit11	Digital Output 11 On
Bit12	Digital Output 12 On
Bit13	Digital Output 13 On
Bit14	Digital Output 14 On
Bit15	Digital Output 15 On

*B29

Parameter Value	Description
Bit0	Digital Output 16 On
Bit1	Digital Output 17 On
Bit2	Digital Output 18 On
Bit3	Digital Output 19 On
Bit4	Digital Output 20 On
Bit5	Digital Output 21 On
Bit6	Digital Output 22 On
Bit7	Digital Output 23 On
Bit8	Digital Output 24 On
Bit9	Digital Output 25 On
Bit10	Digital Output 26 On
Bit11	Digital Output 27 On
Bit12	Digital Output 28 On
Bit13	Digital Output 29 On
Bit14	Digital Output 30 On
Bit15	Digital Output 31 On

*C1 The 'wait setpoint' does not have units of measure. It is a raw numerical value, i.e., it is compared directly to the numerical value of any loop or monitor input selected as a 'wait for' input. The wait for setpoint uses an implied decimal point of 1 for all comparisons regardless of the decimal point configuration of the loop or monitor input.

*C2 The units of measure and range of a loop or monitor input is dependent upon the configuration of the input and/or the units of temperature selection (Celsius or Fahrenheit) of the nCompass. The decimal point position for the loop or monitor input is an implied value based on the configuration of the input. Thus, a register value of 345 can represent an actual process value of 345, 34.5, 3.45 or 0.345 depending upon the decimal point configuration of the loop or monitor input.

5.2 Automatic Ramp/Soak Program Registers

The program parameters are a separate group of registers that are used to download programs to nCompass. The manner in which the program steps are configured and sent to the nCompass is specific and must be followed exactly.

Each program step consists of 28 data registers. Programs must be written one step at a time, using a multiple write command (0x10) to write the data for all 28 registers at once. This allows programs to be stored as two-dimensional arrays, of which code can be written to simply index through the array step-by-step, and transmit the program to nCompass.

The first 28 registers of the Program contain specific settings related to the program. These include autostart settings, the program name, the length of the program (number of steps), and guaranteed soak band settings.

Register #	Modbus Address	Parameter Description	Data *D	Range *E		*F
			Type	Low	High	Unit
2000 **(0x07D0)**	402001	Autostart On/Off	W	*E1	*E1	-
2001 **(0x07D1)**	402002	Year/Month for Autostart	W	*E2	*E2	-
2002 **(0x07D2)**	402003	Day/DOW for Autotstart	W	*E3	*E3	-
2003 **(0x07D3)**	402004	Time of Day for Autostart	W	*E4	*E4	-
2004 **(0x07D4)**	402005	Program Name (Chars 1 & 2)	W	*E5	*E5	-
2005 **(0x07D5)**	402006	Program Name (Chars 3 & 4)	W	*E5	*E5	-
2006 **(0x07D6)**	402007	Program Name (Chars 5 & 6)	W	*E5	*E5	-
2007 **(0x07D7)**	402008	Program Name (Chars 7 & 8)	W	*E5	*E5	-
2008 **(0x07D8)**	402009	Program Name (Chars 9 & 10)	W	*E5	*E5	-
2009 **(0x07D9)**	402010	Total Number of Steps in Program	W	1	99	-
2010 **(0x07DA)**	402011	Not Assigned	W	-	-	-
2011 **(0x07DB)**	402012	Guaranteed Soak Band Loop 1	W	0	32767	PV
2012 **(0x07DC)**	402013	Guaranteed Soak Band Loop 2	W	0	32767	PV
2013 **(0x07DD)**	402014	Guaranteed Soak Band Loop 3	W	0	32767	PV
2014 **(0x07DE)**	402015	Guaranteed Soak Band Loop 4	W	0	32767	PV
2015 **(0x07DF)**	402016	Guaranteed Soak Band Loop 5	W	0	32767	PV
2016 **(0x07E0)**	402017	Guaranteed Soak Band Loop 6	W	0	32767	PV
2017 **(0x07E1)**	402018	Guaranteed Soak Band Loop 7	W	0	32767	PV
2018 **(0x07E2)**	402019	Guaranteed Soak Band Loop 8	W	0	32767	PV
2019 **(0x07E3)**	402020	Guaranteed Soak Band Loop 9	W	0	32767	PV
2020 **(0x07E4)**	402021	Guaranteed Soak Band Loop 10	W	0	32767	PV
2021 **(0x07E5)**	402022	Not Assigned	W	-	-	-
2022 **(0x07E6)**	402023	Not Assigned	W	-	-	-
2023 **(0x07E7)**	402024	Not Assigned	W	-	-	-
2024 **(0x07E8)**	402025	Not Assigned	W	-	-	-
2025 **(0x07E9)**	402026	Not Assigned	W	-	-	-
2026 **(0x07EA)**	402027	Not Assigned	W	-	-	-
2027 **(0x07EB)**	402028	Not Assigned	W	-	-	-

The following 28 registers of the Program contain the data for step 1 of the Program.

Register #	Modbus Address	Parameter Description	Data *D Type	Range *E		*F Unit
				Low	High	
2028 **(0x07EC)**	402029	Step Time Hours	W	0	9999	-
2029 **(0x07ED)**	402030	Step Time Minutes/Seconds	W	*E6	*E6	-
2030 **(0x07EE)**	402031	System Events 1-16	W	*E7	*E7	-
2031 **(0x07EF)**	402032	System Events 17-32	W	*E8	*E8	-
2032 **(0x07F0)**	402033	Guaranteed Soak Events	W	*E9	*E9	-
2033 **(0x07F1)**	402034	Wait For Loop	W	*E10	*E10	-
2034 **(0x07F2)**	402035	Wait For Monitor	W	*E11	*E11	-
2035 **(0x07F3)**	402036	Wait For Digital Input	W	*E12	*E12	-
2036 **(0x07F4)**	402037	Wait For Loop/Monitor Setpoint	W	-3276.8	3276.7	-
2037 **(0x07F5)**	402038	Wait For Type/Jump Step	W	*E13	*E13	-
2038 **(0x07F6)**	402039	Jump Count	W	0	999	PV
2039 **(0x07F7)**	402040	Delta Control	W	*E14	*E14	-
2040 **(0x07F8)**	402041	Delta Setpoint	W	-3276.8	3276.7	-
2041 **(0x07F9)**	402042	Loop 1 Setpoint	W	-32768	32767	PV
2042 **(0x07FA)**	402043	Loop 2 Setpoint	W	-32768	32767	PV
2043 **(0x07FB)**	402044	Loop 3 Setpoint	W	-32768	32767	PV
2044 **(0x07FC)**	402045	Loop 4 Setpoint	W	-32768	32767	PV
2045 **(0x07FD)**	402046	Loop 5 Setpoint	W	-32768	32767	PV
2046 **(0x07FE)**	402047	Loop 6 Setpoint	W	-32768	32767	PV
2047 **(0x07FF)**	402048	Loop 7 Setpoint	W	-32768	32767	PV
2048 **(0x0800)**	402049	Loop 8 Setpoint	W	-32768	32767	PV
2049 **(0x0801)**	402050	Loop 9 Setpoint	W	-32768	32767	PV
2050 **(0x0802)**	402051	Loop 10 Setpoint	W	-32768	32767	PV
2051 **(0x0803)**	402052	Not Assigned	W	-	-	-
2052 **(0x0804)**	402053	Not Assigned	W	-	-	-
2053 **(0x0805)**	402054	Not Assigned	W	-	-	-
2054 **(0x0806)**	402055	Not Assigned	W	-	-	-
2055 **(0x0807)**	402056	Not Assigned	W	-	-	-

All remaining steps of the Program follow the same format and data structure as is represented for step one above. Up to the following 2744 registers are used to contain the additional step data of the Program as required for steps 2 through 99. Since few if any programs will contain the maximum of 99 steps, it is only necessary to write the step data for the number steps used in the Program.

Register Numbers:	*Modbus Addresses:*	
2056 (0x0808) – 2083 (0x0823)	402057 – 402084	Program Step 2 Data Registers
2084 (0x0824) – 2111 (0x083F)	402085 – 402112	Program Step 3 Data Registers
2112 (0x0840) – 2139 (0x085B)	402113 – 402140	Program Step 4 Data Registers
2140 (0x085C) – 2167 (0x0877)	402141 – 402168	Program Step 5 Data Registers
2168 (0x0878) – 2195 (0x0893)	402169 – 402196	Program Step 6 Data Registers
2196 (0x0894) – 2223 (0x08AF)	402197 – 402224	Program Step 7 Data Registers
2224 (0x08B0) – 2251 (0x08CB)	402225 – 402252	Program Step 8 Data Registers
2252 (0x08CC) – 2279 (0x08E7)	402253 – 402280	Program Step 9 Data Registers
through		
4772 (0x12A4) – 4799 (0x12BF)	404773 – 404800	Program Step 99 Data Registers

Notes:

*D W Specifies writable data.

*E1

Parameter Value	Description
0	Autostart Off
1	Autostart by Date
2	Autostart by Day

*E2 See note *B10 in Section 5.1 for information on the range of this parameters.

*E3 See note *B11 in Section 5.1 for information on the range of this parameters.

*E4 See note *B12 in Section 5.1 for information on the range of this parameters.

*E5 These parameters contain data which represent up to ten ASCII characters in order to display the name of the currently loaded (or operating) program in the nCompass.

See note *B9 in Section 5.1 for information on the range of these parameters.

*E6

Parameter Value	Range Low	Range High	Description
High Byte	0	59	Minutes
Low Byte	0	59	Seconds

*E7 See note *B5 in Section 5.1 for information on the range of this parameters.

*E8 See note *B6 in Section 5.1 for information on the range of this parameters.

*E9

Parameter Value	Description
Bit0	Guaranteed Soak Loop 1
Bit1	Guaranteed Soak Loop 2
Bit2	Guaranteed Soak Loop 3
Bit3	Guaranteed Soak Loop 4
Bit4	Guaranteed Soak Loop 5
Bit5	Guaranteed Soak Loop 6
Bit6	Guaranteed Soak Loop 7
Bit7	Guaranteed Soak Loop 8
Bit8	Guaranteed Soak Loop 9
Bit9	Guaranteed Soak Loop 10
Bit10-15	Not Assigned

*E10

Parameter Value	Description
Bit0	Wait For Loop 1
Bit1	Wait For Loop 2
Bit2	Wait For Loop 3
Bit3	Wait For Loop 4
Bit4	Wait For Loop 5
Bit5	Wait For Loop 6
Bit6	Wait For Loop 7
Bit7	Wait For Loop 8
Bit8	Wait For Loop 9
Bit9	Wait For Loop 10
Bit10-15	Not Assigned

*E11

Parameter Value	Description
Bit0	Wait For Monitor 1
Bit1	Wait For Monitor 2
Bit2	Wait For Monitor 3
Bit3	Wait For Monitor 4
Bit4	Wait For Monitor 5
Bit5	Wait For Monitor 6
Bit6	Wait For Monitor 7
Bit7	Wait For Monitor 8
Bit8	Wait For Monitor 9
Bit9	Wait For Monitor 10
Bit10	Wait For Monitor 11
Bit11	Wait For Monitor 12
Bit12	Wait For Monitor 13
Bit13	Wait For Monitor 14
Bit14	Wait For Monitor 15
Bit15	Not Assigned

*E12

Parameter Value	Description
Bit0	Wait For Digital Input 1
Bit1	Wait For Digital Input 2
Bit2	Wait For Digital Input 3
Bit3	Wait For Digital Input 4
Bit4	Wait For Digital Input 5
Bit5	Wait For Digital Input 6
Bit6	Wait For Digital Input 7
Bit7	Wait For Digital Input 8
Bit8	Wait For Digital Input 9
Bit9	Wait For Digital Input 10
Bit10	Wait For Digital Input 11
Bit11	Wait For Digital Input 12
Bit12	Wait For Digital Input 13
Bit13	Wait For Digital Input 14
Bit14	Wait For Digital Input 15
Bit15	Not Assigned

*E13
This parameter is split into upper and lower bytes. The high byte of the word is for setting the 'wait for' type while the lower byte contains the step number for jump step operation.

High Byte (0x**00**XX)

Parameter Value	Description
0	Auto
1	Wait for Rising
2	Wait for Falling

Low Byte (0xXX**00**)

Parameter Value	Description
1 - 99	Jump Step

Important: ***If the jump step is set to a value greater than the number of steps in the Program that was loaded, the NCompass will jump to that step if the recycle count for the step is greater than zero, and attempt to execute that step based on the data contained in the step whether it is valid or not.***

*E14

Parameter Value	Description
Bit0	Enable Delta Control for Loop 1
Bit1	Enable Delta Control for Loop 2
Bit2	Enable Delta Control for Loop 3
Bit3	Enable Delta Control for Loop 4
Bit4	Enable Delta Control for Loop 5
Bit5	Enable Delta Control for Loop 6
Bit6	Enable Delta Control for Loop 7
Bit7	Enable Delta Control for Loop 8
Bit8	Enable Delta Control for Loop 9
Bit9	Enable Delta Control for Loop 10
Bit10-15	Not Assigned

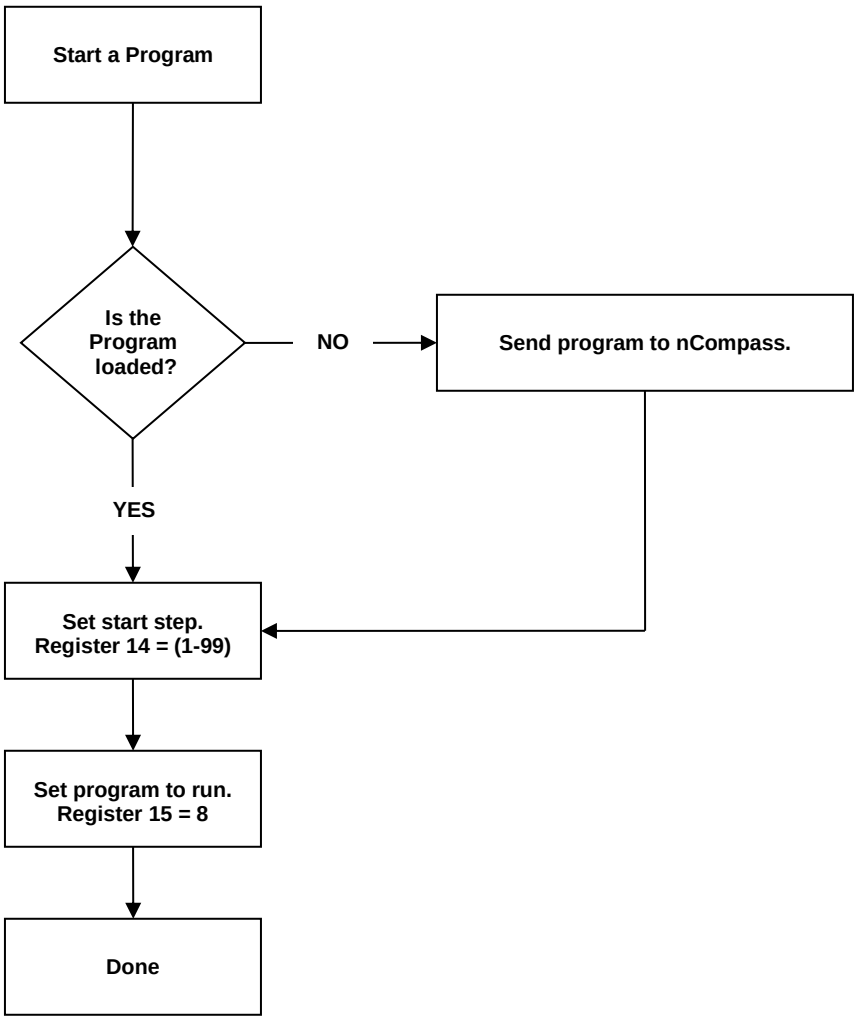
*F The unit PV means that the unit of the parameter is the same as the unit of PV (the loop/monitor input configuration).

Use caution when loading a program to nCompass. You must insure that the decimal point scaling and units of measurement in the program match the loop setting. Loading a program with a temperature setpoint of 80 will result in a control temperature of 80°F if nCompass is in degrees Fahrenheit. However, if nCompass is set for degrees Centigrade, it will result in a control temperature of 80°C (176°F).

5.2.1 Starting an Automatic Ramp/Soak Program on nCompass

Once a program has been downloaded to nCompass, two control parameters must be set in order to start the program. These are the program start step (14) and the program control (15) registers. The program start step register must first be set to the step number that the program is to start on. Note that this must be a value from 1 to the last step of the program.

Once this parameter has been set, the program can be started by setting the program control register to a value of eight (8). This will put the program into run mode and it will begin operation on the step number designated in the program start step parameter. Note that once the program is started, the program start step register (14) will reset to zero. This forces you to set it each time you want to start a program. A program will not start unless this parameter is set. This insures that each time a program is started; it is starting on the proper step number that you designate.



Appendix

Terms and Definitions

address – A unique designator for a location of data or a controller that allows each location or controller on a single communications bus to respond to its own message.

ASCII (pronounced AS-KEY) – American Standard Code for Information Interchange. A universal standard for encoding alphanumeric characters into 7 or 8 binary bits.

Asynchronous – Communications where characters can be transmitted at an unsynchronized point in time. In other words, it can start and stop anytime. The time between transmitted characters may be of varying lengths. Communication is controlled by "start" and "stop" bits at the beginning and end of each character.

Baud – Unit of signaling speed derived from the number of events per second (i.e., bits per second).

Baud rate – The rate of information transfer in serial communications, measured in bits per second.

Binary – Number based system where only two characters exist, 0 and 1. Counting is 0, 1, 10, 11...

Bit – Derived from "**Bi**nary digi**t**", a one or zero condition in the binary system.

Byte – A term referring to eight associated bits of information, sometimes called a "character".

Character – Letter, numeral, punctuation, control figure or any other symbol contained in a message. Typically this is encoded in one byte.

Communications – The use of digital computer messages to link components. (See serial communications and baud rate)

Converter – This device will convert from one hardware interface to another such as from EIA-232 to EIA-485. The converter may be transparent to the software, which means you do not have to give any special considerations to software programming.

CRC – When data is corrupted during transmission, a method is used to return the data to its correct value. This can be accomplished through several methods: parity, checksum and CRC (cyclic redundancy checksum) are three of these. **C**yclic **R**edundancy **C**hecksum is an error-checking mechanism using a polynomial algorithm based on the content of a message frame at the transmitter and included in a field appended to the frame. At the receiver, it is then compared with the results of the calculation that is performed by the receiver.

Data – The information that is transferred across the communications bus. This may be a setpoint, setup parameter, or any character. This information is transferred to an address or register.

DB-9 – A standardized connector shaped like the letter "D" when viewed on edge. This connector has 9 contacts. It is utilized on most IBM AT compatible PCs as the serial port.

Decode – This is the reverse of encode. When a piece of data has information embedded in it, decode is to extract that information. Example: to extract an "A" from 01000001.

Duplex – The ability to send and receive data at the same time. "To listen and talk at the same time."

EIA-232 – Electronic Industries Association developed this standard hardware interface to allow one device to talk to another device in full duplex mode. This method uses a differential voltage between one wire and ground. Also called an unbalanced system since the ground wire carries the sum of current of all lines. Transmission is limited to about 50 feet.

EIA-485 – Electronic Industries Association developed this standard hardware interface to allow up to 32 devices to be on a bus at one time. This method uses a differential voltage between two wires. Also called a balanced system since each wire carries the same current value. This has the advantage of being immune to outside electrical disturbances.

EIA/TIA -232 and -485 – Data communications standards set by the Electronic Industries Association and Telecommunications Industry Association. Formerly referred to as RS- (Recommended Standard). (See EIA-232 and EIA-485)

Electronic Industries Association (EIA) – An association in the US that establishes standards for electronics and data communications.

Encode – To embed information into a piece of data. This is the reverse of decode. Example: let 01000001 stand for an "A".

Error Correction – When an inconsistency is in the data, a method is used to detect and/or return the data to its correct value. This can be done through several methods, parity, checksum and CRC (cyclic redundancy checksum) area three of these.

Even – This term is used with parity. See parity.

Firmware – Instruction or data stored in an IC (integrated circuit) or on a read only disk. This data is programmed once and cannot easily be changed as software can.

Full Duplex – Full is used to mean the duplex's full capability. The ability to send and receive data at the same time. The same as duplex.

GPIB – See IEEE488

Half Duplex – The ability to send or receive data, but not at the same time. "To listen or talk, but not both at the same time."

Handshake (Handshaking) – Exchange of predetermined signals between two devices establishing a connection. Using extra wires or software signals to coordinate communications, signals can be sent to tell the transmitter the current status of the other device receiver. Example: Are you busy or are you ready?

Hex or Hexadecimal – Number based system where sixteen characters exist, 0 to 9, A to F. Counting is 0..9,A,B,C...

Integer – Two bytes make an integer. This contains 16 bits. An integer can represent a decimal value of -32768 to 32767.

Logic Level – A voltage measurement system where only two stable voltage values exist. Example: 0v and 5V, or -3v and +3v.

Mark – Represents the transmission of data bit logic 1 (see logic level). Usually this is the most negative voltage value in serial communications.

Master – The device on the bus that controls all communications. Only the master can initiate conversation.

Modbus – A software protocol developed by Gould Modicon (now AEG) for process control systems. No hardware interface is defined. Modbus is accessed on the master/slave principle, the protocol providing for one master and up to 247 slaves. Only the master can initiate a transaction. This is a
half duplex protocol.

Network – When two or more devices share communication lines, the devices are “networked”.

Node – A point of interconnection to a network.

Noise Immunity – The ability of communication lines to ignore electrical noise generated in the lines by nearby magnetic and electrostatic fields.

Odd – This term is used with parity. See parity.

Parallel – Communication using this method, transfers eight bits or one byte at a time over eight data wires and one ground wire. This method is eight times faster than using serial but utilizes more hardware.

Parity – A bit is assigned at the beginning of a byte to stand for parity. When the ‘1’ bits are counted, the number will be even or odd. A parity bit is used to ensure that the answer is always even if even parity or odd if odd parity. If the receiving end counts the ‘1’ bits and the sum is not the same odd or even, an error is generated. Parity is used to detect errors caused by noise in data transmission.

Protocol – A set of rules for communication. This will specify what method to transfer information, packet size, information headers and who should talk when. It is used to coordinate communication activity.

Receive – To accept data sent from another device. The device that receives the data is the receiver.

Register – An area of memory that provides temporary storage of digital data.

Slave – A device that only responds to commands. This device never starts communication on it’s own. Only the Master can do this. (See Master)

SCADA – Supervisory Control and Data Acquisition

Serial – To process something in order. First item, second item, etc.

Serial Communications – A method of transmitting information between devices by sending all bits serially (see serial) over a single communication channel.

Software – Information of data or program stored in an easily changeable format. (RAM, Floppy Disk, Hard Disk)

Space – Represents the transmission of a data bit logic 0 (see logic level). Usually this is the most positive voltage value in serial communications.

Start Bit – A binary bit or logic level that represents when the serial data information is about to start (at the beginning of a character or byte). This voltage level is positive.

Stop Bit – A binary bit or logic level that represents when the serial data information is complete (at the end of a character or byte). This voltage level is negative.

Synchronous – When data is transmitted on a data line and a clock signal is used on another line to determine when to check the data line for a logic level. This clock is said to “synchronize” the data.
Transmit – To send data from one device to another. The device that sends the data is the transmitter.

Word – Two bytes make a word. This contains 16 bits. A word can represent a decimal value of 0 to 65535.

Software Usage Note:

The selection, application and use of Future Design Control products or software is the sole responsibility of the purchaser or end user. No claims will be allowed for any damages or losses, whether direct, indirect, incidental, special or consequential.

In addition, Future Design reserves the right to make changes without notification to purchaser or user to materials or processing that do not affect compliance with any applicable specification. Future Design makes no warranties when using the nCompass system.

Warranty:

Future Design Controls products described in this book are warranted to be free from functional defects in material and workmanship at the time the products shipped from Future Design Controls facilities and to conform at that time to the specifications set forth in the relevant Future Design Controls manual, sheet or sheets for a period of one year after delivery to the first purchaser.

Future Design Controls FDC-0450 products are warranted to be free from functional defects in materials and workmanship at the time the products shipped from Future Design Controls facilities and to conform at that time to the specifications set forth in the relevant Future Design Controls manual, sheet or sheets for a period of one year after delivery to the first purchaser for use.

There are no expressed or implied Warranties extending beyond the Warranties herein and above set forth. Limitations: Future Design Controls provides no warranty or representations of any sort regarding the fitness of use or application of its products by the purchaser. Users are responsible for the selection, suitability of the products for their application or use of Future Design Controls products.

Future Design Controls shall not be liable for any damages or losses, whether direct, indirect, incidental, special, consequential or any other damages, costs or expenses excepting only the cost or expense of repair or replacement of Future Design Control products as described below.

Future Design Controls sole responsibility under the warranty, at Future Design Controls option, is limited to replacement or repair, free of charge, or refund of purchase price within the warranty period specified. This warranty does not apply to damage resulting from transportation, alteration, misuse or abuse.

Future Design Controls reserves the right to make changes without notification to purchaser to materials or processing that do not affect compliance with any applicable specifications.

Return Material Authorization:

Contact Future Design Controls for Return Material Authorization Number prior to returning any product to our facility:

7524 West 98th Place – Bridgeview, IL 60455 – Phone 888.751.5444 – Fax 888.307.8014

http://www.futuredesigncontrols.com