GS1 Modbus Communications

Chapter 5

Contents of this Chapter

Unless otherwise stated, numeric data is in the unsigned decimal data format.

Communications Parameters Summary (P9.xx)

A summary of the GS1 Communications Parameters is listed below. For a complete listing of the GS1 Parameters, refer to Chapter 4.

GS1 Parameter Summary – Communications Parameters (P9.xx)

Parameter	Description	Range	Default
P9.00	Communication Address	1 to 254	1
P9.01	Transmission Speed	0: 4800 baud 1: 9600 baud 2: 19200 baud	1
P9.02	Communication Protocol	0: MODBUS ASCII mode, 7 data bits, no parity, 2 stop bits 1: MODBUS ASCII mode, 7 data bits, even parity, 1 stop bit 2: MODBUS ASCII mode, 7 data bits, odd parity, 1 stop bit 3: MODBUS RTU mode, 8 data bits, no parity, 2 stop bits 4: MODBUS RTU mode, 8 data bits, even parity, 1 stop bit 5: MODBUS RTU mode, 8 data bits, odd parity, 1 stop bit	0
P9.03	Transmission Fault Treatment	0: Display fault and continue operating 1: Display fault and RAMP to stop 2: Display fault and COAST to stop 3: No fault displayed and continue operating	0
P9.04	Time Out Detection	0: Disable 1: Enable	0
P9.05	Time Out Duration	0.1 to 60.0 seconds	0.5
♦ P9.07	Parameter Lock	0: All parameters can be set and read 1: All parameters are read-only	0
P9.08	**Restore to Default**	99: Restores all parameters to factory defaults	0
♦ P9.11	Block Transfer Parameter 1	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.12	Block Transfer Parameter 2	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.13	Block Transfer Parameter 3	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.14	Block Transfer Parameter 4	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.15	Block Transfer Parameter 5	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.16	Block Transfer Parameter 6	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.17	Block Transfer Parameter 7	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.18	Block Transfer Parameter 8	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.19	Block Transfer Parameter 9	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.20	Block Transfer Parameter 10	Parameters 0.00 to 8.01, 9.99	9.99
♦ P9.26	**Serial Comm Speed Reference**	0.0 to 400.0 Hz	60.0
♦ P9.27	**Serial Comm RUN Command**	0: Stop 1: Run	0
♦ P9.28	**Serial Comm Direction Command**	0: Forward 1: Reverse	0
♦ P9.29	**Serial Comm External Fault**	0: No fault 1: External fault	0
♦ P9.30	**Serial Comm Fault Reset**	0: No action 1: Fault Reset	0
♦ P9.31	**Serial Comm JOG Command**	0: Stop 1: Jog	0
P9.39 *	**Firmware Version**	#.##	#.##
P9.41	GS Series Number	1: GS1 2: GS2 3: GS3 4: GS4	##
P9.42	Manufacturer Model Information	0: GS1-10P2 (120V, 1ph, 0.25HP) 1: GS1-10P5 (120V, 1ph, 0.5HP) 2: GS1-20P2 (230V, 1ph/3ph, 0.25HP) 3: GS1-20P5 (230V, 1ph/3ph, 0.5HP) 4: GS1-21P0 (230V, 1ph/3ph, 1HP) 5: GS1-22P0 (230V, 3ph, 2HP)	##
** This parameter is available only with AC drive firmware v1.07 or higher.*			
♦ Parameter can be set during RUN Mode.			

GS1 Parameter Memory Addresses

The octal address also can be used in the WX / RX instruction of the DL-250-1, DL-450, and DL05.

Parameter Memory Addresses – Motor Parameters (P0.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P0.00	Motor Nameplate Voltage	0000	40001	0
P0.01	Motor Nameplate Amps	0001	40002	1
P0.02	Motor Base Frequency	0002	40003	2
P0.03	Motor Base RPM	0003	40004	3
P0.04	Motor Maximum RPM	0004	40005	4
** For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Ramp Parameters (P1.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P1.00	Stop Methods	0100	40257	400
♦ P1.01	Acceleration Time 1	0101	40258	401
♦ P1.02	Deceleration Time 1	0102	40259	402
P1.03	Accel S-curve	0103	40260	403
P1.04	Decel S-curve	0104	40261	404
♦ P1.05	Acceleration Time 2	0105	40262	405
♦ P1.06	Deceleration Time 2	0106	40263	406
P1.07	Select method to use 2nd Accel/Decel	0107	40264	407
P1.08	Accel 1 to Accel 2 frequency transition	0108	40265	410
P1.09	Decel 1 to Decel 2 frequency transition	0109	40266	411
P1.10	Skip Frequency 1	010A	40267	412
P1.11	Skip Frequency 2	010B	40268	413
P1.12	Skip Frequency 3	010C	40269	414
P1.17	Skip Frequency Band	0111	40274	421
P1.19	DC Injection Voltage Level	0113	40276	423
P1.20	DC Injection during Start-up	0114	40277	424
P1.21	DC Injection during Stopping	0115	40278	425
P1.22	Start-point for DC Injection	0116	40279	426
♦ Parameter can be set during RUN Mode.				
** For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Volts/Hertz Parameters (P2.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P2.00	Volts/Hertz Settings	0200	40513	1000
♦ P2.01	Slip Compensation	0201	40514	1001
♦ P2.03	Manual Torque Boost	0203	40516	1003
P2.04	Mid-point Frequency	0204	40517	1004
P2.05	Mid-point Voltage	0205	40518	1005
P2.06	Min. Output Frequency	0206	40519	1006
P2.07	Min. Output Voltage	0207	40520	1007
P2.08	PWM Carrier Frequency	0208	40521	1010
♦ Parameter can be set during RUN Mode.				
** For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Digital Parameters (P3.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P3.00	Source of Operation Command	0300	40769	1400
P3.01	Multi-function Inputs 1 & 2 (DI1 & DI2)	0301	40770	1401
P3.02	Multi-function Input 3 (DI3)	0302	40771	1402
P3.03	Multi-function Input 4 (DI4)	0303	40772	1403
P3.11	Multi-Function Output Terminal	030B	40780	1413
♦ P3.16	Desired Frequency	0310	40785	1420
♦ P3.17	Desired Current	0311	40786	1421
♦ *Parameter can be set during RUN Mode.*				
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Analog Parameters (P4.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P4.00	Source of Frequency Command	0400	41025	2000
P4.01	Analog Input Offset Polarity	0401	41026	2001
♦ P4.02	Analog Input Offset	0402	41027	2002
♦ P4.03	Analog Input Gain	0403	41028	2003
P4.04	Analog Input Reverse Motion Enable	0404	41029	2004
P4.05	Loss of ACI Signal (4–20mA)	0405	41030	2005
♦ *Parameter can be set during RUN Mode.*				
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Presets Parameters (P5.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
♦ P5.00	Jog	0500	41281	2400
♦ P5.01	Multi-Speed 1	0501	41282	2401
♦ P5.02	Multi-Speed 2	0502	41283	2402
♦ P5.03	Multi-Speed 3	0503	41284	2403
♦ *Parameter can be set during RUN Mode.*				
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*				

Parameter Memory Addresses – Protection Parameters (P6.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P6.00	Electronic Thermal Overload Relay	0600	41537	3000
P6.01	**Auto Restart after Fault**	0601	41538	3001
P6.02	**Momentary Power Loss**	0602	41539	3002
P6.03	Reverse Operation Inhibit	0603	41540	3003
P6.04	Auto Voltage Regulation	0604	41541	3004
P6.05	Over-Voltage Trip Prevention	0605	41542	3005
P6.06	Auto Adjustable Accel/Decel	0606	41543	3006
P6.07	Over-Torque Detection Mode	0607	41544	3007
P6.08	Over-Torque Detection Level	0608	41545	3010
P6.09	Over-Torque Detection Time	0609	41546	3011
P6.10	Over-Current Stall Prevention during Acceleration	060A	41547	3012
P6.11	Over-Current Stall Prevention during Operation	060B	41548	3013
P6.12	Maximum Allowable Power Loss Time	060C	41549	3014
P6.13	Base-Block Time for Speed Search	060D	41550	3015
P6.14	Maximum Speed Search Current Level	060E	41551	3016
P6.15	Upper Bound of Output Frequency	060F	41552	3017
P6.16	Lower Bound of Output Frequency	0610	41553	3020
P6.30 **	**Line Start Lockout**	061E	41567	3036
P6.31	Present Fault Record	061F	41568	3037
P6.32	Second Most Recent Fault Record	0620	41569	3040
P6.33	Third Most Recent Fault Record	0621	41570	3041
P6.34	Fourth Most Recent Fault Record	0622	41571	3042
P6.35	Fifth Most Recent Fault Record	0623	41572	3043
P6.36	Sixth Most Recent Fault Record	0624	41573	3044

♦ *Parameter can be set during RUN Mode.*
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*
** *This parameter is available only with AC drive firmware v1.07 or higher (refer to P9.39 for firmware version).*

Parameter Memory Addresses – Display Parameters (P8.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
♦ P8.00	User Defined Display Function	0800	42049	4000
♦ P8.01	Frequency Scale Factor	0801	42050	4001

♦ *Parameter can be set during RUN Mode.*
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*

Parameter Memory Addresses – Communications Parameters (P9.xx)				
GS1 Parameter	***Description***	***Hexadecimal***	***Modbus Decimal ****	***Octal***
P9.00	Communication Address	0900	42305	4400
P9.01	Transmission Speed	0901	42306	4401
P9.02	Communication Protocol	0902	42307	4402
P9.03	Transmission Fault Treatment	0903	42308	4403
P9.04	Time Out Detection	0904	42309	4404
P9.05	Time Out Duration	0905	42310	4405
♦ P9.07	Parameter Lock	0907	42312	4407
P9.08	**Restore to Default**	0908	42313	4410
♦ P9.11	Block Transfer Parameter 1	090B	42316	4413
♦ P9.12	Block Transfer Parameter 2	090C	42317	4414
♦ P9.13	Block Transfer Parameter 3	090D	42318	4415
♦ P9.14	Block Transfer Parameter 4	090E	42319	4416
♦ P9.15	Block Transfer Parameter 5	090F	42320	4417
♦ P9.16	Block Transfer Parameter 6	0910	42321	4420
♦ P9.17	Block Transfer Parameter 7	0911	42322	4421
♦ P9.18	Block Transfer Parameter 8	0912	42323	4422
♦ P9.19	Block Transfer Parameter 9	0913	42324	4423
♦ P9.20	Block Transfer Parameter 10	0914	42325	4424
♦ P9.26	Serial Comm Speed Reference	091A	42331	4432
♦ P9.27	Serial Comm RUN Command	091B	42332	4433
♦ P9.28	Serial Comm Direction Command	091C	42333	4434
♦ P9.29	Serial Comm External Fault	091D	42334	4435
♦ P9.30	Serial Comm Fault Reset	091E	42335	4436
♦ P9.31	Serial Comm JOG Command	091F	42336	4437
P9.39 **	**Firmware Version**	0927	42344	4447
P9.41	GS Series Number	0929	42346	4451
P9.42	Manufacturer Model Information	092A	42347	4452

♦ *Parameter can be set during RUN Mode.*
* *For Modbus Decimal addresses used with CLICK PLCs, insert another zero as the next-to-most-significant digit, e.g., 402333 instead of 42333.*
** *This parameter is available only with AC drive firmware v1.07 or higher.*

GS1 Status Addresses

The GS1 Series AC drive has status memory addresses that are used to monitor the AC drive. The status addresses and value definitions are listed below.

Status Addresses (Read Only)

GS1 Status Addresses			
Description	***Hexadecimal***	***Modbus Decimal***	***Octal***
Status Monitor 1	2100	48449	20400
Status Monitor 2	2101	48450	20401
Frequency Command F	2102	48451	20402
Output Frequency H	2103	48452	20403
Output Current A	2104	48453	20404
DC Bus Voltage d	2105	48454	20405
Output Voltage U	2106	48455	20406
Motor RPM	2107	48456	20407
Scale Frequency (Low Word)	2108	48457	20410
Scale Frequency (High Word)	2109	48458	20411
% Load	210B	48460	20413
Firmware Version	2110	48465	20420

Status Monitor 1 – Error Codes **h2100**

00: No fault occurred
01: Over-current(oc)
02: Over-voltage(ov)
03: Overheat (oH)
04: Overload (oL)
05: Overload 1 (oL1)
06: Overload 2 (oL2)
07: External Fault (EF)
08: CPU Failure 1 (cF1)
09: CPU Failure 2 (cF2)
10: CPU Failure 3 (cF3)
11: Hardware Protection Failure (HPF)
12: Over-current during accel (ocA)
13: Over-current during decel (ocd)
14: Over-current during steady state (ocn)
16: Low Voltage (Lv)
18: External Base-Block (bb)
19: Auto Adjust accel/decel Failure (cFA)
20: Software Protection Code (codE)

Some error codes will not display under status address if only a warning message. The drive must have a hard trip. To manually check this, set "External Fault" to Terminal Control, and trip. This will simulate the result of a hard trip.

Status Monitor 2 h2101

GS1 Memory Address (hexadecimal)	GS1 Memory Data (binary)																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bits
2101	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	
	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	Bit Values (decimal)

Address Bit(s)	*Bit(s) Value Binary (Decimal)*	*AC Drive Status*
0 and 1	00 (0)	Drive operation stopped (STOP)
	01 (1)	Run to Stop transition
	10 (2)	Standby
	11 (3)	Drive operation running (RUN)
2	1 (4)	JOG active
3 and 4	00 (0)	Rotational direction forward (FWD)
	01 (8)	REV to FWD transition
	10 (16)	FWD to REV transition
	11 (24)	Rotational direction reverse (REV)
5	1 (32)	Source of frequency determined by serial comm interface (P4.00 = 5)
6	1 (64)	Source of frequency determined by AI terminal (P4.00 = 2, 3, or 4)
7	1 (128)	Source of operation determined by serial comm interface (P3.00 = 3 or 4)
8	1 (256)	Parameters have been locked (P9.07 = 1)
9 ~ 15	N/A	Reserved

Frequency Command F (xxx.x) h2102

Status location for the frequency setting of the AC drive.

Output Frequency H (xxx.x) h2103

Status location for the actual operating frequency present at terminals T1, T2, and T3.

Output Current A (xxx.x) h2104

Status location for the output current present at terminals T1, T2, and T3.

DC BUS Voltage d (xxx.x) h2105

Status location for the DC Bus Voltage.

Output Voltage U (xxx.x) h2106

Status location for the output voltage present at terminals T1, T2, and T3. (This is the RMS voltage between phases.)

Motor RPM h2107

Status location for the present estimated speed of the motor.

Scale Frequency (Low word) h2108

Status location for result of output frequency x P8.01 (low word).

Scale Frequency (High word) h2109

Status location for result of output frequency x P8.01 (high word).

% Load h210B

Status location for the amount of load on the AC drive. (Output Current ÷ Drive Rated Current) x 100.

Firmware Version h2110

Status location for firmware version of the AC drive.

Block Transfer Parameters for Modbus Programs

For writing to any of the parameters from P0.00 to P8.01, a group of 10 block transfer parameters (P9.11 to P9.20) is available in the GS1 AC drive. This sequential block of parameters can be used to “group” various miscellaneous non-sequential parameters, so that you can update the parameters in one programming write block instead of having to use multiple write commands.

For example, it would typically take three different write commands to change the three non-sequential parameters Accel Time 1 (P1.01), Accel S-curve (P1.03), and Multi-speed 1 (P5.01).

However, you could make the same three changes using one write command by setting P9.11 to P1.01, P9.12 to P1.03, and P9.13 to P5.01, so that the parameters become sequential.

Communicating with AutomationDirect PLCs

The following steps explain how to connect and communicate with GS1 AC drives using AutomationDirect PLCs.

GS1 drives have a provision for shutting down control or power to the inverter in the event of a communications time out. This feature can be set up through parameters P9.03, P9.04, and P9.05.

Step 1: Choose the Appropriate CPU

The GS1 AC drives will communicate with the following AutomationDirect PLCs using Modbus communications.

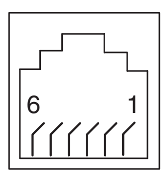
- *Modbus control is easier to accomplish from a DirectLOGIC PLC with an RS-485 port and MRX/ MWX, or from a CLICK PLC using Send/Receive instructions.*

Choose Your CPU	
Primary Choices	CLICK Analog CPU with Send/Receive instructions & RS-485 comm port D2-260 or DL06 with MRX / MWX instructions & RS-485 comm port
Secondary Choices	CLICK Basic CPU with Send/Receive instructions & RS-232 comm port DL05, D2-250(-1), or D4-450 with RX / WX instructions & RS-232 comm port

Step 2: Make the Connections

GS1 RS-485 Serial Comm Port

GS1 Serial Comm Port
RS-485 Interface
RJ12 (6P4C)



1: +17V
2: GND
3: SG-
4: SG+
5: nc
6: reserved

The GS1 Comm Port requires an RS-485 input. RS-232 signals can be converted to RS-485 by using a separate converter.

PLC Connections for RS-485 Modbus RTU Control of GS1 Drive

Drive	PLC *	PLC Port *	Communication	Direct Cable	Length
GS1	CLICK	3	RS-485	ZL-RJ12-CBL-2P ***	2m [6.6 ft] ***
	DL05	2 **	RS-232 – RS-485 **	N/A **	
	DL06 D0-DCM	2	RS-485	**GS-485HD15-CBL-2 *****	2m [6.6 ft] ***
	D2-DCM D2-250(-1)	2 **	RS-232 – RS-485 **	N/A **	
	D2-260	2	RS-485	**GS-485HD15-CBL-2 *****	2m [6.6 ft] ***
	D4-450	3 **	RS-232 – RS-485 **	N/A **	

** If a PLC type or port is not listed in this chart, it cannot function as a Modbus RTU master.*
*** Requires RS-232–RS-485 converter & generic cabling options described later in this chapter.*
**** Termination resistors not required due to short cable length.*

RS-485 Connections For Multiple Drives

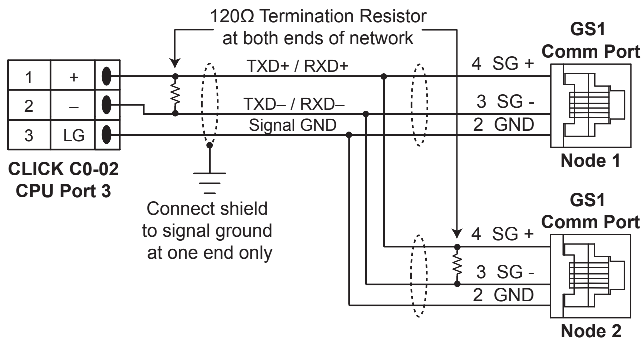
ZIPLink™ RS-485 communication boards (ZL-CDM-RJ12X4 or ZL-CDM-RJ12X10) provide an easy means to break out the RS-485 signal to several drives at one location, which creates a star configuration. However, the transmission errors are negligible, so this configuration is acceptable for proper operation of the VFDs.

RS-485 Direct Connections

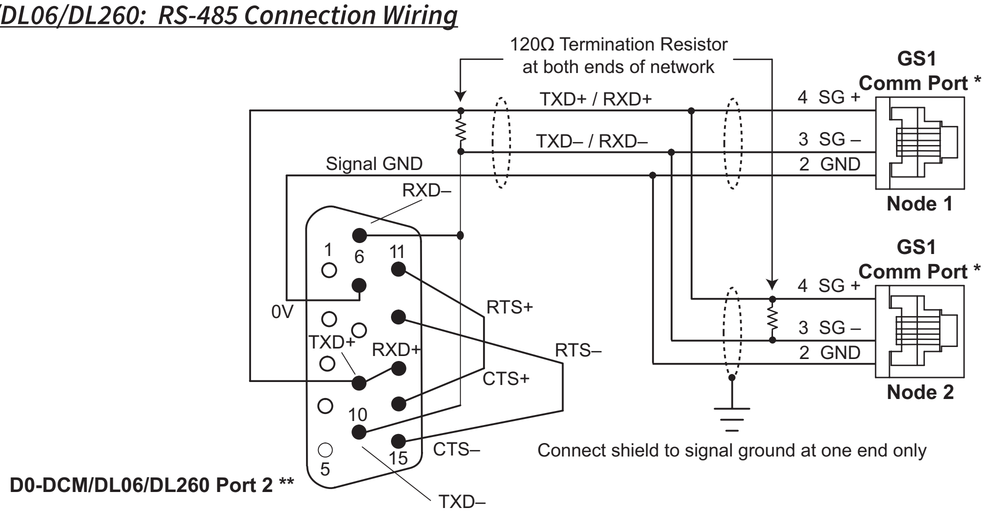
Termination Resistors are required on both ends of RS-485 networks; especially on long runs. Select resistors that match the impedance rating of the cable (between 100 and 500W).

Recommended RS-485 cable: Belden 9842 or equivalent.

CLICK C0-02: RS-485 Connection Wiring



D0-DCM/DL06/DL260: RS-485 Connection Wiring



* Consider using ZIPLink RJ12 Feedthrough Modules ZL-RTB-RJ12 for easy wiring termination.
** Consider using ZIPLink 15-pin high-density Comm Port Adapter, ZL-CMA15 or ZL-CMA15L, for easy wiring termination.

For Single Cable Runs of 2m (6.6 ft) or less to only one AC Drive: Use pre-terminated cable GS-485HD15-CBL-2 for easy wiring.

RS-232C to RS-485 Conversion

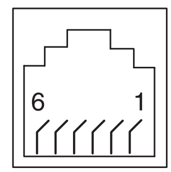
An RS-485 network cable can span up to 1000 meters (4000 feet). However, most DirectLOGIC PLCs have only RS-232C communication ports, and require an FA-ISOCON (RS-232C to RS-422/485 network adapter) in order to make an RS-485 connection.

If an FA-ISOCON module is used, set the module DIP switches as required.
Refer to the FA-ISOCON manual for more detailed information.

FA-ISOCON Switch Settings:

S21~S23	OFF, ON, ON (19200 baud)
S24~S27	OFF (Automatic Network Transmit Enable)
Terminate	ON (end of run term resistors)
Bias (2)	ON (end of run bias resistors)
1/2 DPX (2)	ON (RS-485 TXD/RXD jumpers)

FA-ISOCON RJ-12 Serial Comm Port A
RS-232 Input Port

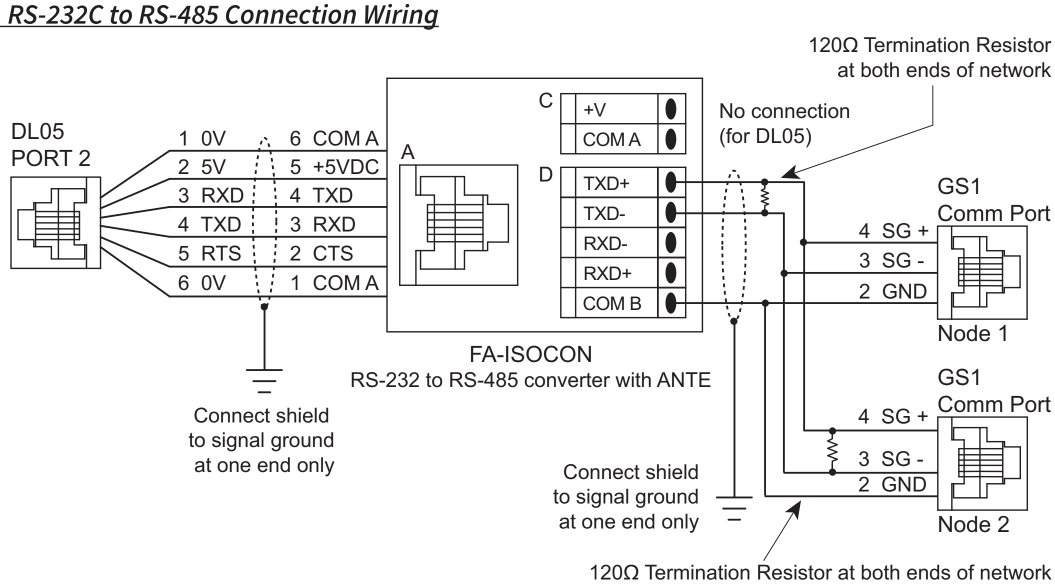


1: Signal Ground
2: CTS (input)
3: RXD (input)
4: TXD (output)
5: +5VDC in
6: Signal Ground

Use the following wiring diagrams to connect DirectLOGIC RS-232C PLCs to a GS1 Series AC drive with an FA-ISOCON network adapter module:

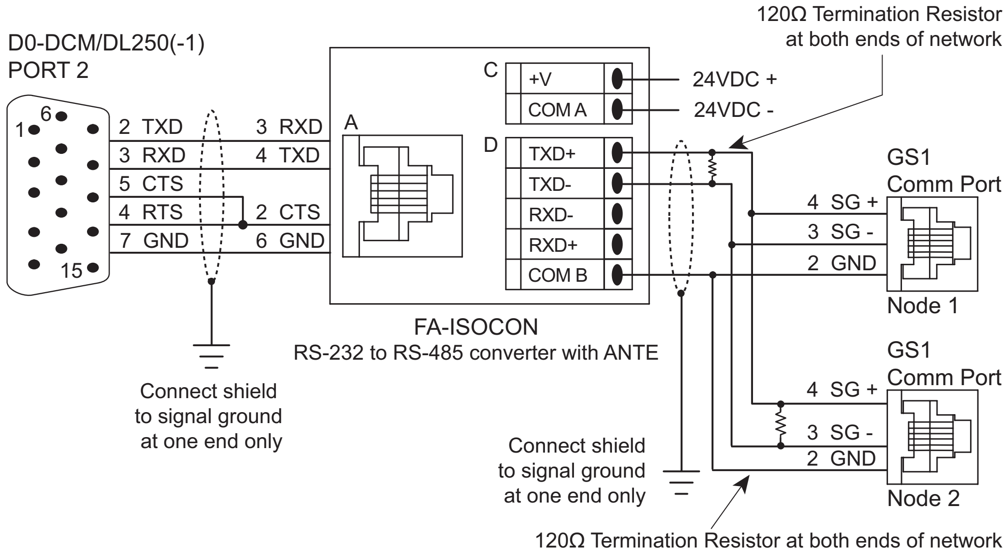
Recommended cable for RS-232: Belden 8102 or equivalent.
Recommended cable for RS-485: Belden 9842 or equivalent.
Various pre-terminated cables for specific wiring connections are available from AutomationDirect, as listed in applicable individual wiring sections of this chapter.

DL05: RS-232C to RS-485 Connection Wiring



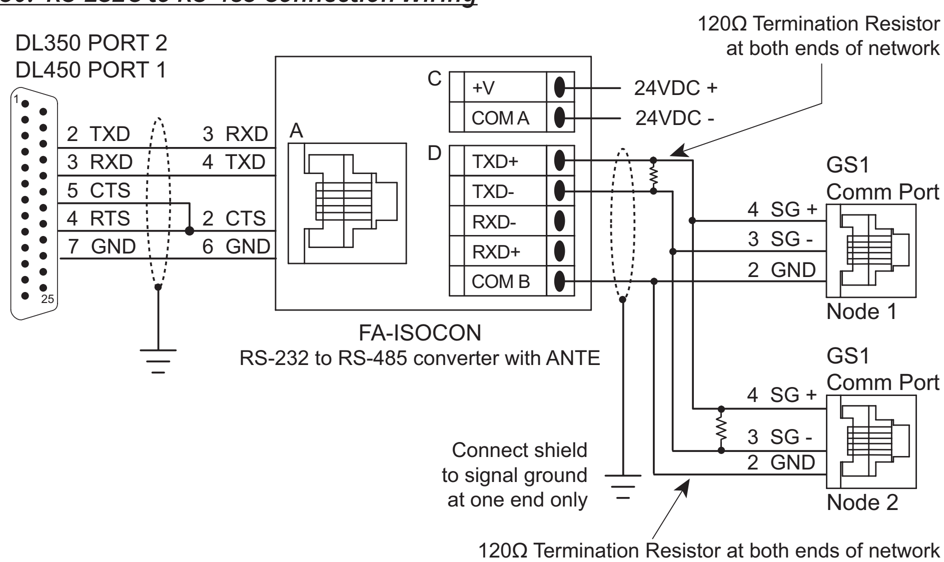
Cable D0-DSCBL (12ft; 3.7m) is available for connecting the DL05 to the FA-ISOCON.
For a single run to only one AC Drive, cable GS-ISOCON-CBL-2 (2m; 6.6ft) is available for directly connecting the FA-ISOCON to the GS1 Com Port.

D0-DCM/DL250(-1): RS-232C to RS-485 Connection Wiring



A cable that will connect the D0-DCM or DL250(-1) to the FA-ISOCON can be constructed using the FA-15HD adapter and the D0-CBL cable. A cable can also be constructed using the FA-15HD adapter and RJ12-6P6C cable from the FA-CABKIT.

DL350/DL450: RS-232C to RS-485 Connection Wiring



A cable that will connect the DL450 to the FA-ISOCON can be constructed using the DB25-pin-male-to-RJ12 adapter and the RJ12-6P6C cable from the FA-CABKIT.

Ethernet Connection using GS-EDRV(100)

The GS-EDRV(100) provides an Ethernet link between a control system and a GS1 AC drive. It mounts on DIN rail and connects a drive to an Ethernet hub/switch or PC. The GS-EDRV(100) processes signals to and from the drive. It formats the signals to conform with the Ethernet standard to the H2-ERM(100) or H4-ERM(100), KEPdirect EBC I/O server, or independent controller with a MODBUS TCP/IP driver. This Ethernet interface allows for great connectivity to many control system architectures. An additional feature is the built-in web browser which allows users to configure and control the drive from any web browser via the IP address of the GS-EDRV(100) card.

Step 3: Set AC Drive Parameters

The following parameters need to be set as shown in order to communicate properly.

• *P3.00:*	*03 or 04*	*Operation Determined by RS-485 interface. Keypad STOP is enabled (03) or disabled (04).*
• *P4.00:*	*05*	*Frequency determined by RS-485 communication interface.*
• *P9.00:*	*xx*	*Communication address 1-254 (unique for each device, see P9.00).*
• *P9.01:*	*01*	*9600 baud data transmission speed (higher baud rate setting may be required with FA-ISOCON network adapter; set adapter DIP switches accordingly).*
• *P9.02:*	*05*	*MODBUS RTU mode <8 data bits, odd parity, 1 stop bit>.*

This list of parameter settings is the minimum required to communicate with a DirectLOGIC PLC. There may be other parameters that need to be set to meet the needs of your particular application.

Step 4: Configure the PLC CPU

The PLC CPUs must be configured to communicate with the GS1 AC drives. This configuration includes setting up the communication port and adding instructions to your logic program.

The set up for all of the AutomationDirect PLC CPUs is very similar, although there are some subtle differences between CPUs. Refer to the appropriate CPU User Manual for the specifics on your specific PLC CPU if more details are needed.

For instructions on Modbus Configuration for your specific PLC CPU, refer to the appropriate PLC User Manual.

Configure the CLICK PLC

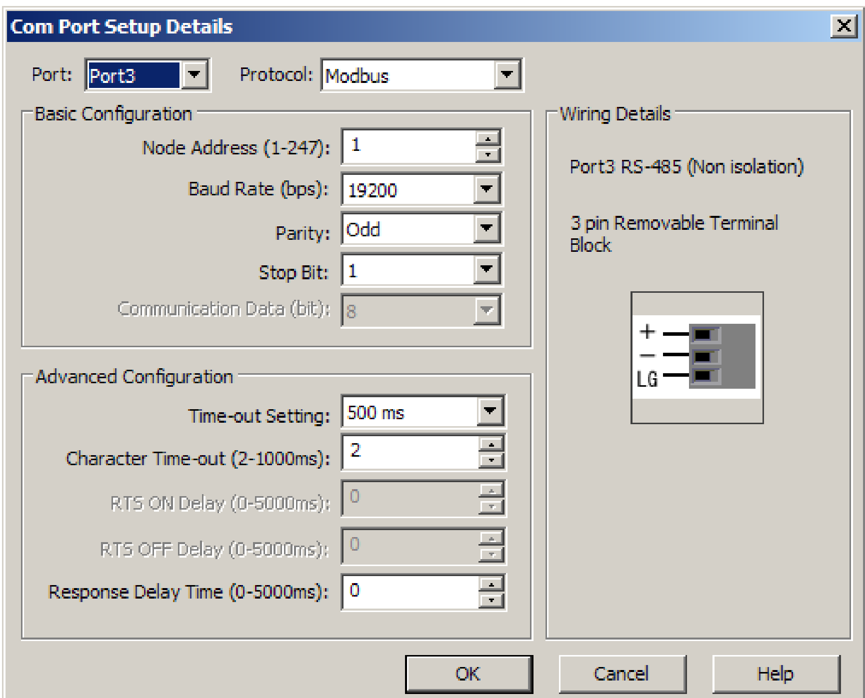
Configure the CLICK CPU communication port before writing communication instructions into your logic program.

For more detailed instructions on Modbus Configuration for your CLICK, refer to the CLICK PLC Hardware User Manual, C0-USER-M, or to the CLICK software help file.

CLICK Port 3 MODBUS Configuration for RS-485

The following configuration example is specific for CLICK PLC CPUs.

- *Configure the communication port before writing communication instructions into the logic program.*
- *In CLICK programming software, open the "Comm Port Details Setup" dialog box by choosing the Setup menu, then Comm Port Setup, then Port 2 Setup.*
- *From the "Port:" list box, choose "**Port 3**."*
- *For the "Protocol:" list box, select "**Modbus**."*
- *Set the "Node Address" to "**1**" to make the CLICK PLC a MODBUS master.*
- *Set the "Baud Rate" to "**19200**."*
- *Set the "Parity" to "**Odd**."*
- *Set the "Stop Bit" to "**1**."*
- *Set the "Time-out Setting" to "**500ms**."*
- *Set the "Response Delay Time" to "**0ms**."*



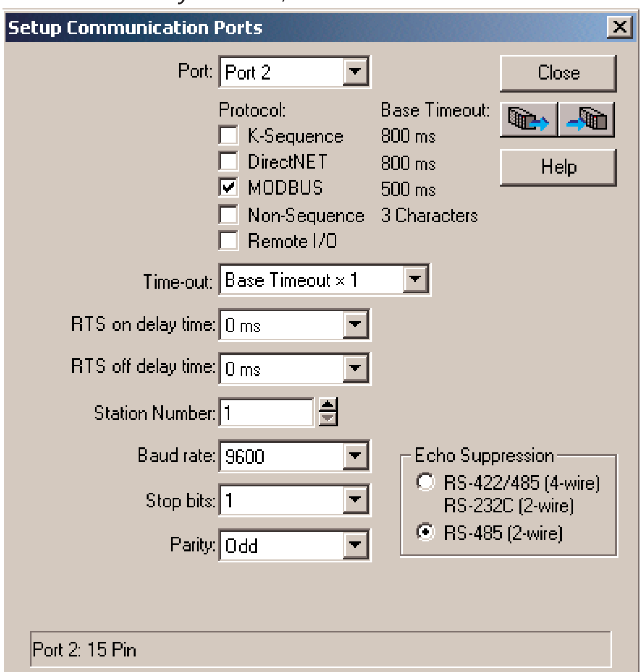
The communication port settings are saved in the project file. The project must be transferred to the CLICK PLC in order for any port setting changes to take effect.

Configure the DirectLOGIC CPUs

DirectLOGIC MODBUS Port Configuration for D2-260 and DL06

The following configuration example is specific to the D2-260 and DL06. Refer to the appropriate CPU User Manual for the specifics on your DirectLOGIC CPU.

- *In DirectSOFT, choose the PLC menu, then Setup, then "**Secondary Comm Port.**"*
- *From the Port number list box at the top, choose "**Port 2**."*
- *For the Protocol, select ONLY "**MODBUS**." (Do not select multiple protocols.)*
- *Response Delay Time should be "**0ms**." Both RTS on and off delay times must be set to 0ms.*
- *The Station Number should be set to "**1**" to make the D2-260 or DL06 CPU a MODBUS master.*
- *The Baud Rate should be set at "**9600**."*
- *In the Stop Bits list box, Choose "**1**."*
- *In the Parity list box, choose "**Odd**."*



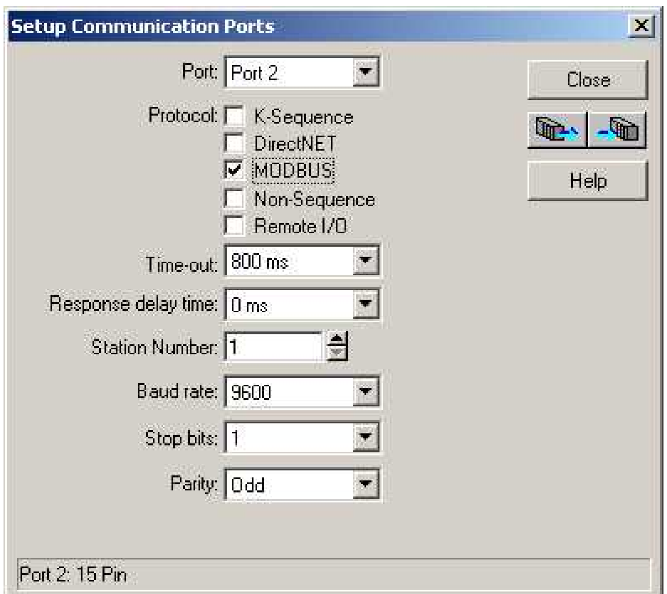
Configure the DirectLOGIC CPUs (continued)

DirectLOGIC MODBUS Port Configuration for DL05, D2-250(-1), and D4-450

The following configuration example is specific to the D2-250(-1) and DL05. Refer to the appropriate CPU User Manual for the specifics on your DirectLogic CPU.

- *In DirectSOFT, choose the PLC menu, then Setup, then “**Secondary Comm Port**.”*
- *From the Port list box, choose “**Port 2**.”*
- *For the Protocol, select ONLY “**MODBUS**.” (Do not select multiple protocols.)*
- *In the Timeout list box, select “**800ms**.”*
- *Response Delay Time should be “**0ms**.”*
- *The Station Number should be set to “**1**” to make the D2-250(-1) or DL05 CPU a MODBUS Master.*
- *The Baud Rate should be set at “**9600**” (or higher, if using an FA-ISOCON network adapter module).*
- *In the Stop Bits list box, choose “**1**.”*
- *In the Parity list box, choose “**Odd**.”*

The DL250 network instructions used in Master mode will access only slaves 1 to 90. Each slave must have a unique number.



CLICK Modbus Ladder Programming

The set up for all of the CLICK CPUs is very similar. However, there may be some subtle differences between CPUs, or for the requirements of your particular program. Refer to the CLICK programming software internal help file for more information regarding CLICK programming.

The following ladder program shows some examples of how to control the GS1 AC drive through Modbus RTU. The drive should be set up and tested for communications before it is connected to a load.

Warning: A drive should never be connected to a load until any applicable communication programs have been proven.

Warning: Write programs in such a way that the program does not erroneously overwrite a remote Stop command with a Run command, such as when P3.00 is set to 03. This example program prevents such an accidental overwrite.

These programs are for illustrational purposes only, and are not intended for a true application.

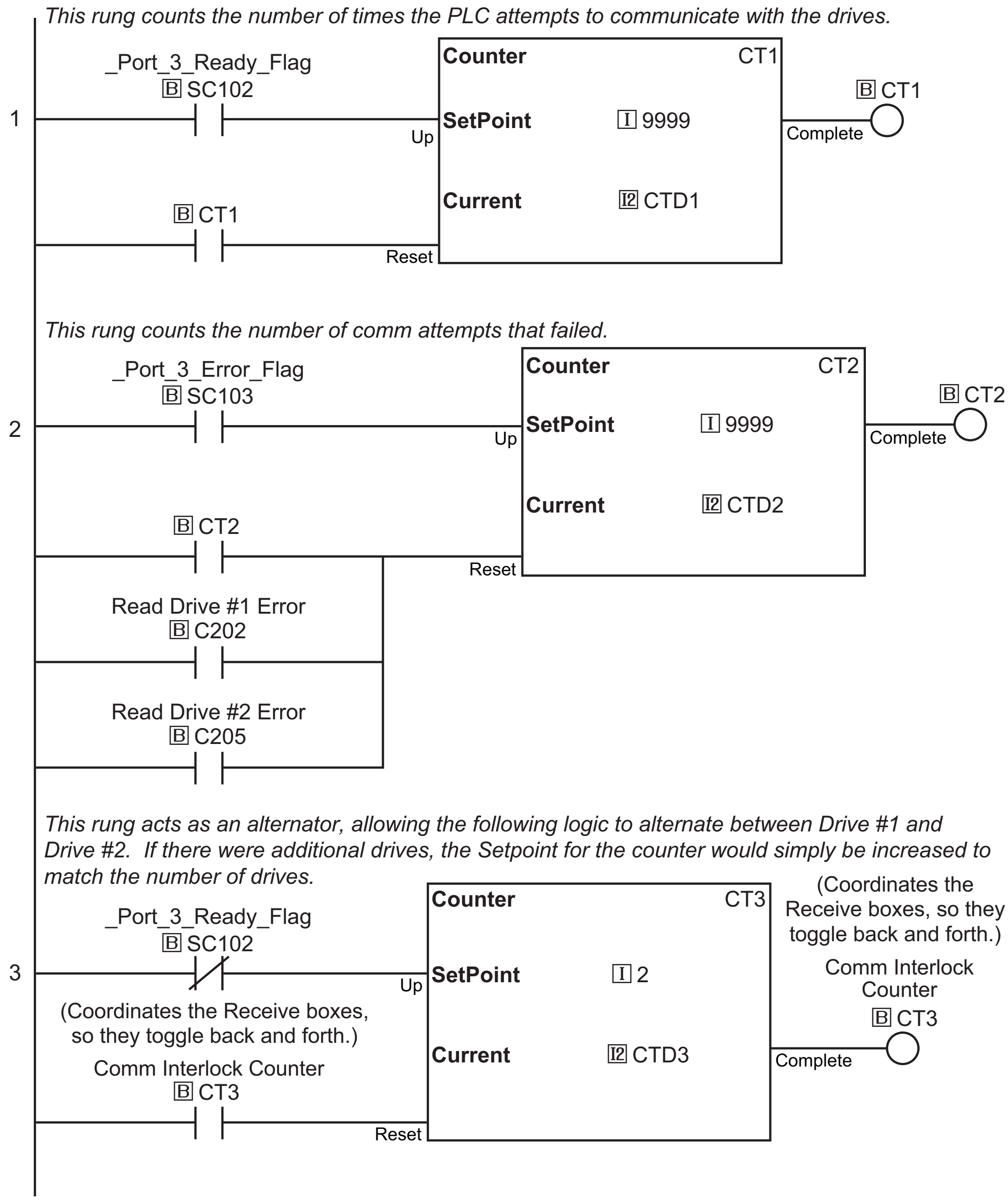
Separate Run Command Write Instruction

Why do we write the Run Command with a separate write instruction? If we write the Run Command to the drive along with the Speed Reference, Direction, External Fault, and Fault Reset Commands, we can keep the parameter addresses in sequence, and we can update all five of the commands with one write instruction. This method is valid only if we disable the drive's keypad STOP button (P3.00 = 04).

Typically, the keypad STOP button will be enabled (P3.00 = 03), and we need to prevent a change in one of the other commands from overriding a keypad Stop Command by causing a previous Run Command to be rewritten to the drive. By using a separate Run Command write instruction, only a deliberate Run Command change by the program will run the drive again after a stop.

CLICK Communication Program Example – (for CLICK PLCs)

This program is for illustrational purposes only, and is not intended for a true application.

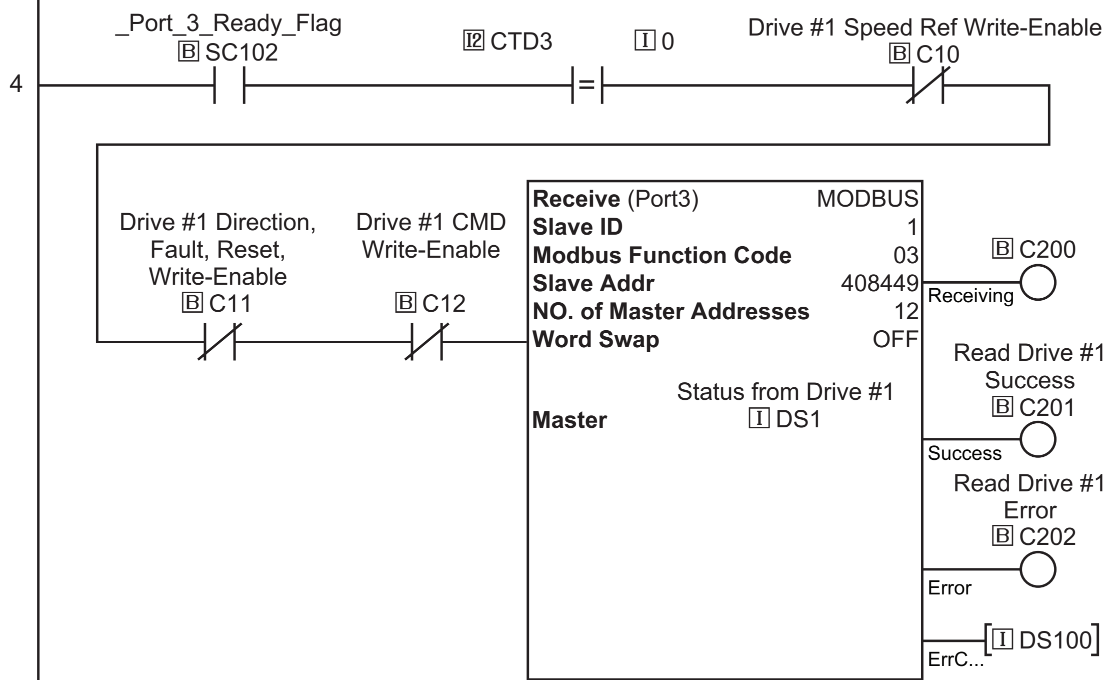


(continued next page – CLICK PLC communication program example)

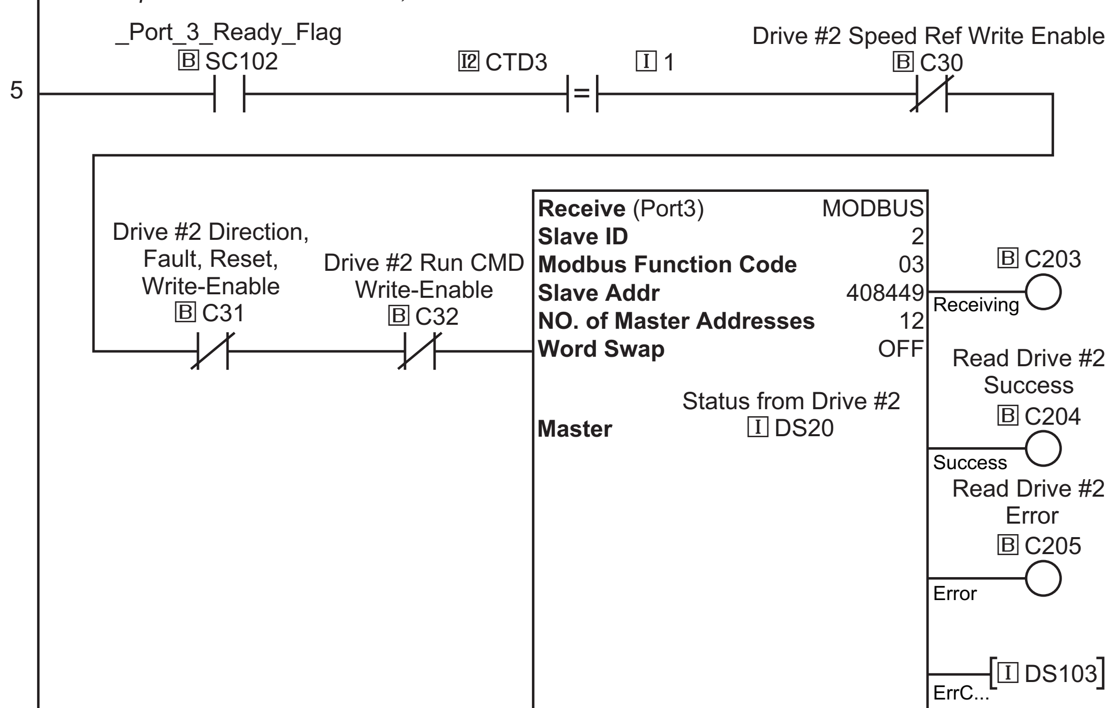
(continued from previous page – CLICK PLC communication program example)

This program is for illustrational purposes only, and is not intended for a true application.

This rung checks to see if it is time to communicate to Drive #1, and also if there are no current Write requests to that drive. If not, it reads data from Drive #1.



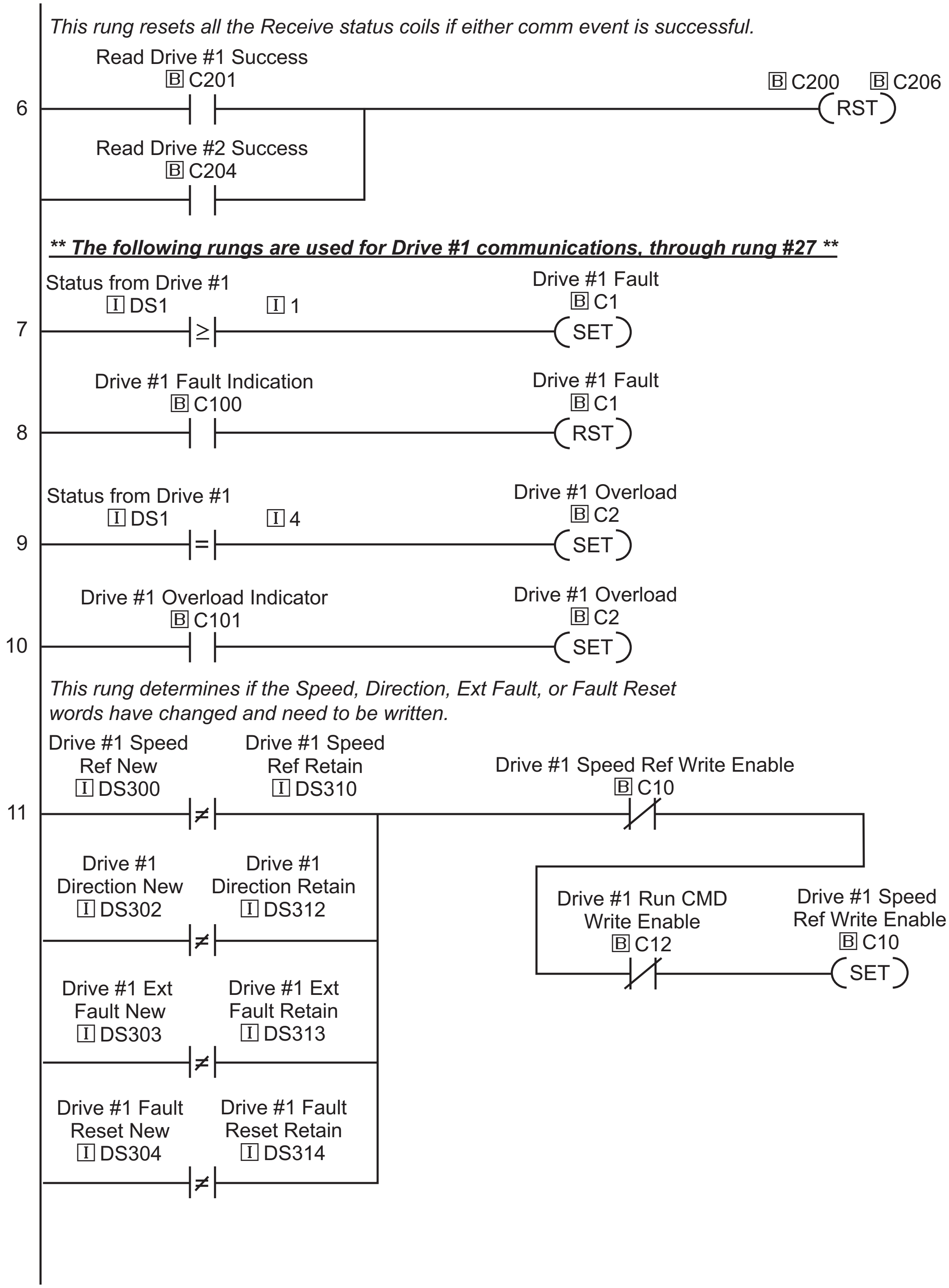
This rung checks to see if it is time to communicate to Drive #2, and also if there are no current Write requests to that drive. If not, it reads data from Drive #2.



(continued next page – CLICK PLC communication program example)

(CONTINUED FROM PREVIOUS PAGE – CLICK PLC COMMUNICATION PROGRAM EXAMPLE)

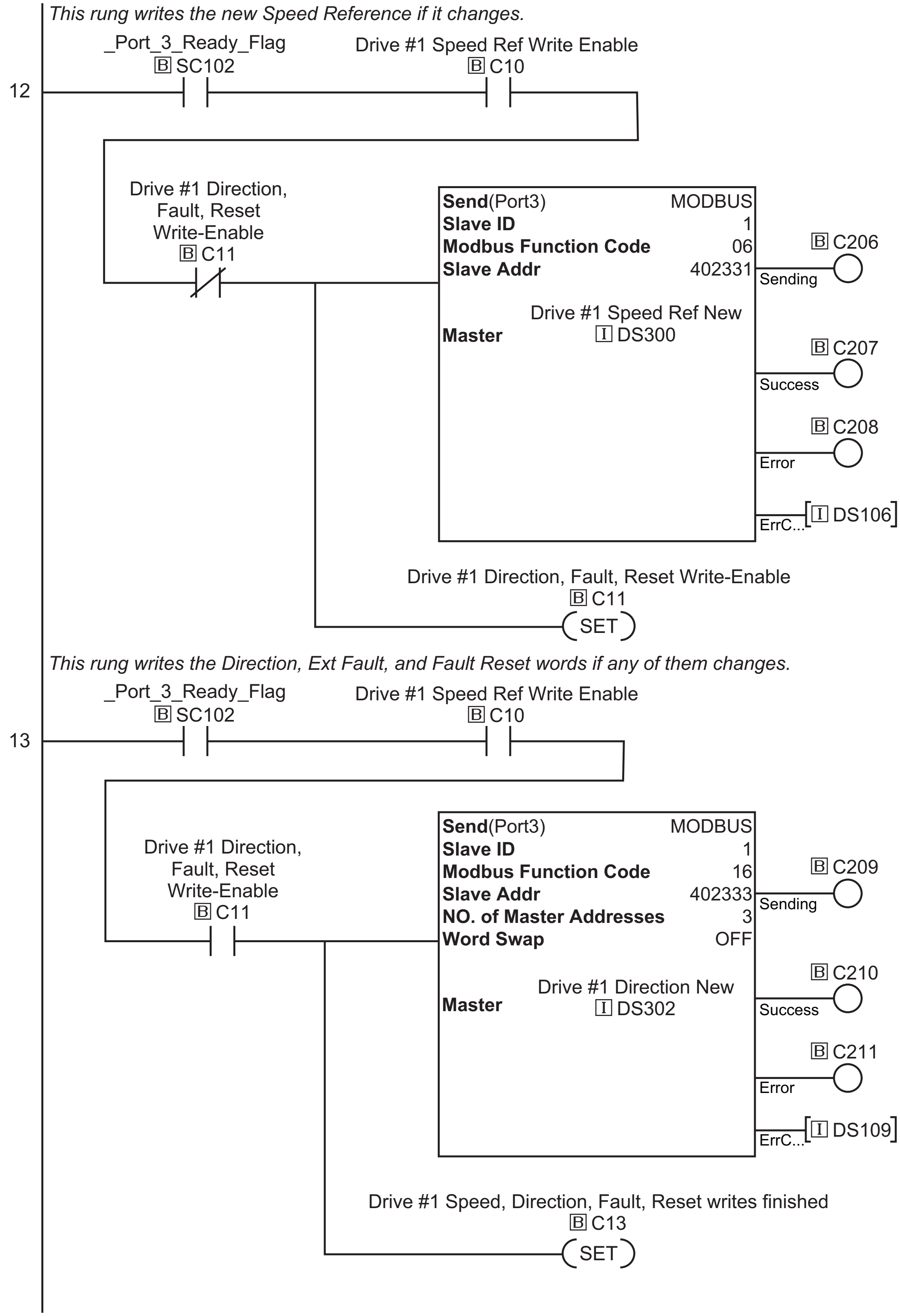
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(Continued from previous page – CLICK PLC communication program example)

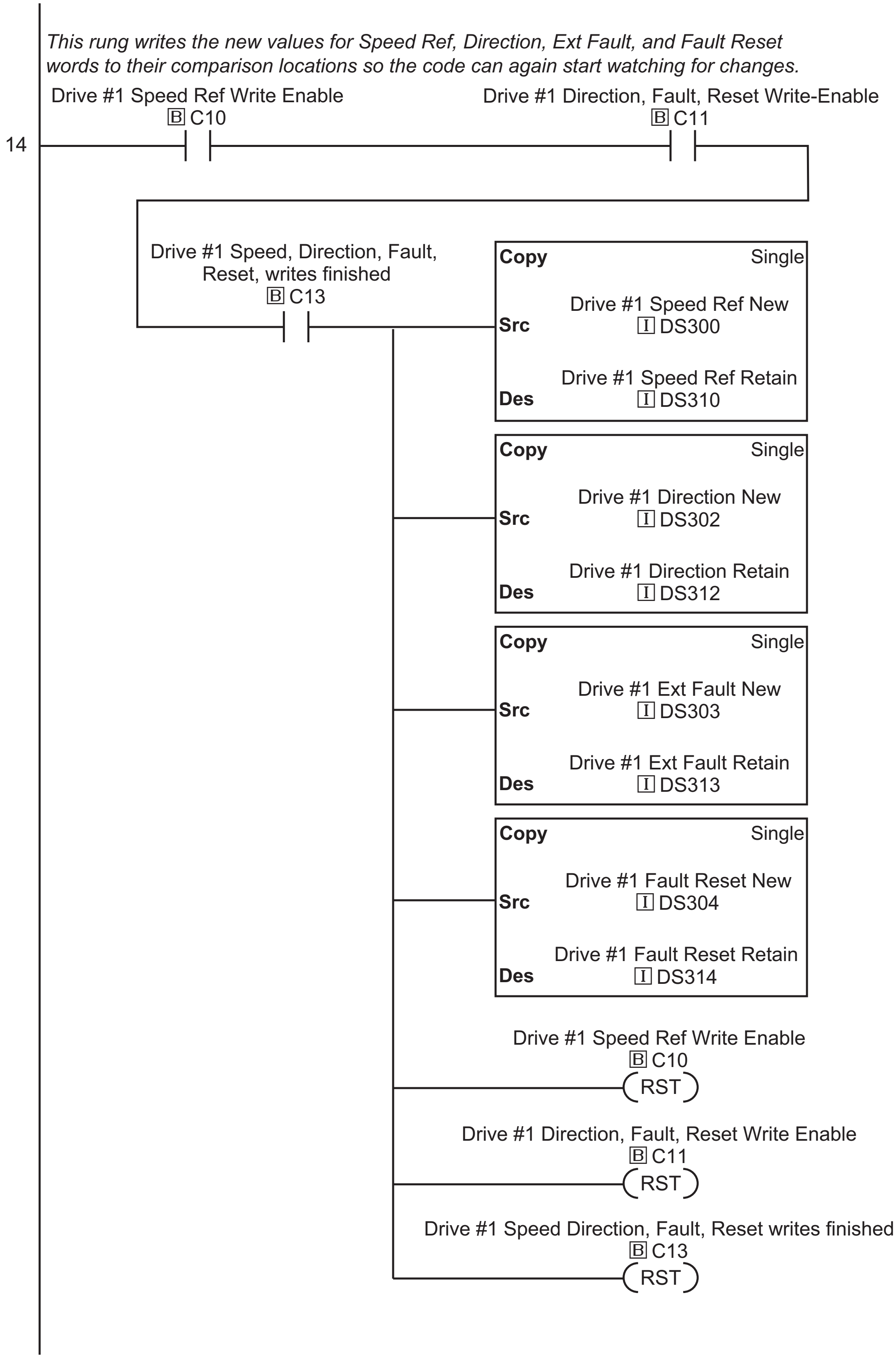
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

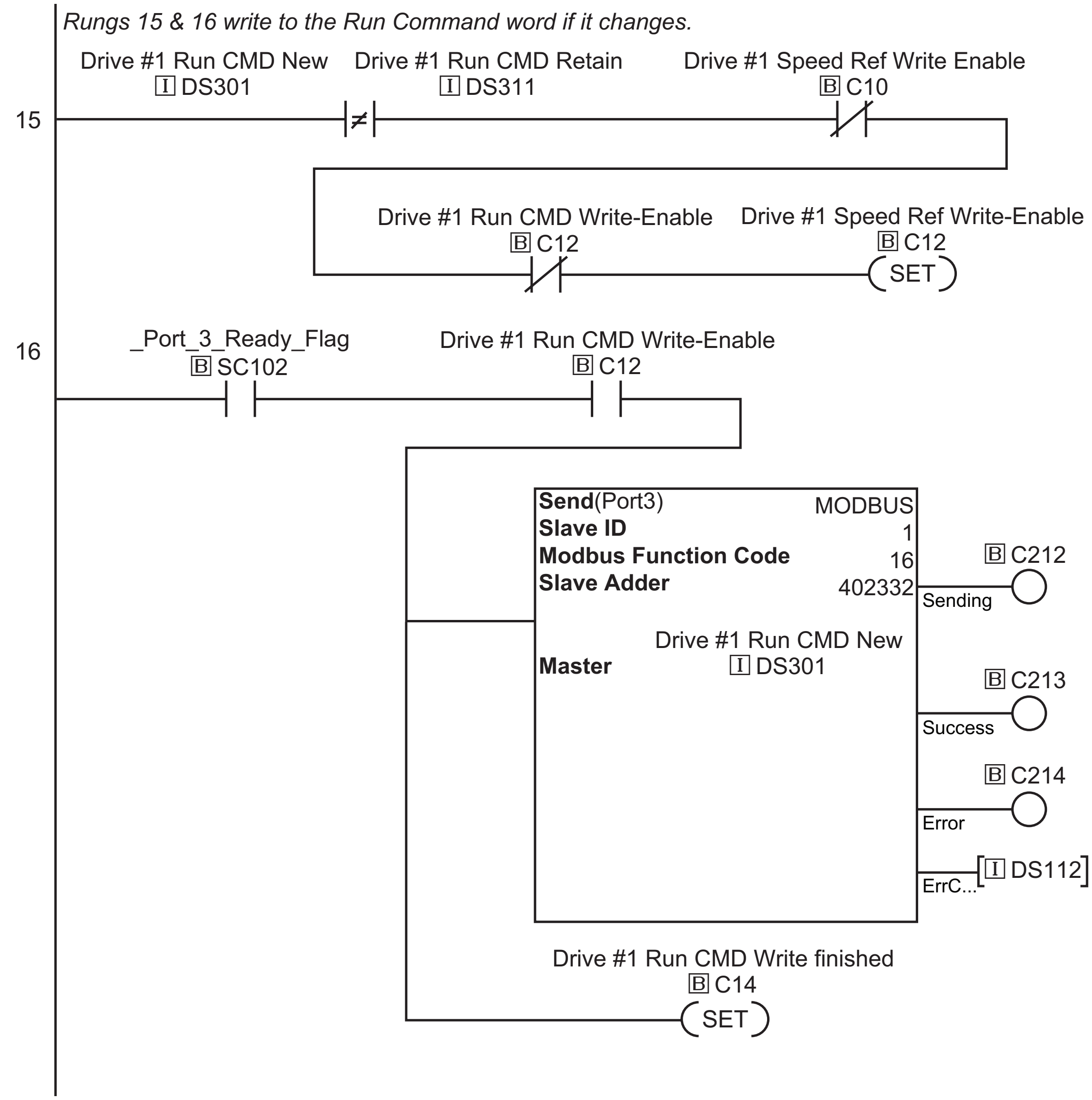
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

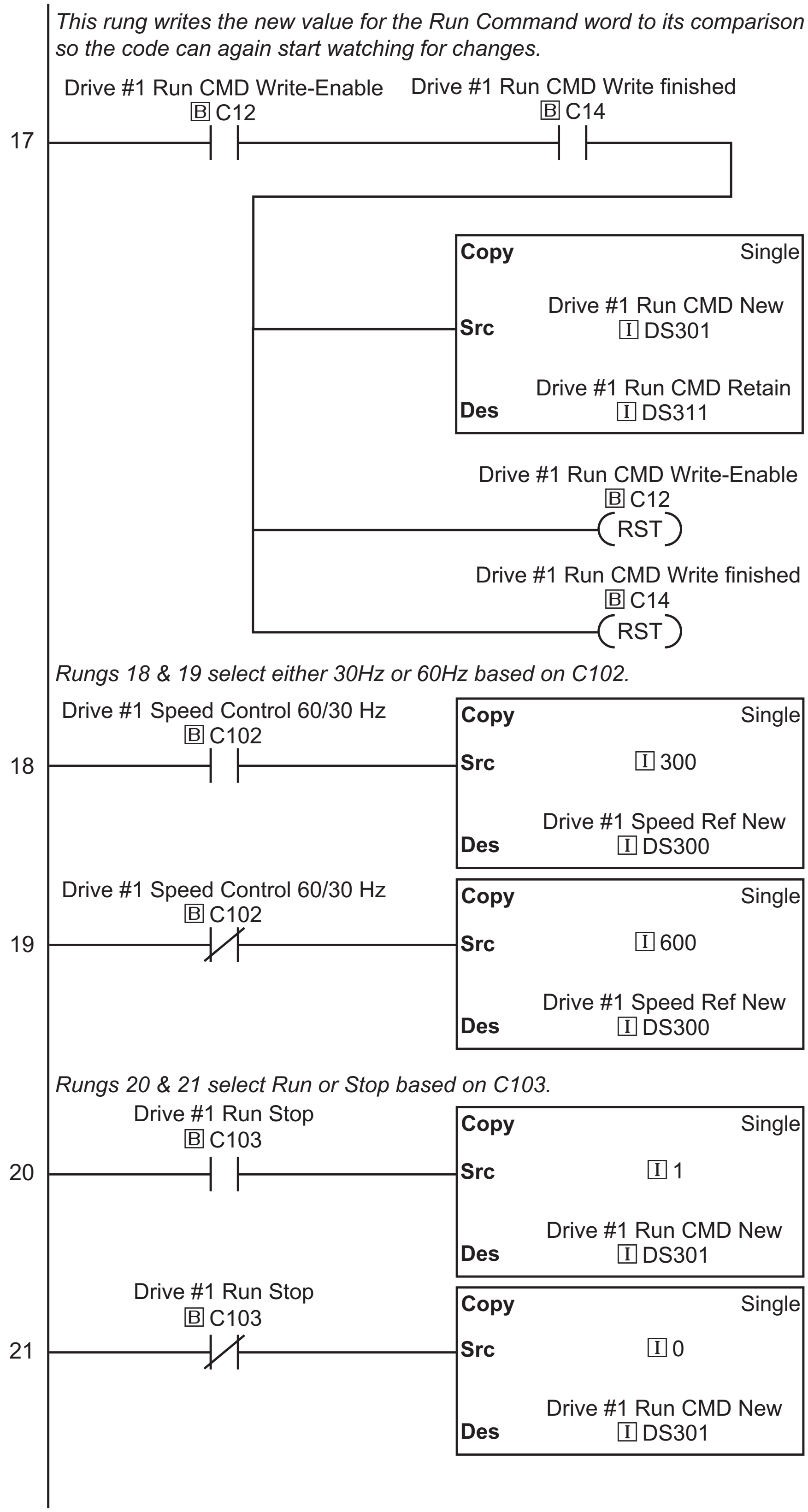
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

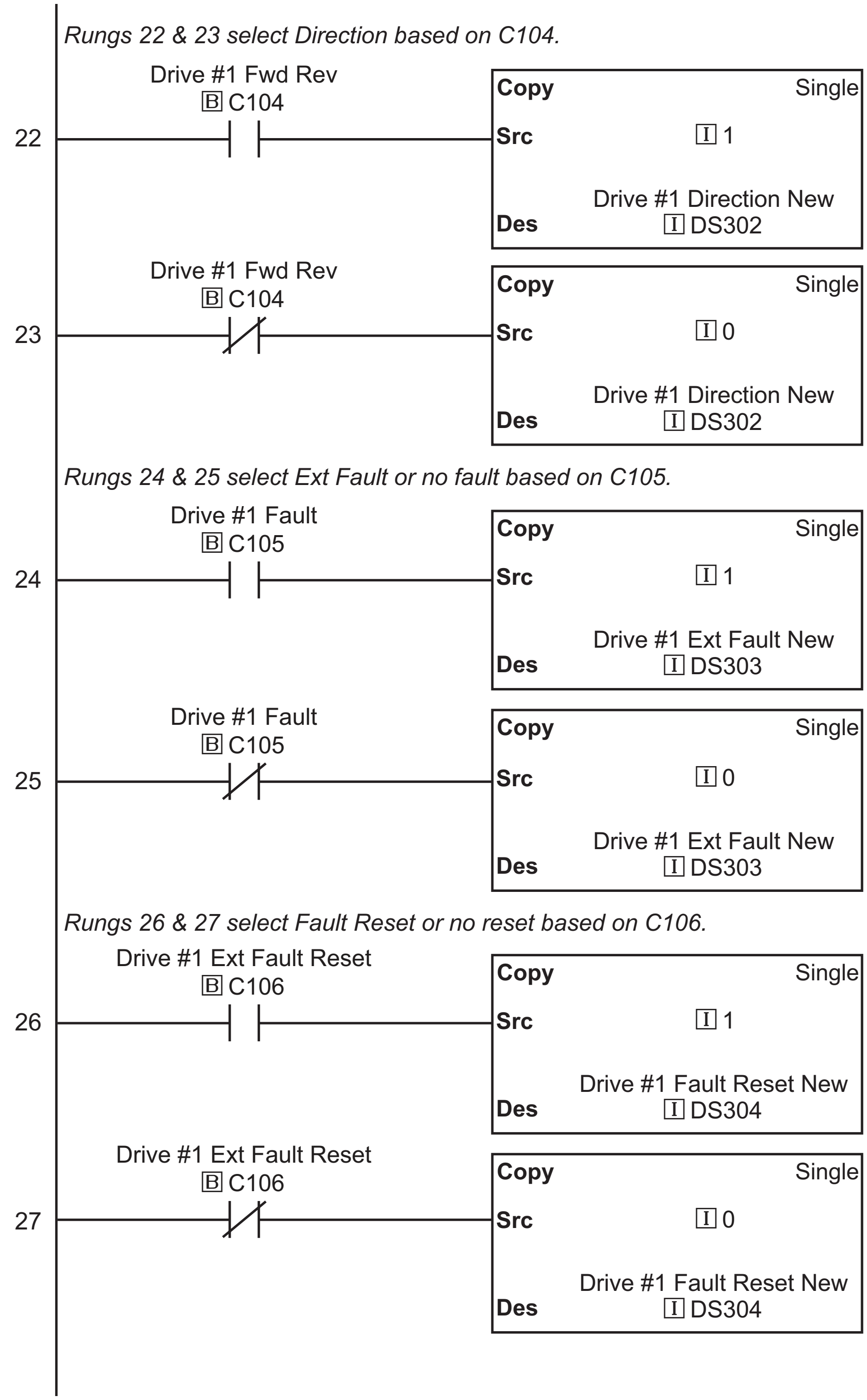
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(CONTINUED FROM PREVIOUS PAGE – CLICK PLC COMMUNICATION PROGRAM EXAMPLE)

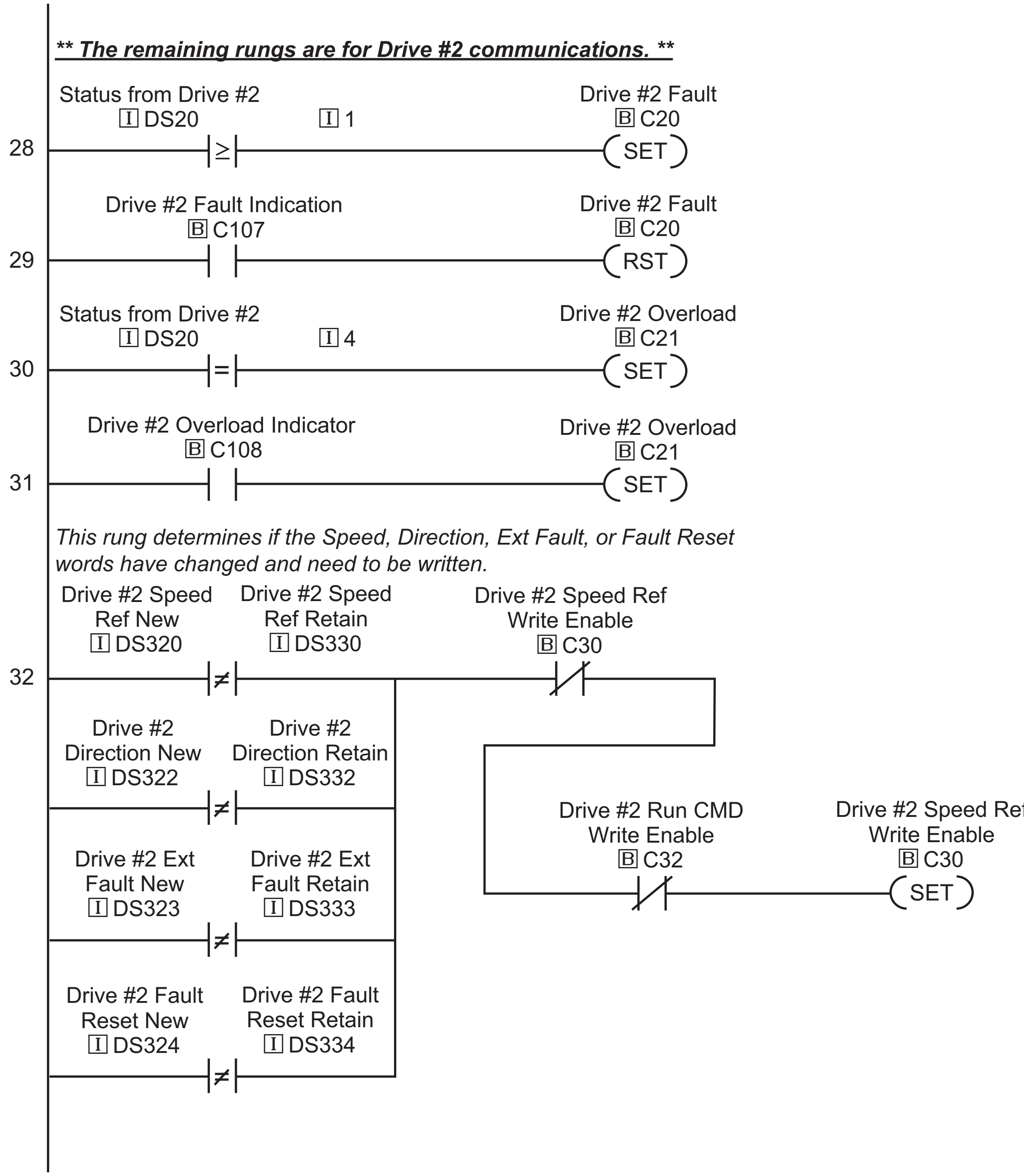
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(CONTINUED FROM PREVIOUS PAGE – CLICK PLC COMMUNICATION PROGRAM EXAMPLE)

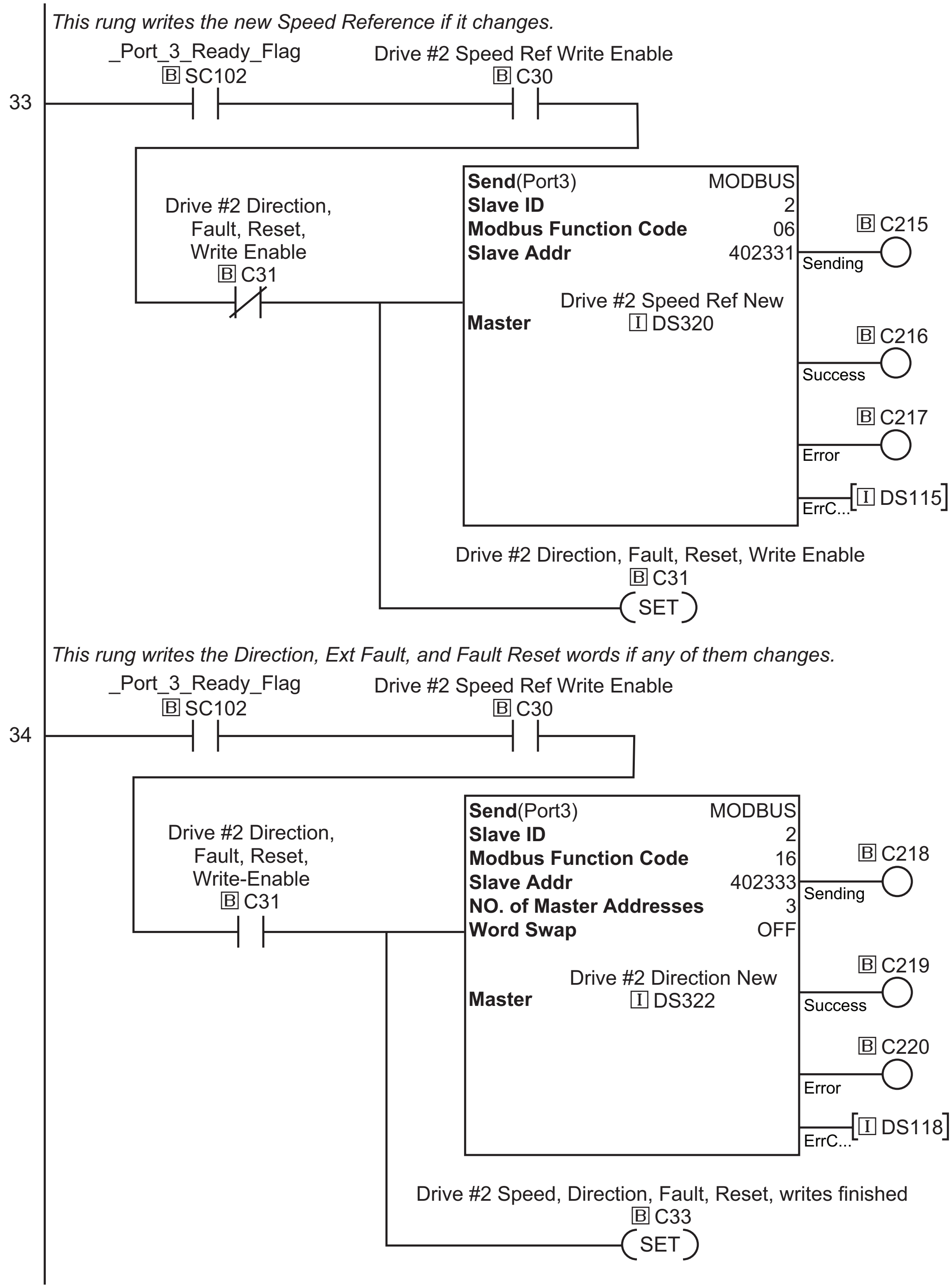
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(CONTINUED FROM PREVIOUS PAGE – CLICK PLC COMMUNICATION PROGRAM EXAMPLE)

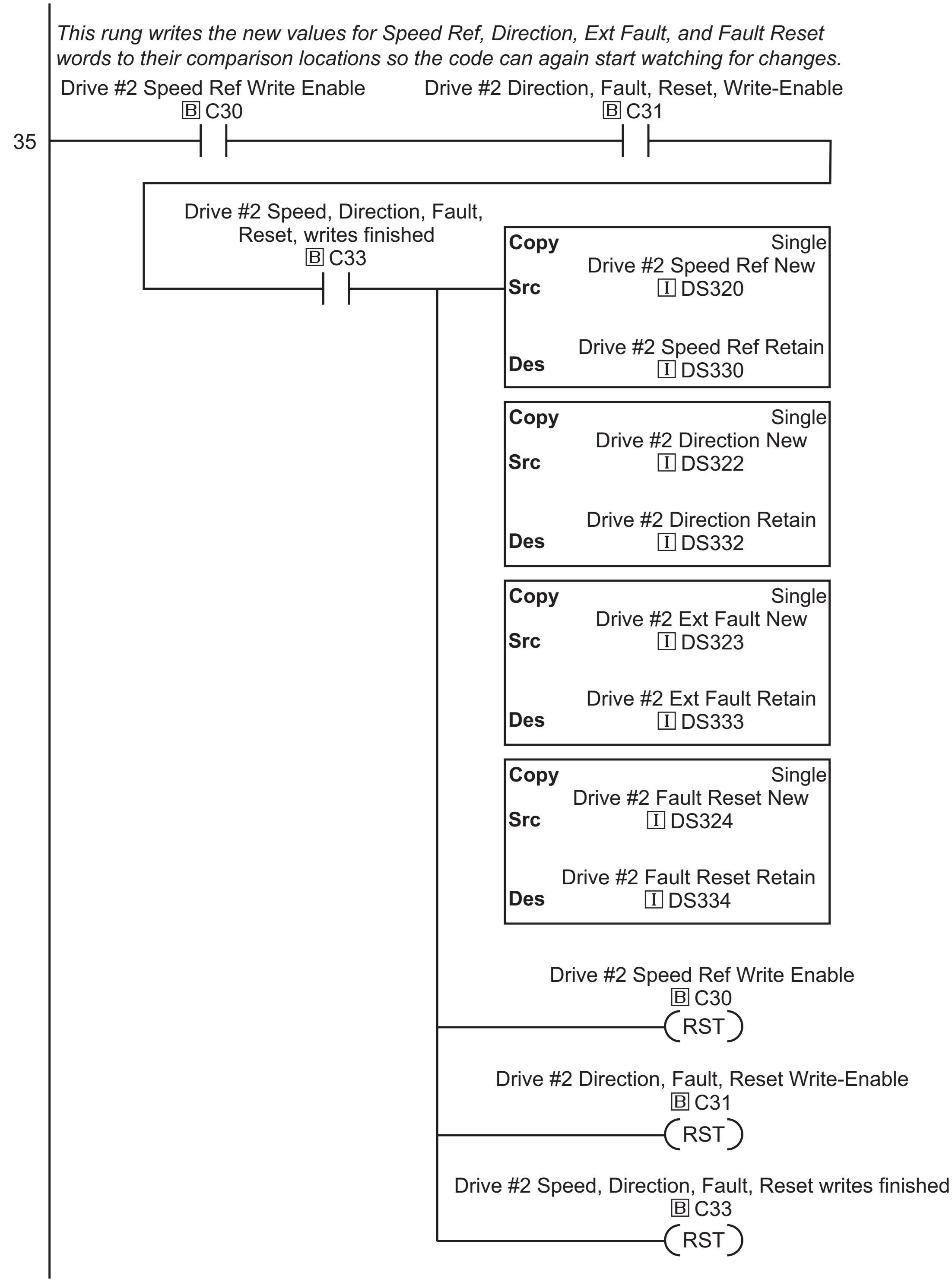
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

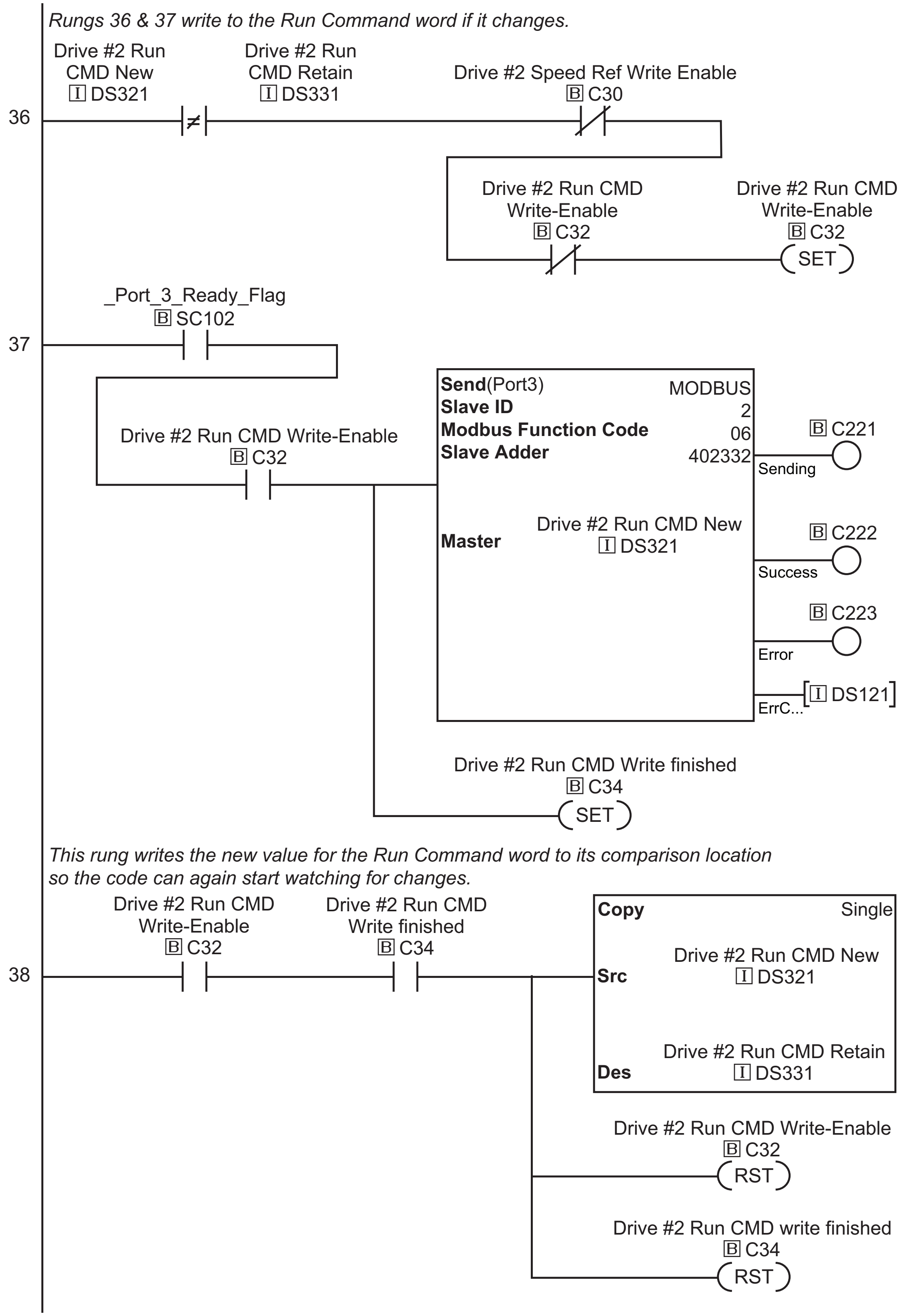
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

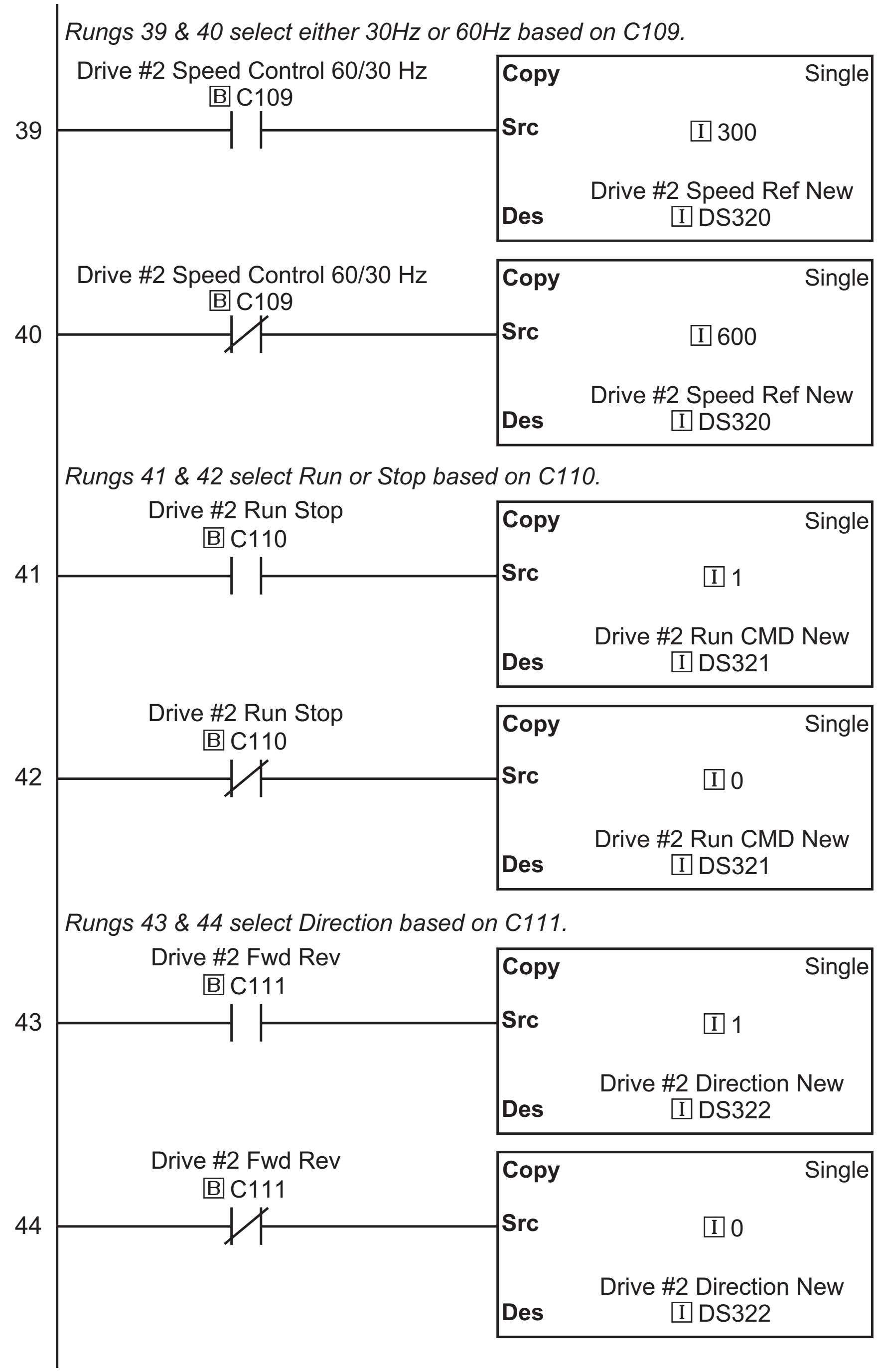
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

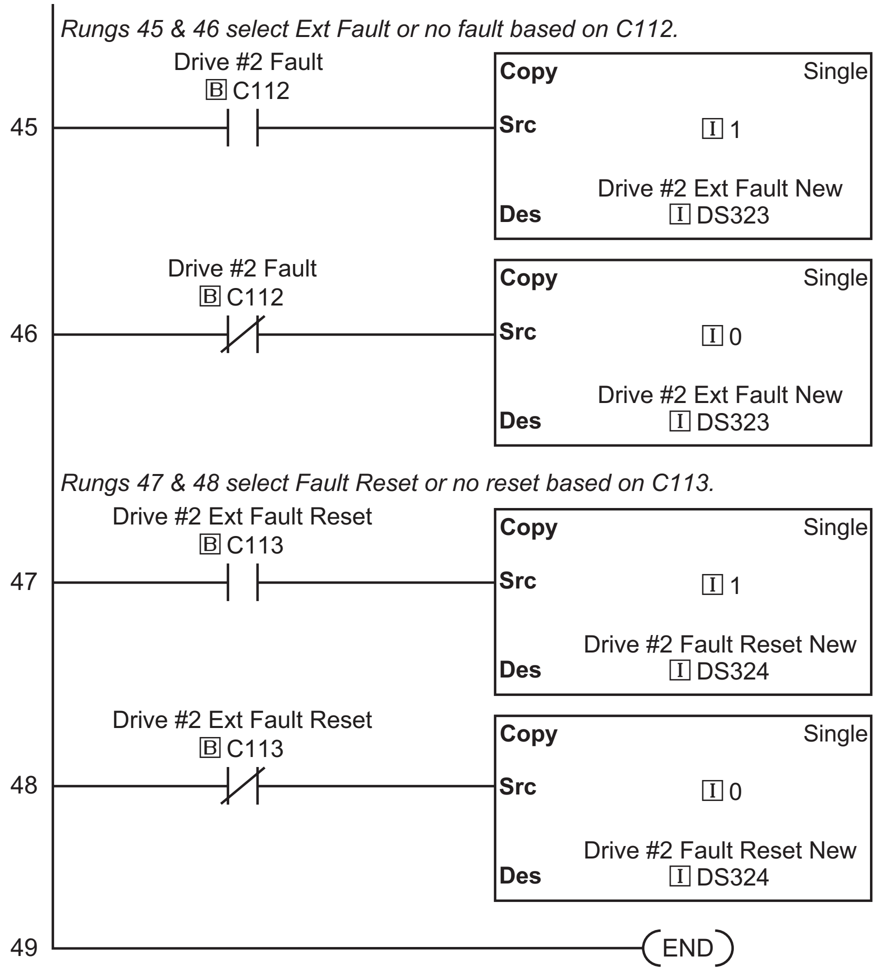
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – CLICK PLC communication program example)

(continued from previous page – CLICK PLC communication program example)

This program is for illustrational purposes only, and is not intended for a true application.



*Direct*LOGIC Modbus Ladder Programming

The set up for all of the *Direct*LOGIC CPUs is very similar. However, there may be some subtle differences between CPUs. Refer to the appropriate CPU User Manual for the specifics on your *Direct*LOGIC CPU.

The following ladder program shows some examples of how to control the GS1 AC drive through Modbus RTU. The drive should be setup and tested for communications before it is connected to a load.

Warning: A drive should never be connected to a load until any applicable communication programs have been proven.

Warning: Write programs in such a way that the program does not erroneously overwrite a remote Stop command with a Run command, such as when P3.00 is set to 03. This example program prevents such an accidental overwrite.

These programs are for illustrational purposes only, and are not intended for a true application.

Separate Run Command Write Instruction

Why do we write the Run Command with a separate write instruction? If we write the Run Command to the drive along with the Speed Reference, Direction, External Fault, and Fault Reset Commands, we can keep the parameter addresses in sequence, and we can update all five of the commands with one write instruction. This method is valid only if we disable the drive's keypad STOP button (P3.00 = 04).

Typically, the keypad STOP button will be enabled (P3.00 = 03), and we need to prevent a change in one of the other commands from overriding a keypad Stop Command by causing a previous Run Command to be rewritten to the drive. By using a separate Run Command write instruction, only a deliberate Run Command change by the program will run the drive again after a stop.

Block Transfer Parameters for Modbus Programs

For writing to any of the parameters from P0.00 to P8.01, a group of 10 block transfer parameters (P9.11 to P9.20) is available in the GS1 AC drive. This sequential block of parameters can be used to "group" various miscellaneous non-sequential parameters, so that you can update the parameters in one programming write block instead of having to use multiple write commands.

For example, it would typically take three different write commands to change the three non-sequential parameters Accel Time 1 (P1.01), Accel S-curve (P1.03), and Multi-speed 1 (P5.01).

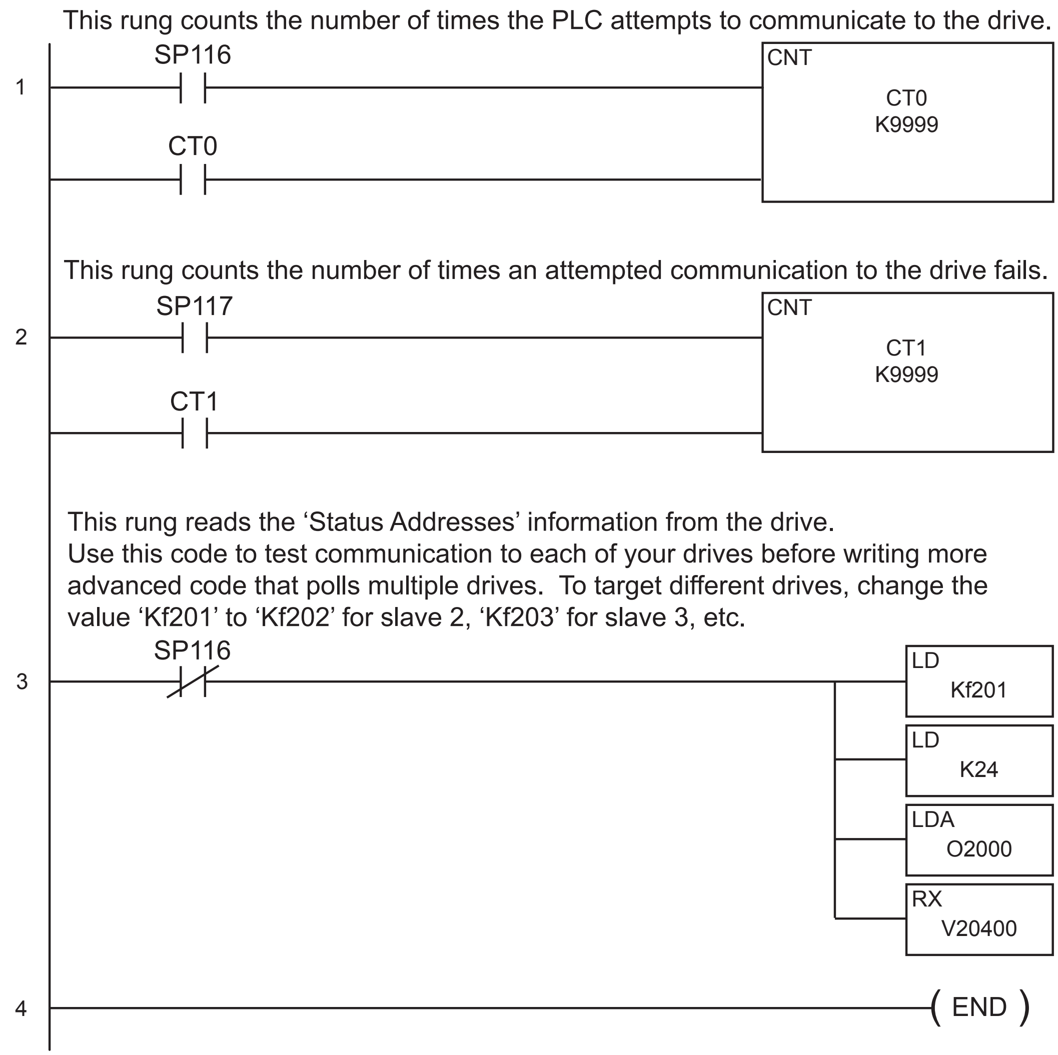
However, you could make the same three changes using one write command by setting P9.11 to P1.01, P9.12 to P1.03, and P9.13 to P5.01, so that the parameters become sequential.

DirectLOGIC Basic Communication Program – start with this code

We recommend starting with the following program code, and using it to test communication to each of your drives before adding more advanced code for your application.

To target different drives, change the value Kf201 to Kf202 for slave 2, Kf203 for slave 3, etc.

This program is for illustrational purposes only, and is not intended for a true application.



SP116 is a special relay in the DirectLOGIC CPUs that monitors the PLC’s communications. SP116 is on when Port 2 is communicating with another device.

SP117 is a special relay in the DirectLOGIC CPUs that monitors the PLC’s communications. SP117 is on when Port 2 has encountered a communication error.

Programming Differences for DirectLOGIC PLCs

Different types of DirectLOGIC PLCs can be programmed differently, depending upon the types of network read and write instructions they can perform. There are two different types of these instructions, and this User Manual shows programming examples of both types.

RX/WX Instructions for DL05, D2-250(-1), D4-450

PLCs with DL05, D2-250, D2-250-1, and D4-450 CPUs can read from and write to networks using RX (Read from Network) and WX (Write to Network) programming instructions.

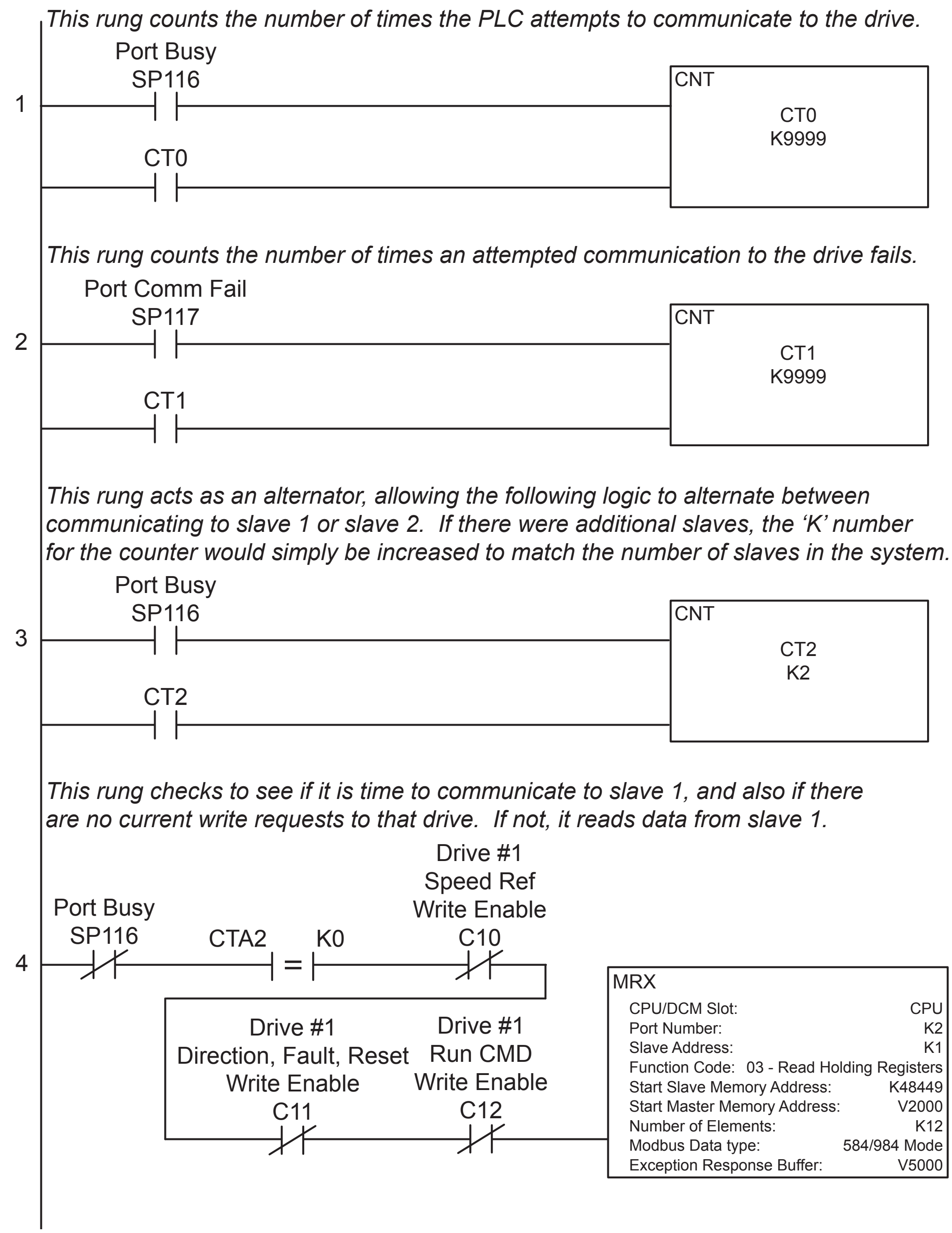
MRX/MWX Instructions for DL06, D2-260

In addition to the RX and WX instructions listed above, PLCs with DL06 and D2-260 CPUs can also read from and write to networks using MRX (Modbus Read from Network) and MWX (Modbus Write to Network) programming instructions.

The MRX and MWX instructions are simpler and easier to use than are the RX and WX instructions. Therefore, we recommend that you use DL06 or D2-260 with MRX and MWX instructions if you have a choice.

DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs

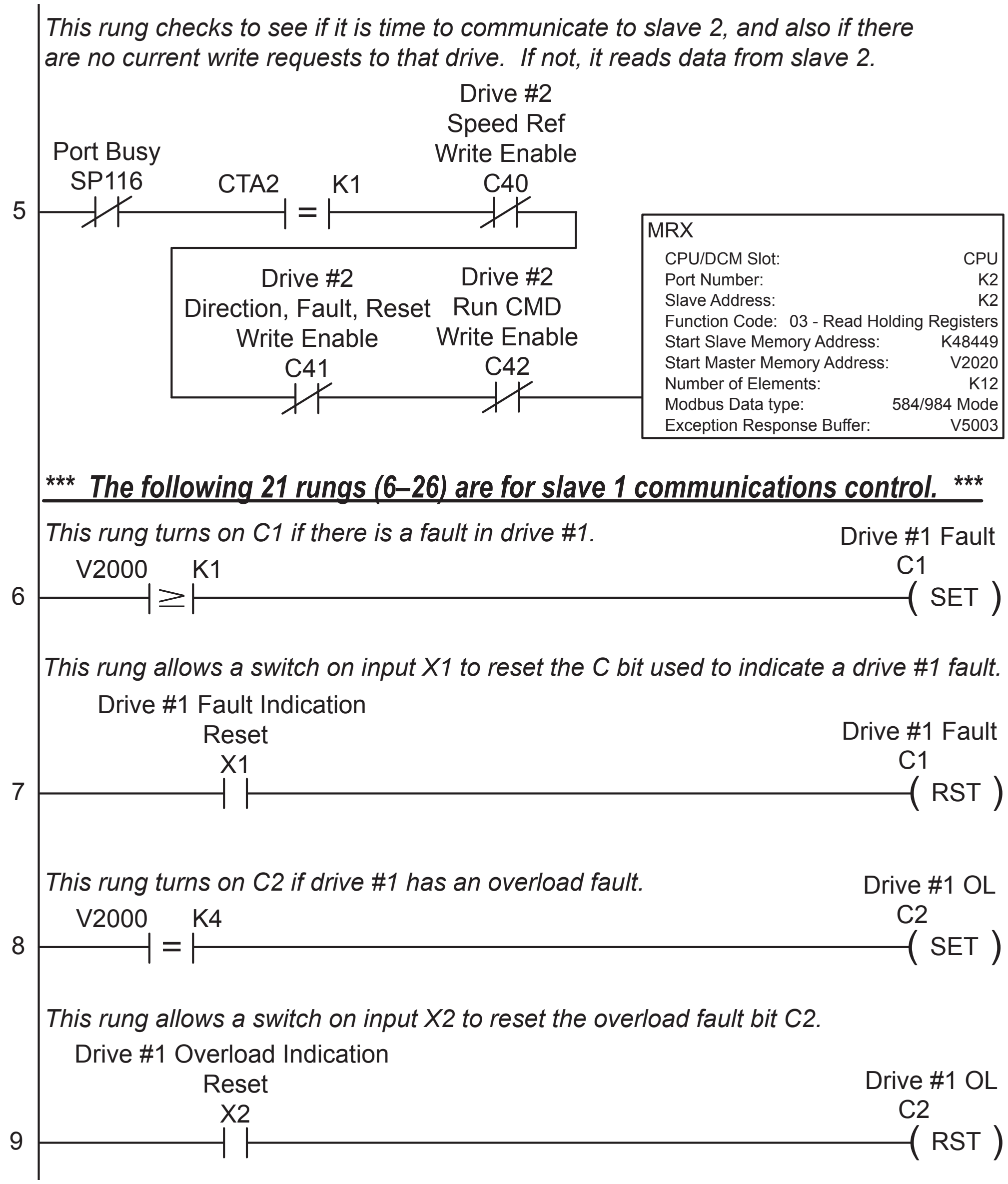
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

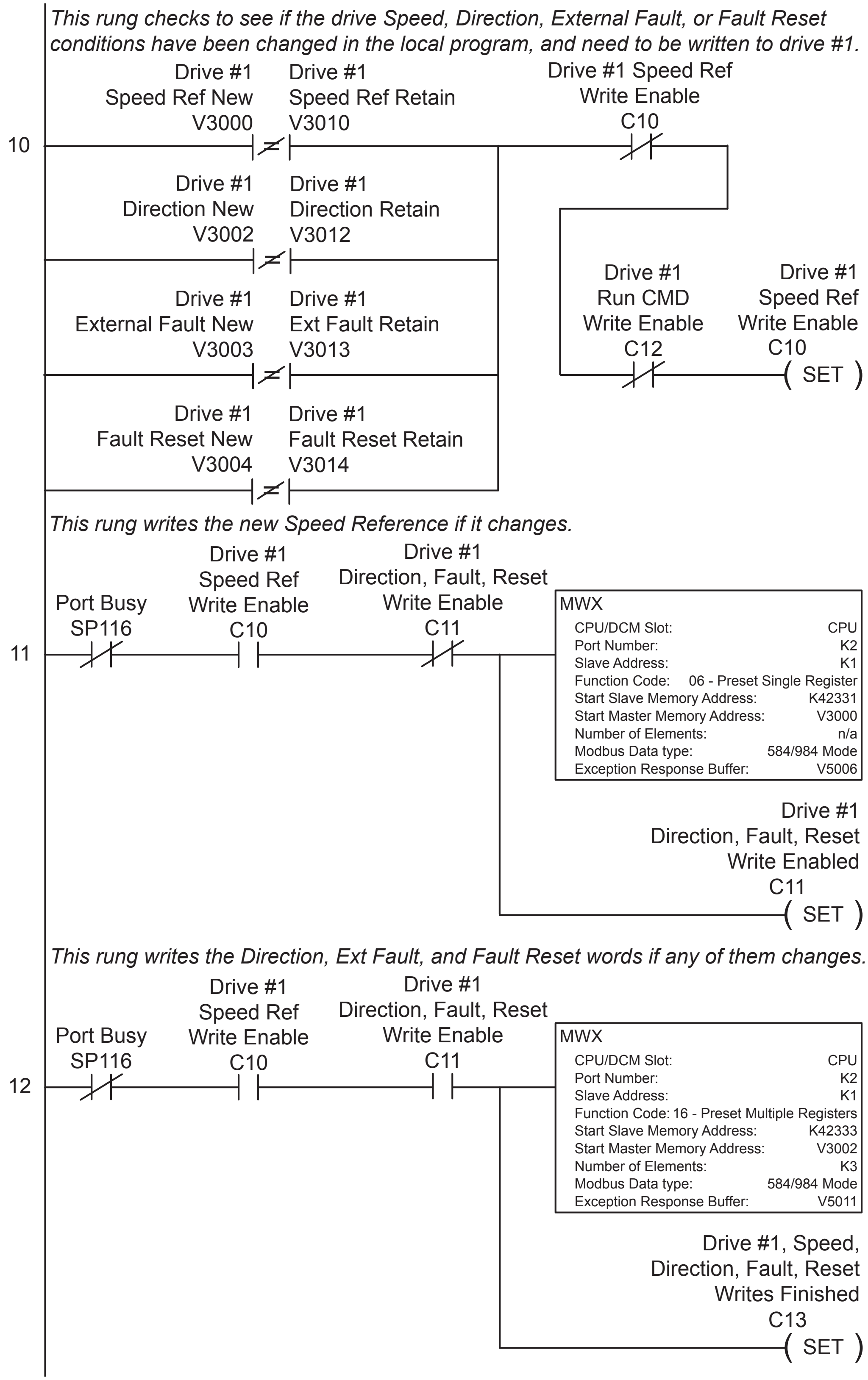
This program is for illustrative purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(Continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

This program is for illustrational purposes only, and is not intended for a true application.

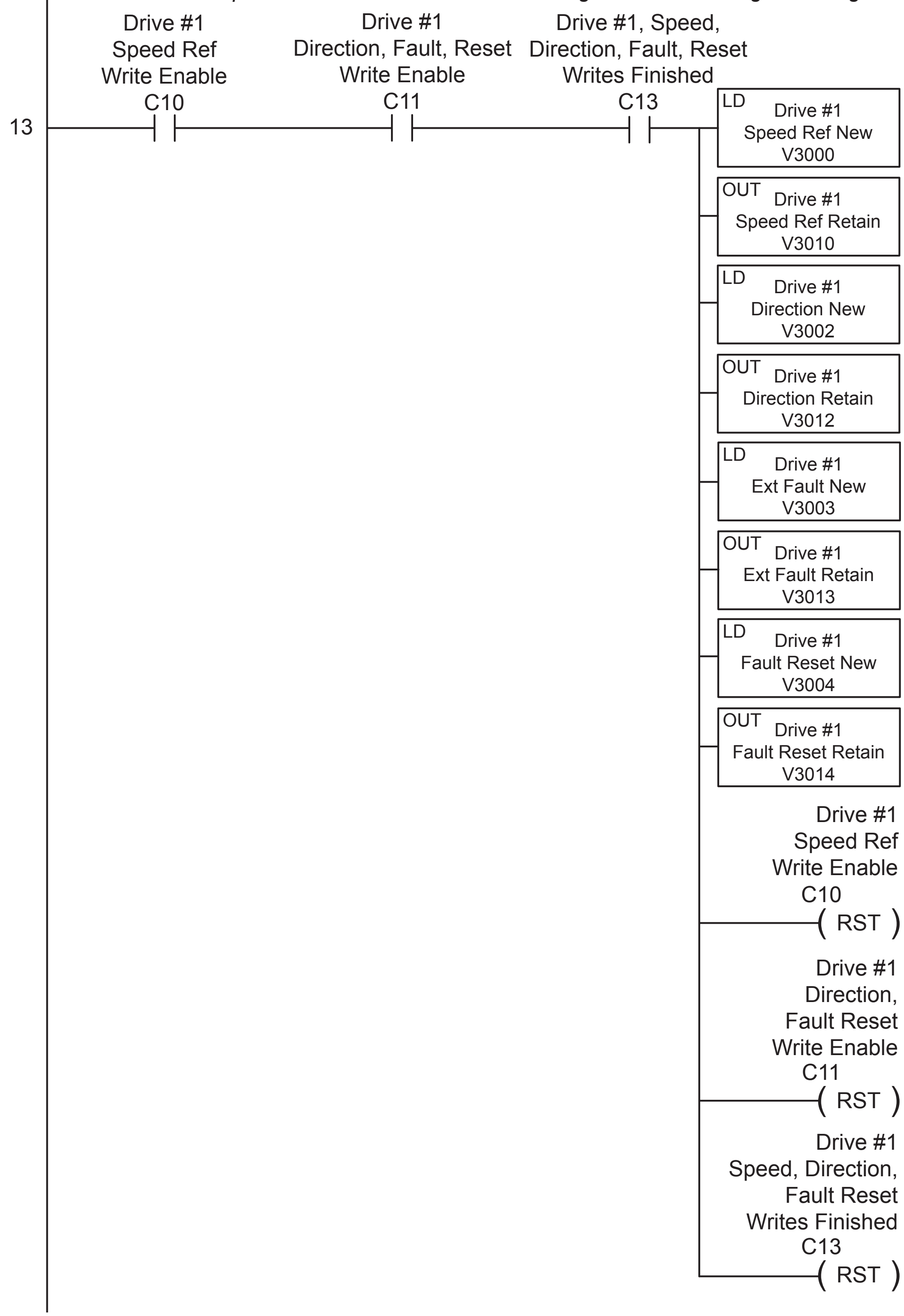


(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

This program is for illustrational purposes only, and is not intended for a true application.

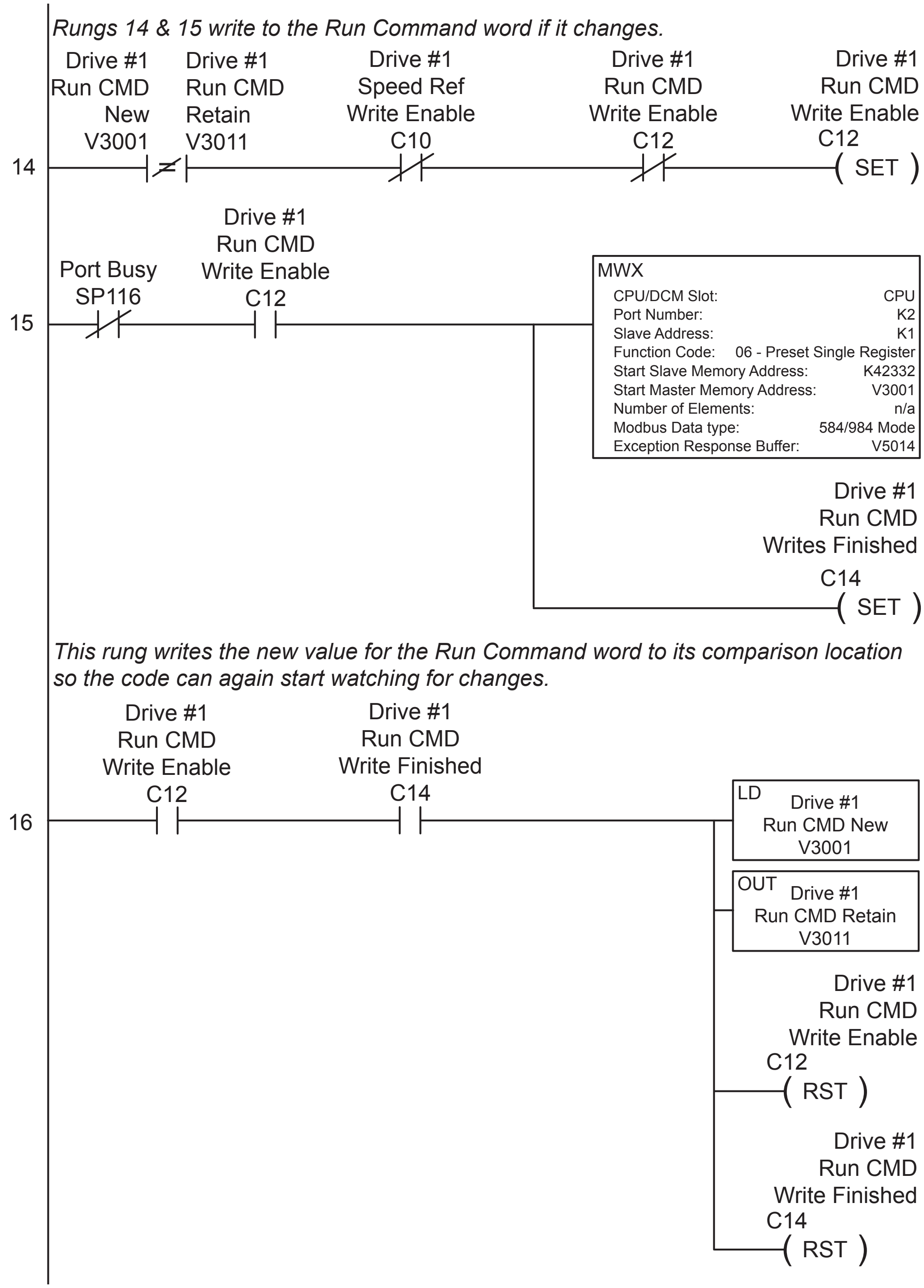
This rung writes the new values for Speed Ref, Direction, Ext Fault, and Fault Reset words to their comparison locations so the code can again start watching for changes.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

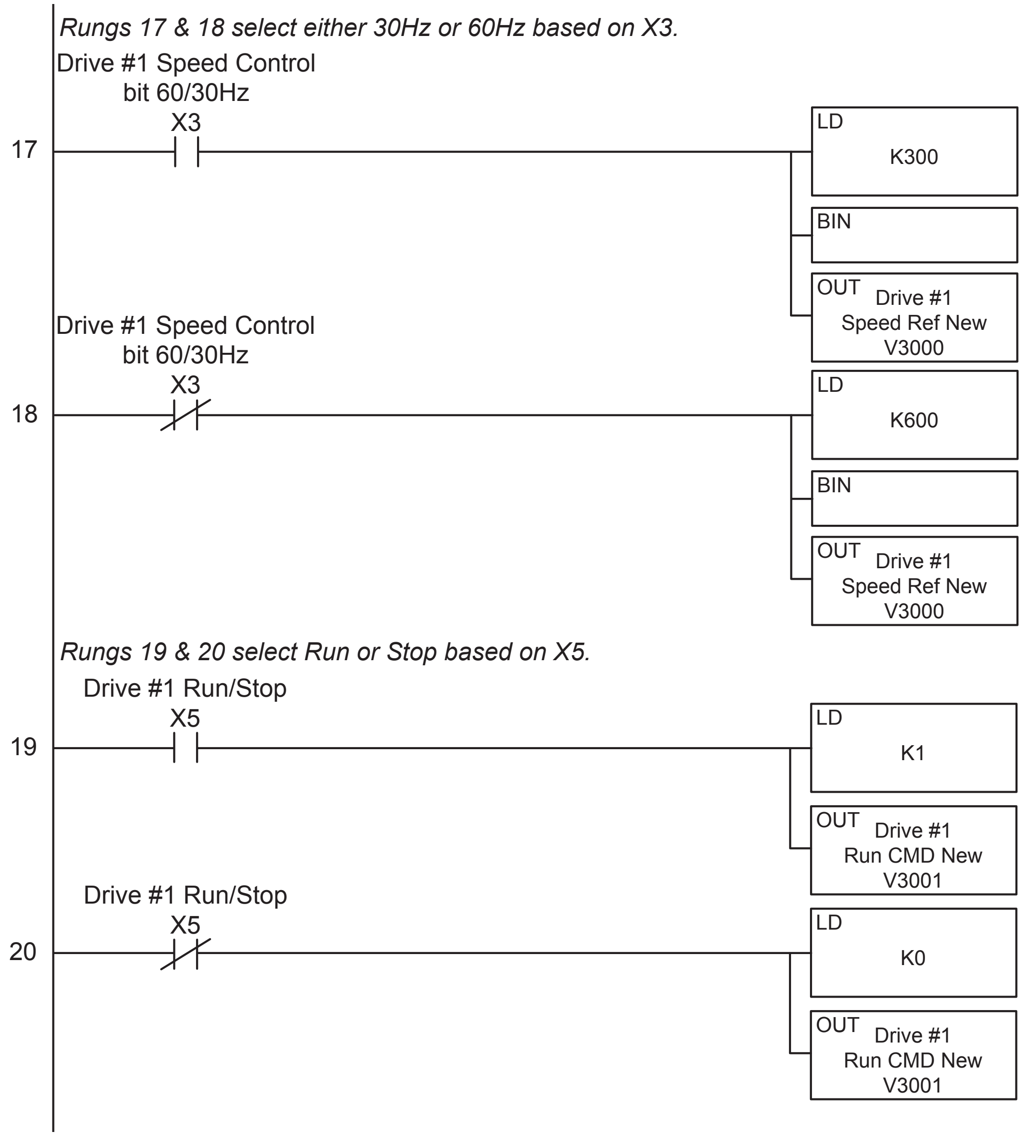
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

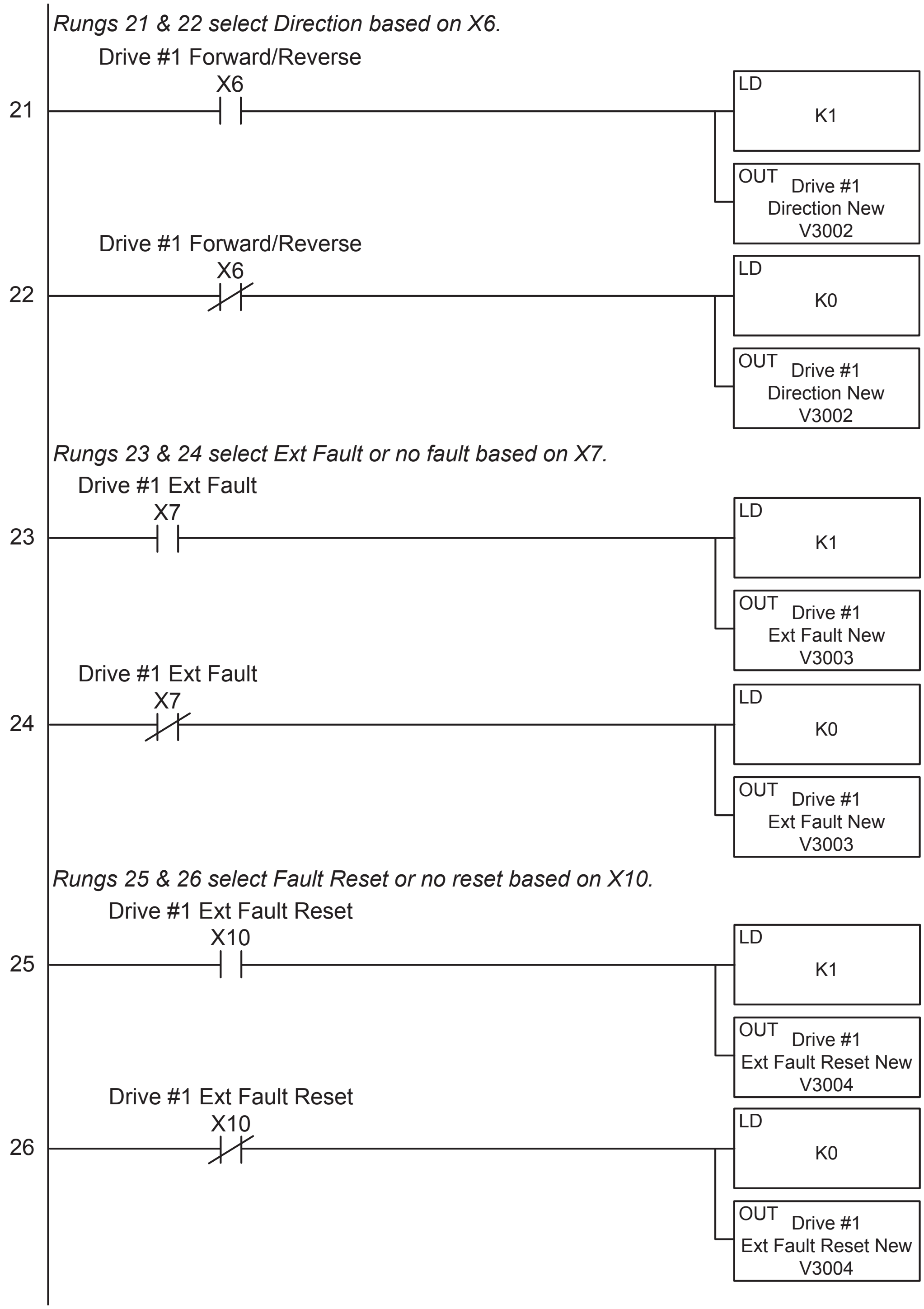
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

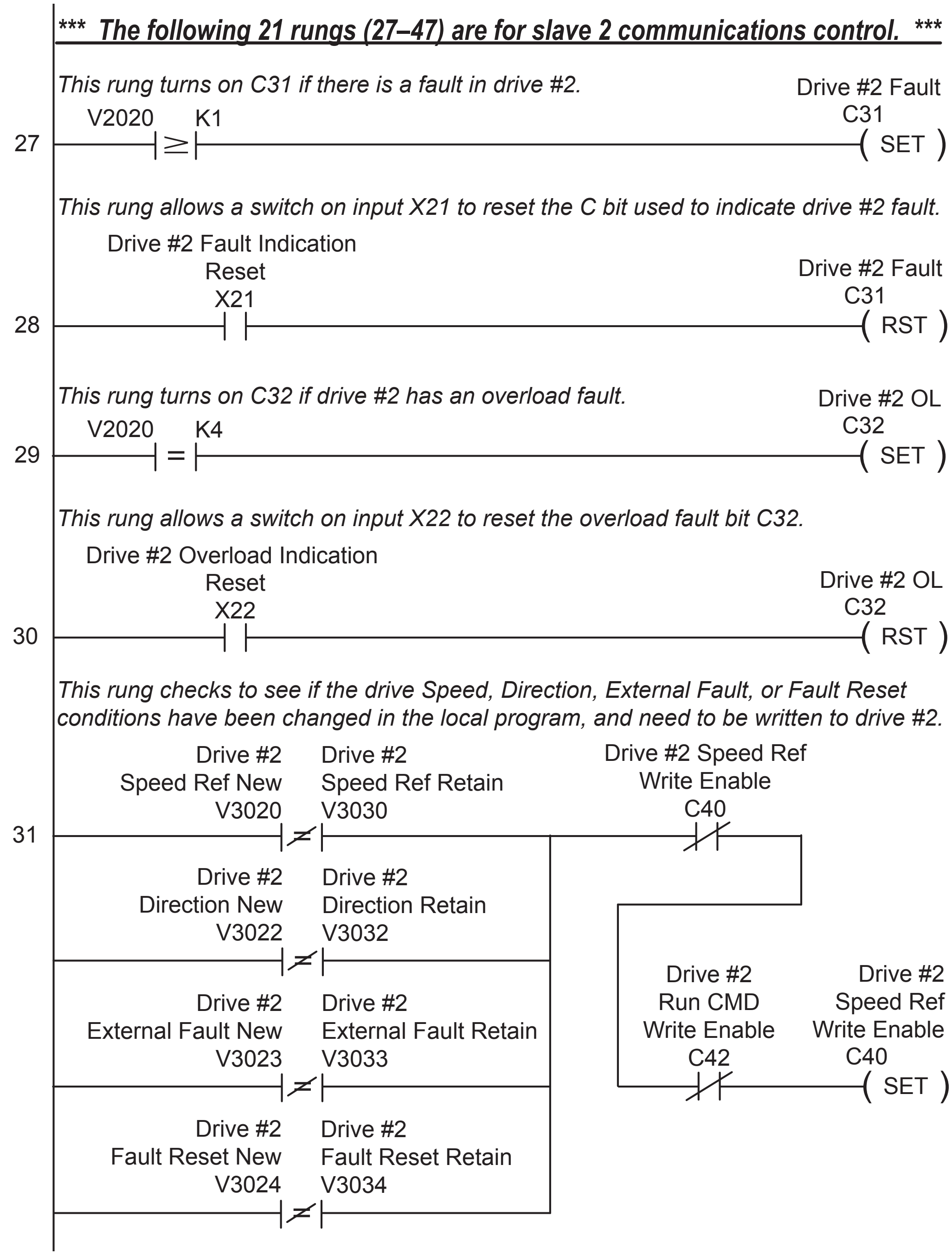
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(CONTINUED FROM PREVIOUS PAGE – DL MRX/MWX COMMUNICATION PROGRAM – FOR DL06 & D2-260 PLCS)

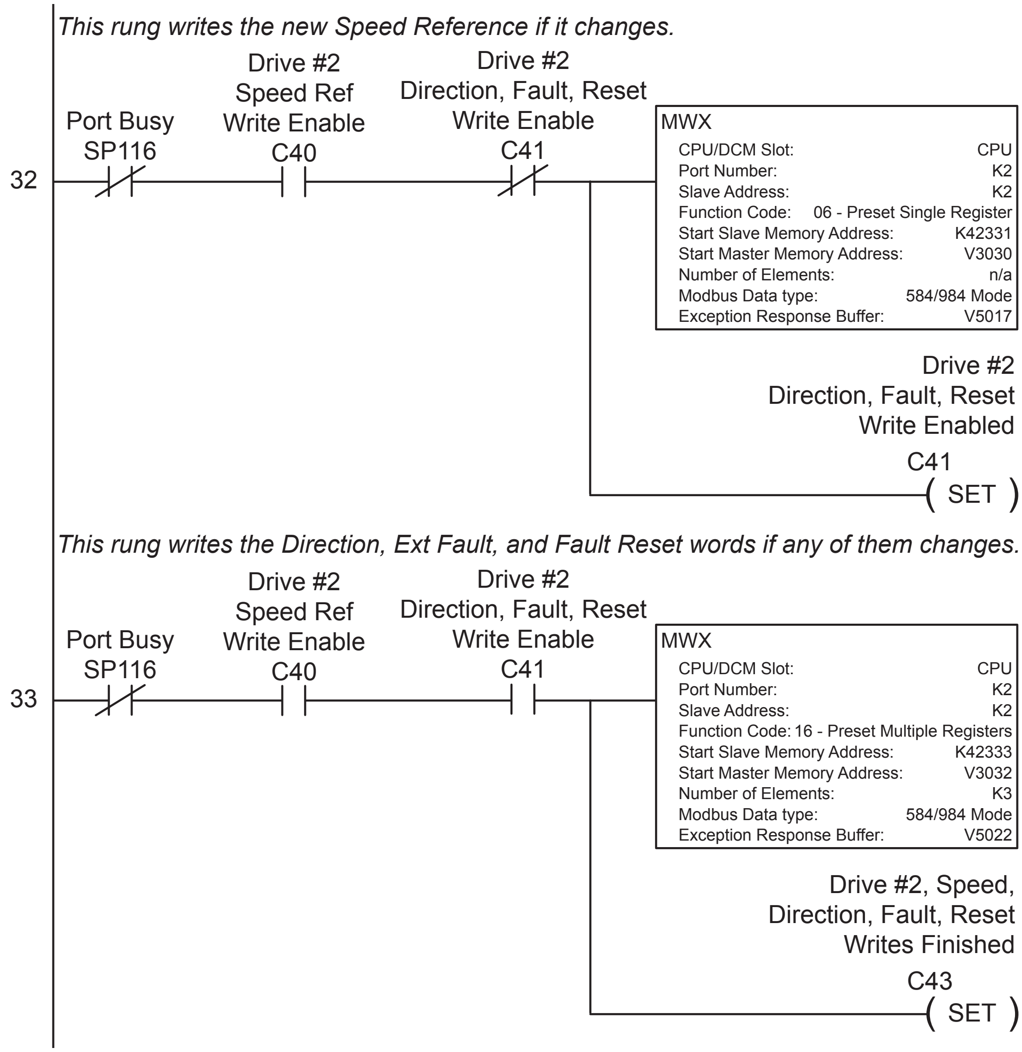
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

*(continued from previous page – **DL MRX/MWX Communication Program** – for **DL06 & D2-260 PLCs**)*

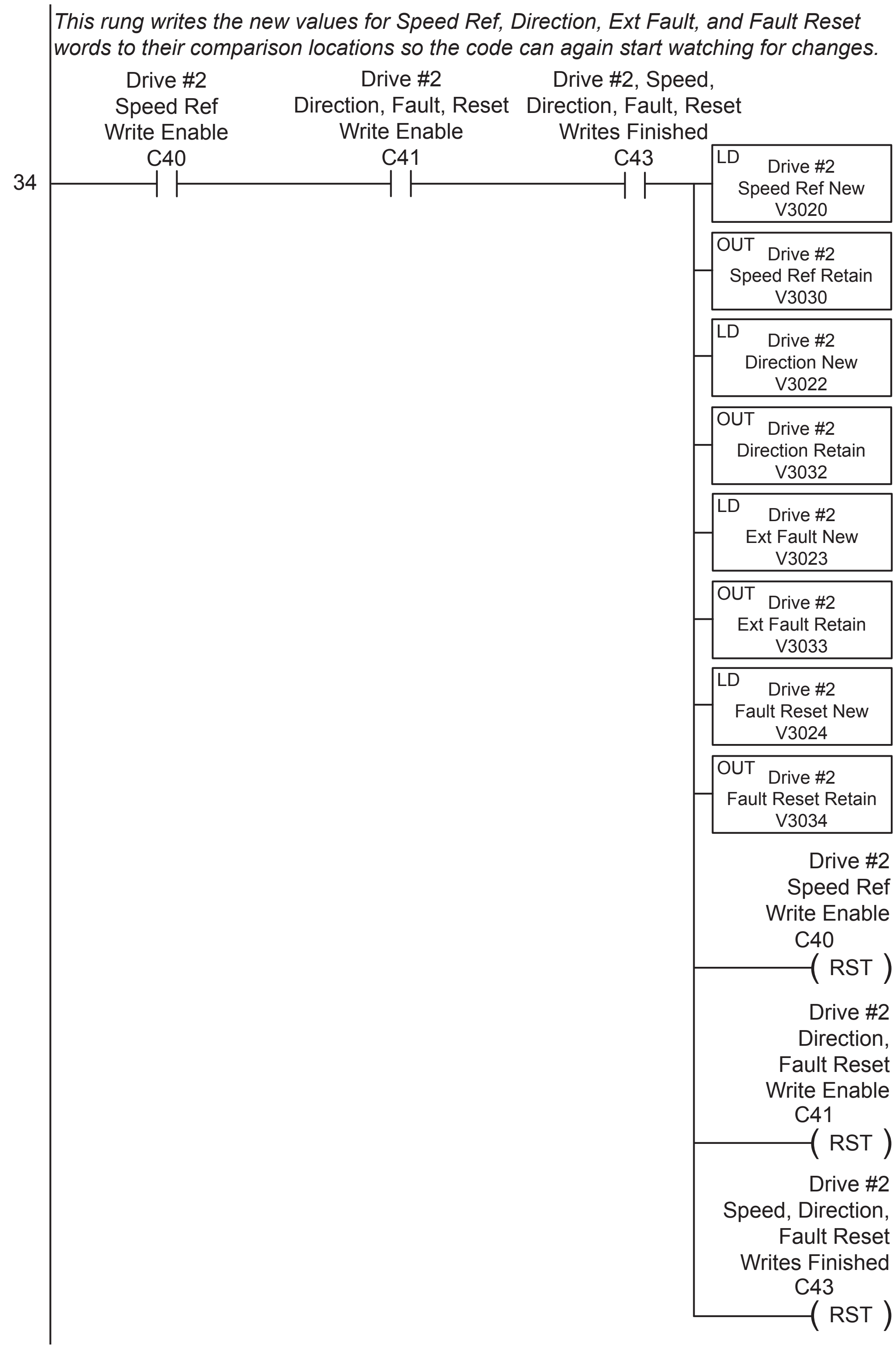
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

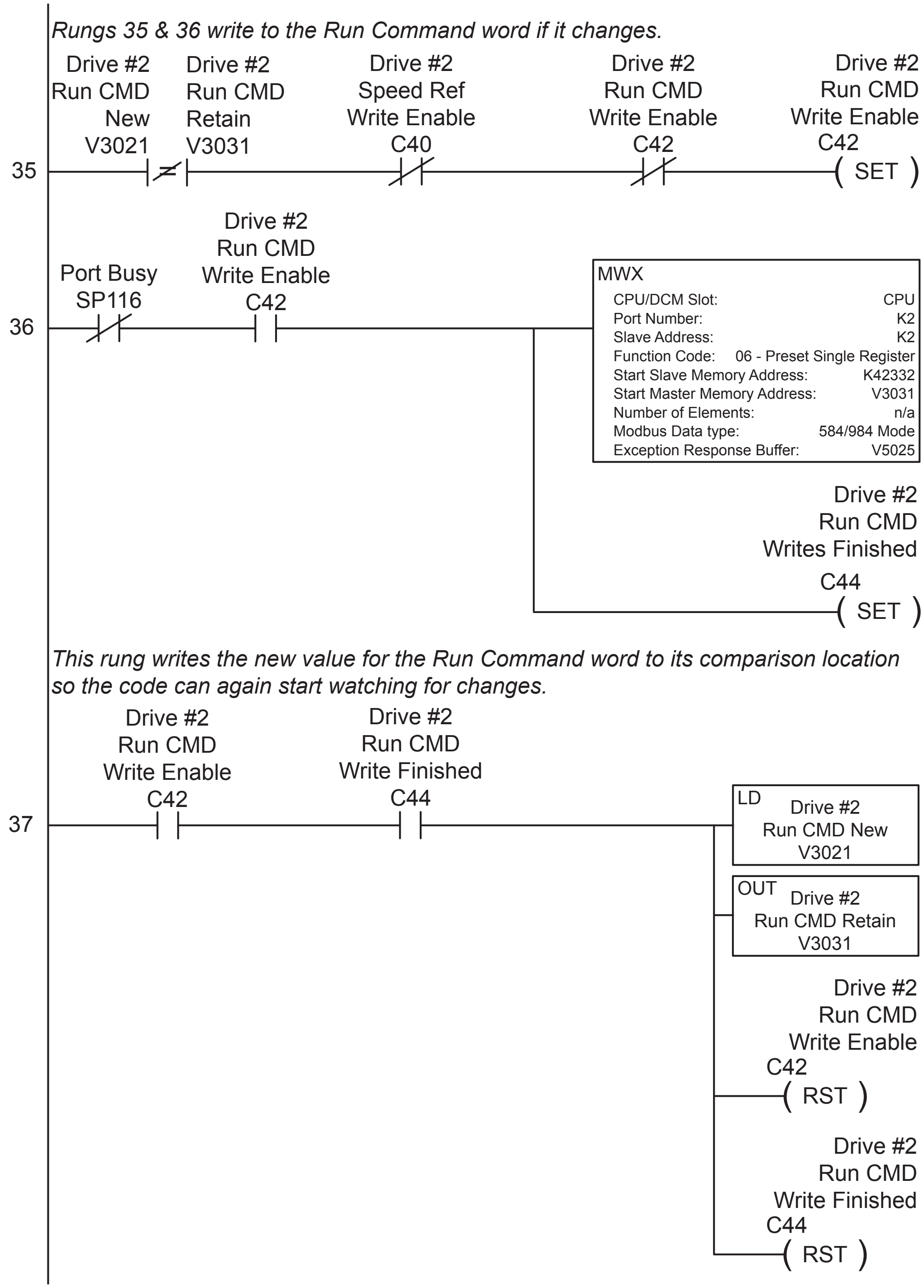
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

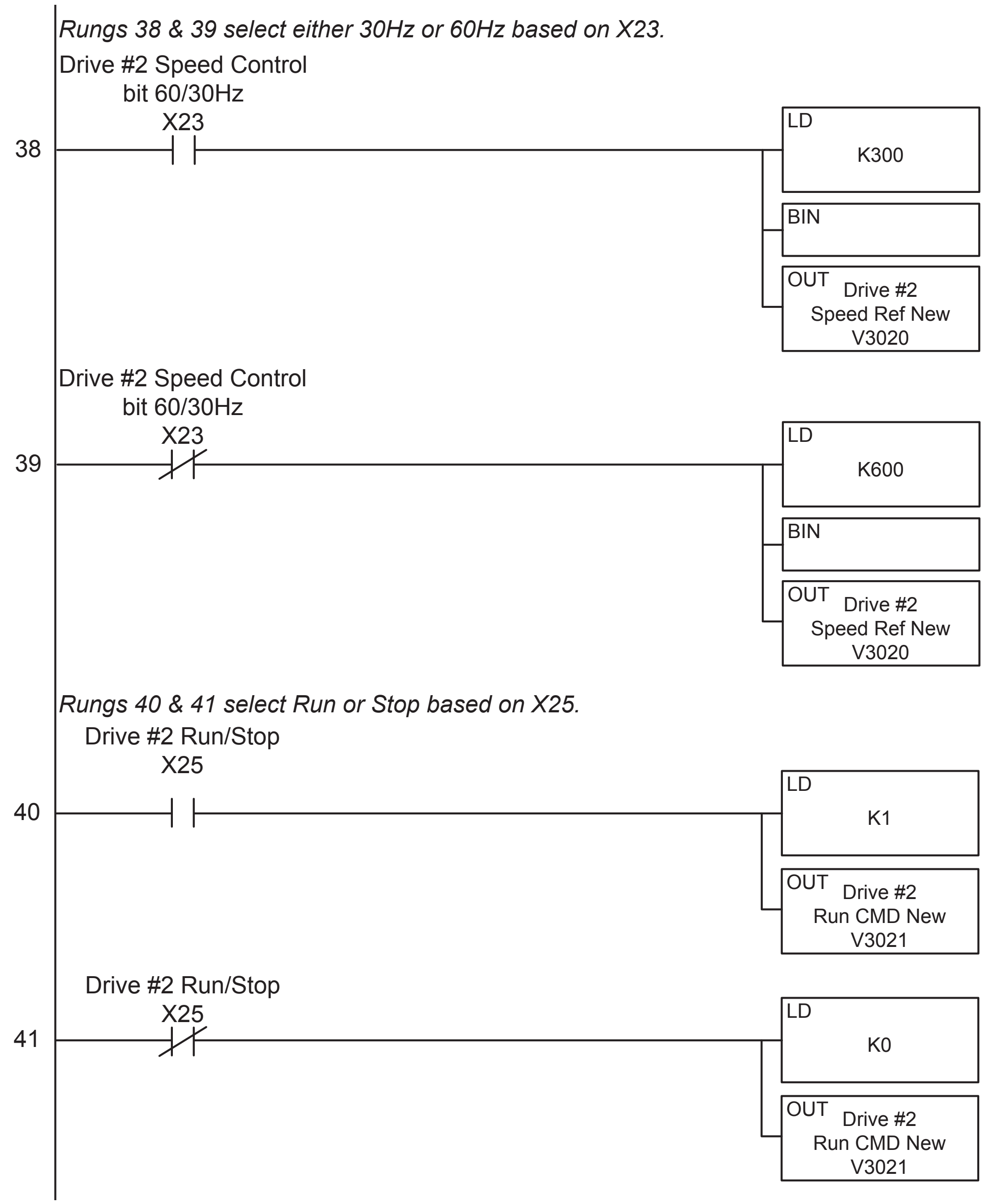
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

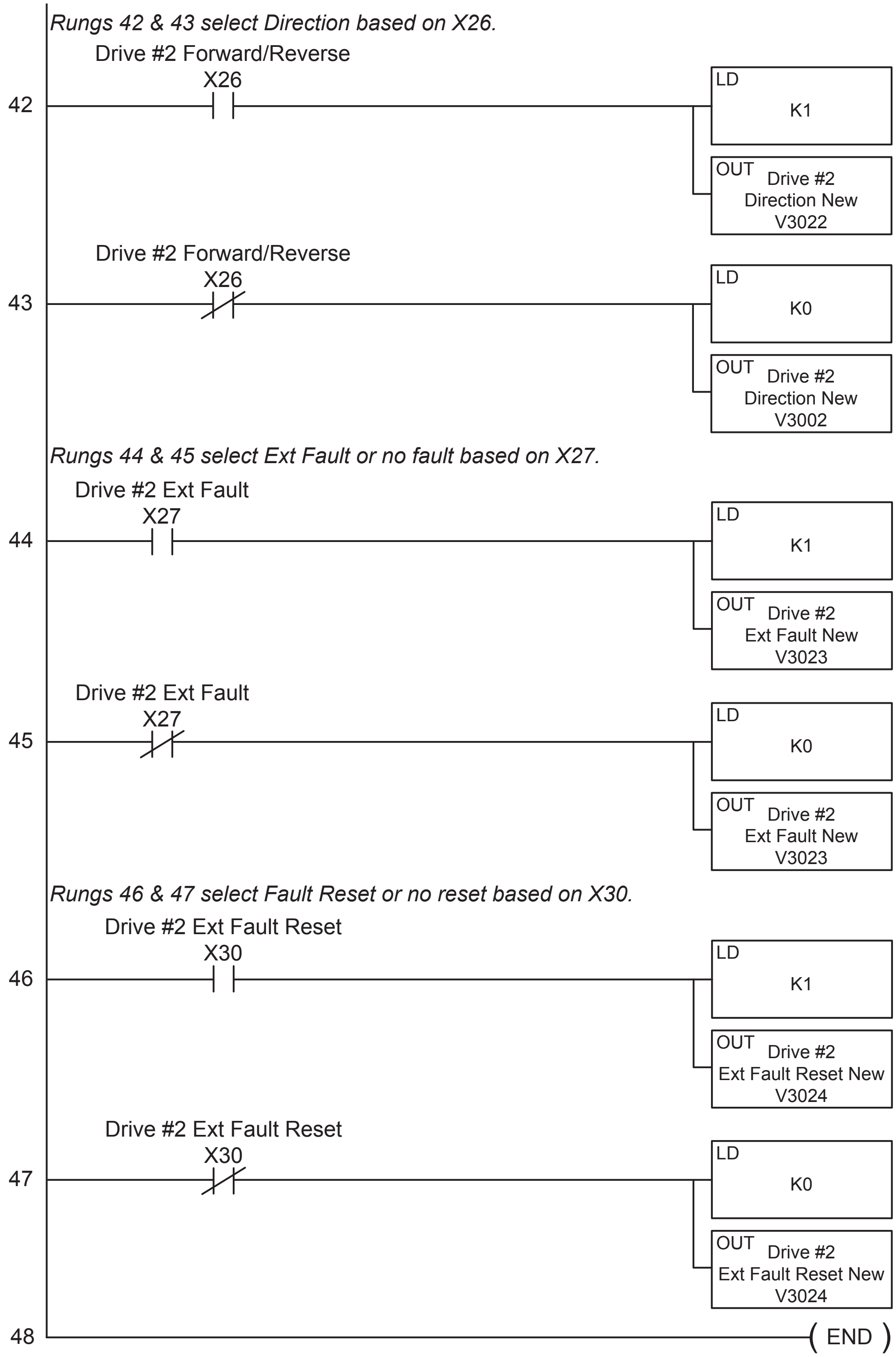
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)

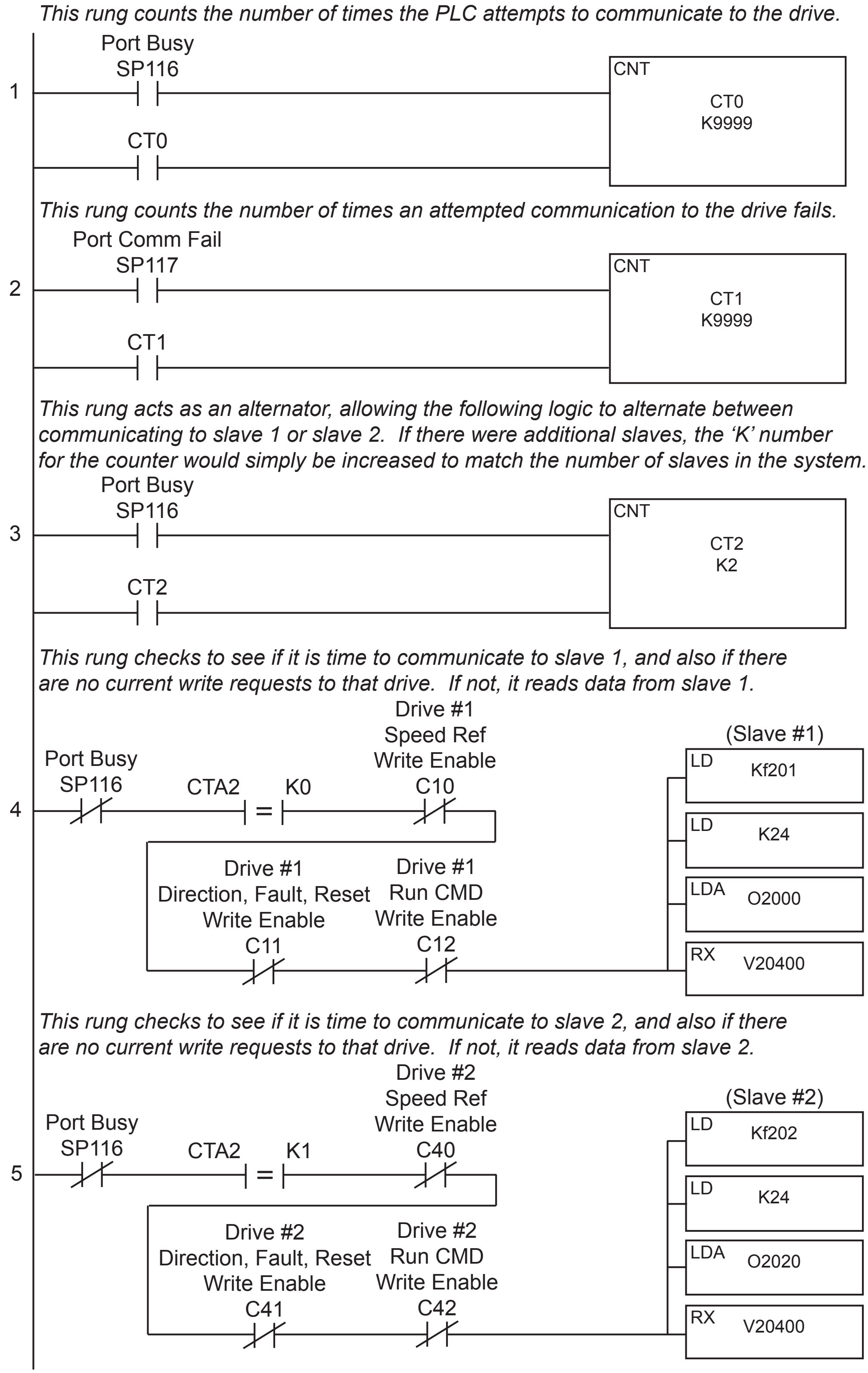
(continued from previous page – DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs)

This program is for illustrational purposes only, and is not intended for a true application.



DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450 PLCs

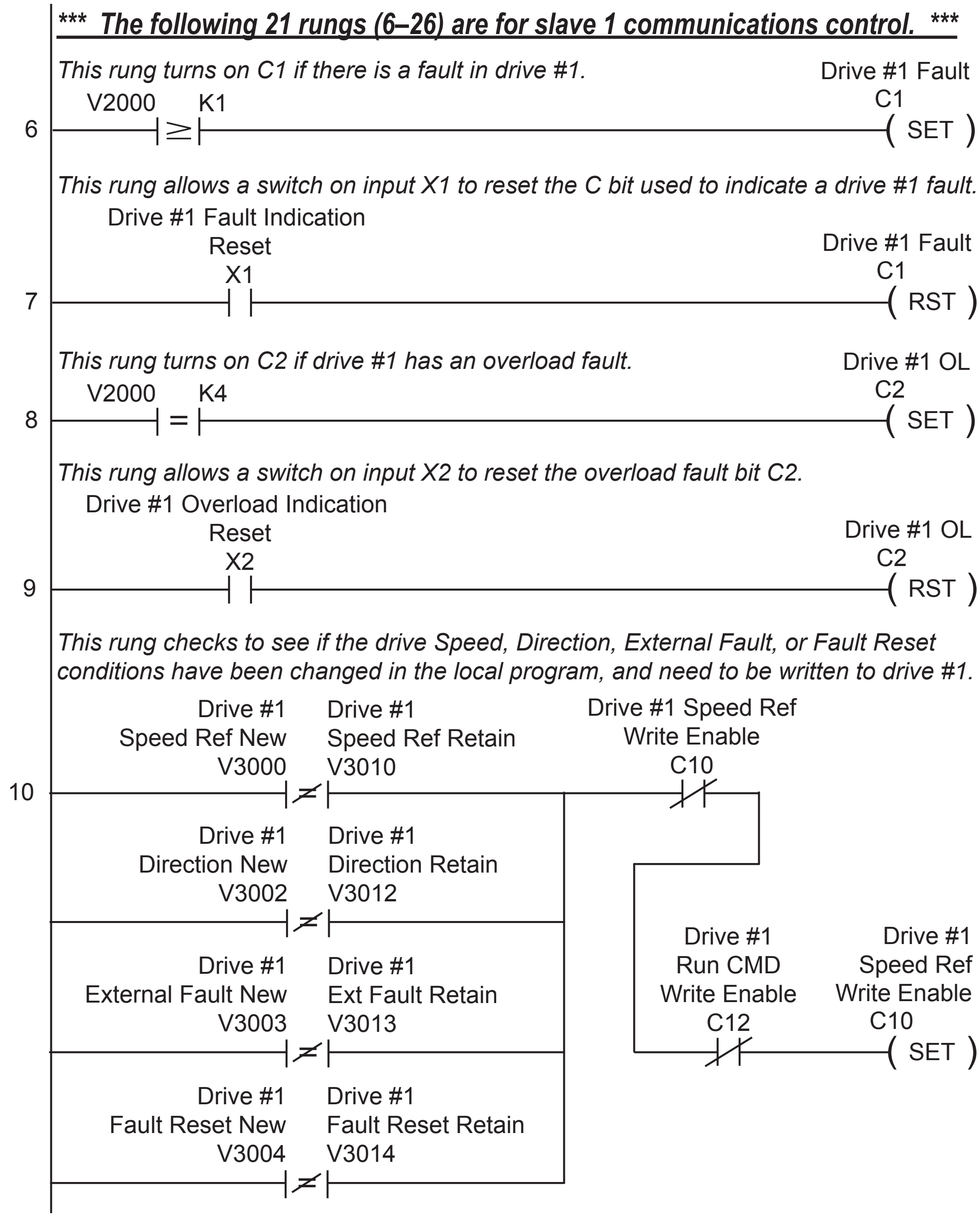
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

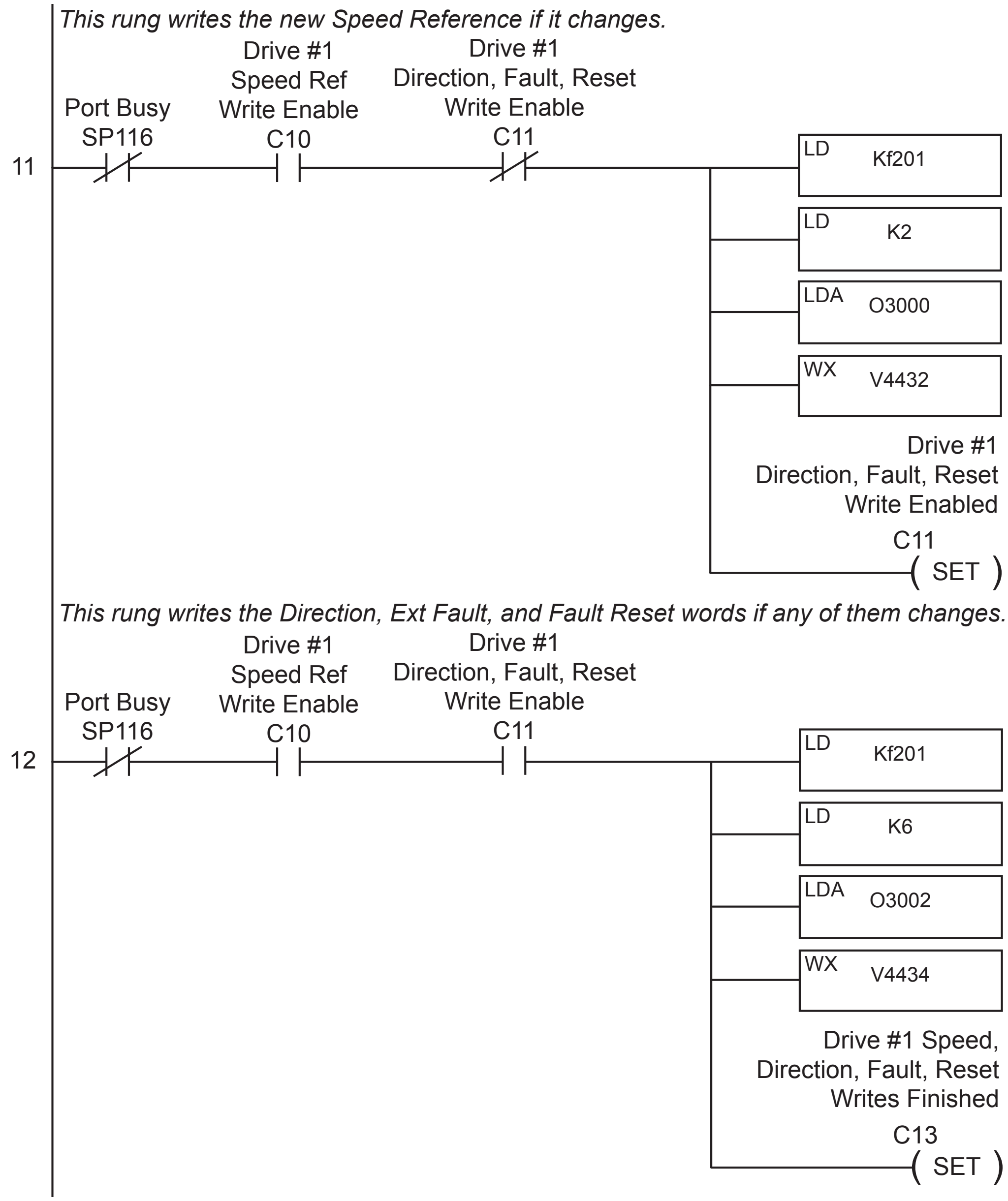
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

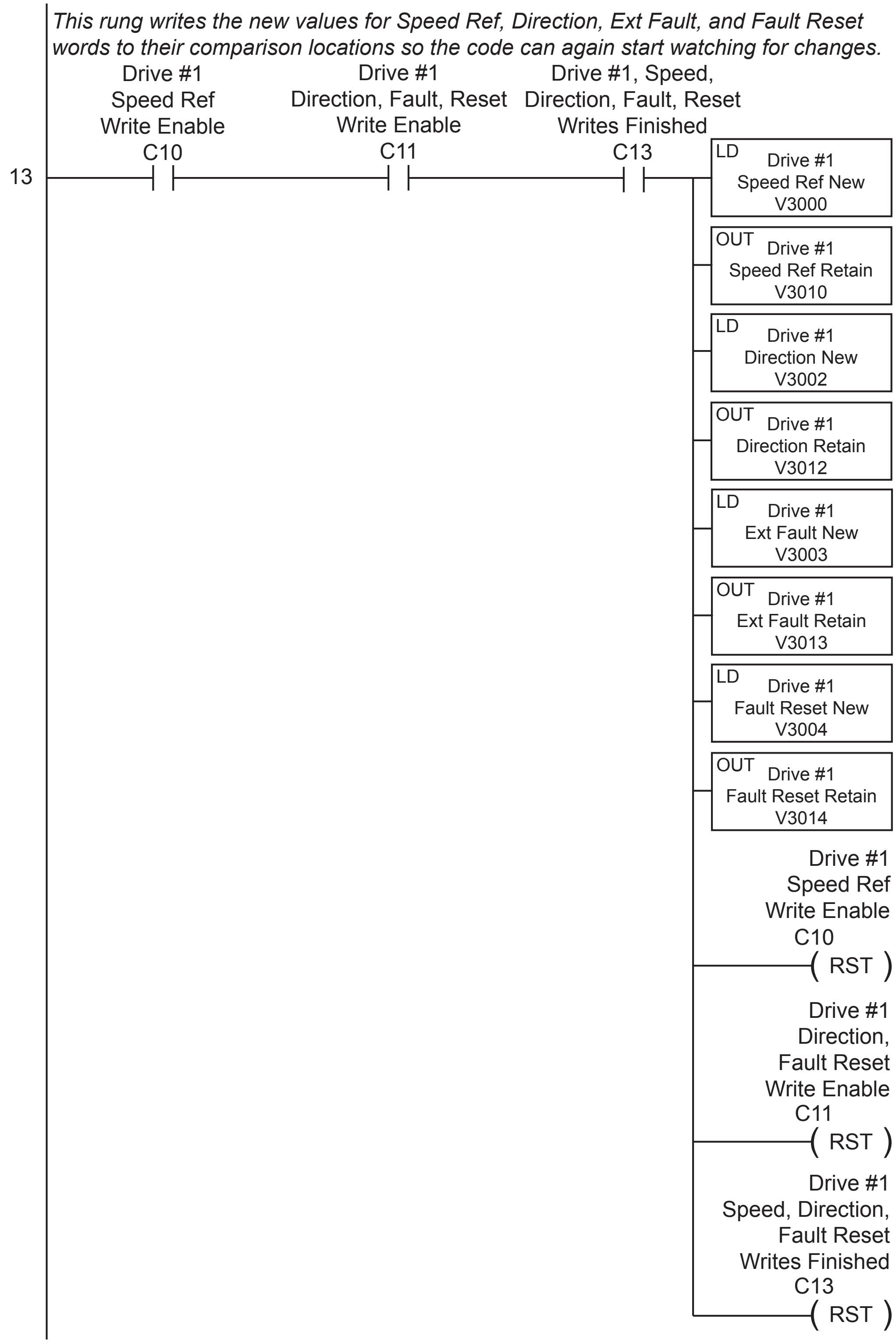
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(CONTINUED FROM PREVIOUS PAGE – DL RX/WX COMMUNICATION PROGRAM – FOR DL05, D2-250(-1), D4-450)

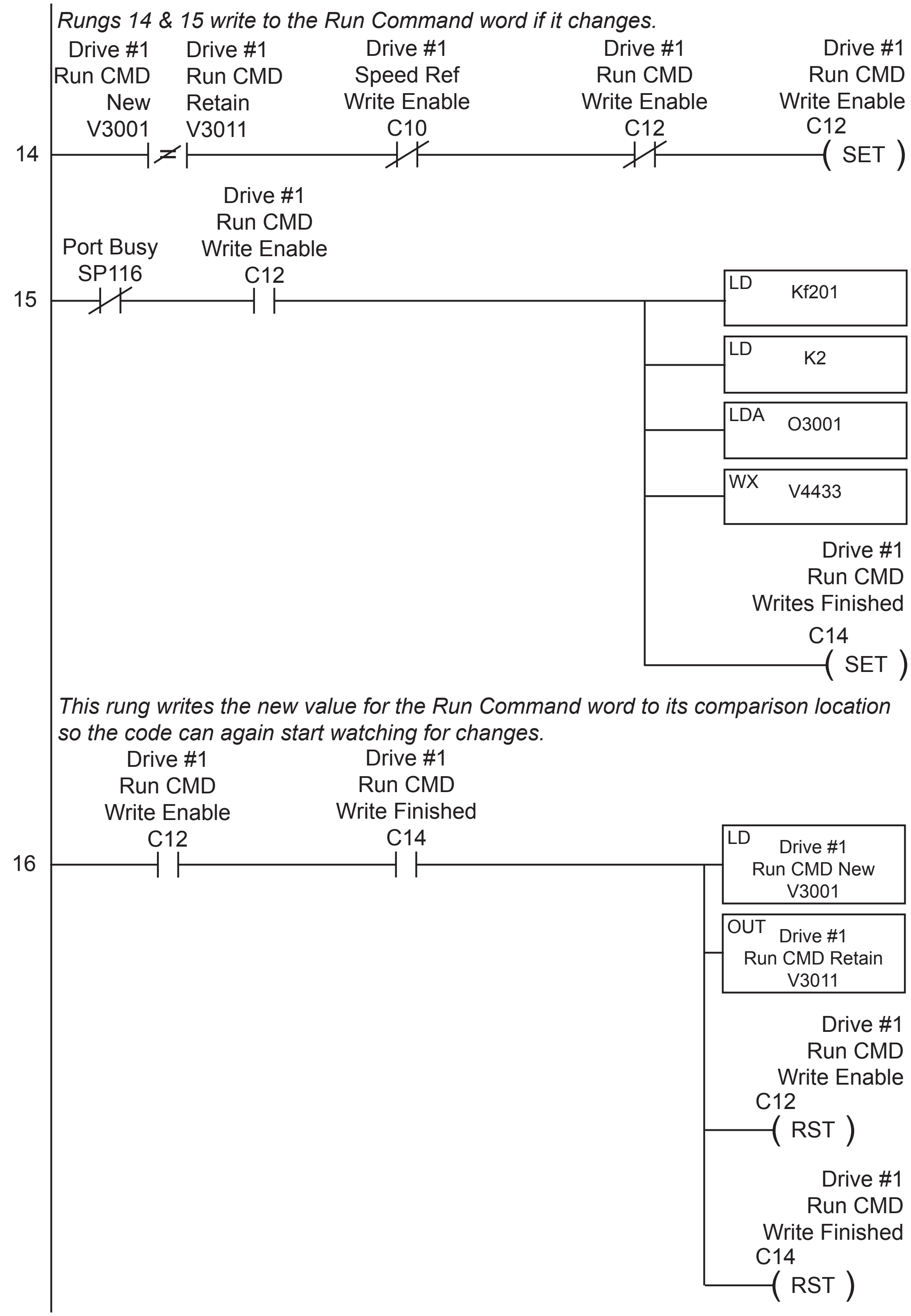
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

*(CONTINUED FROM PREVIOUS PAGE – **DL RX/WX COMMUNICATION PROGRAM – FOR DL05, D2-250(-1), D4-450**)*

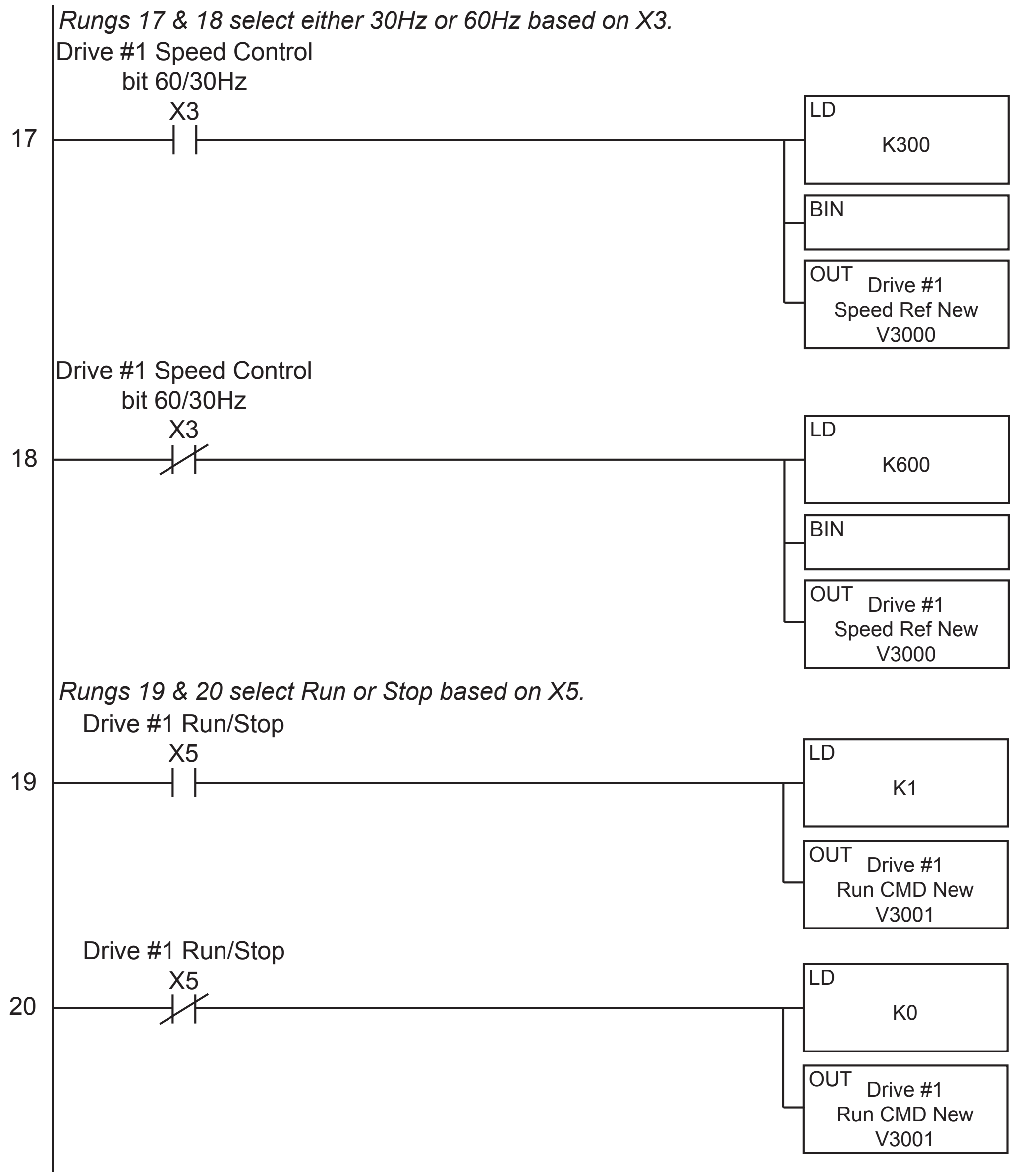
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

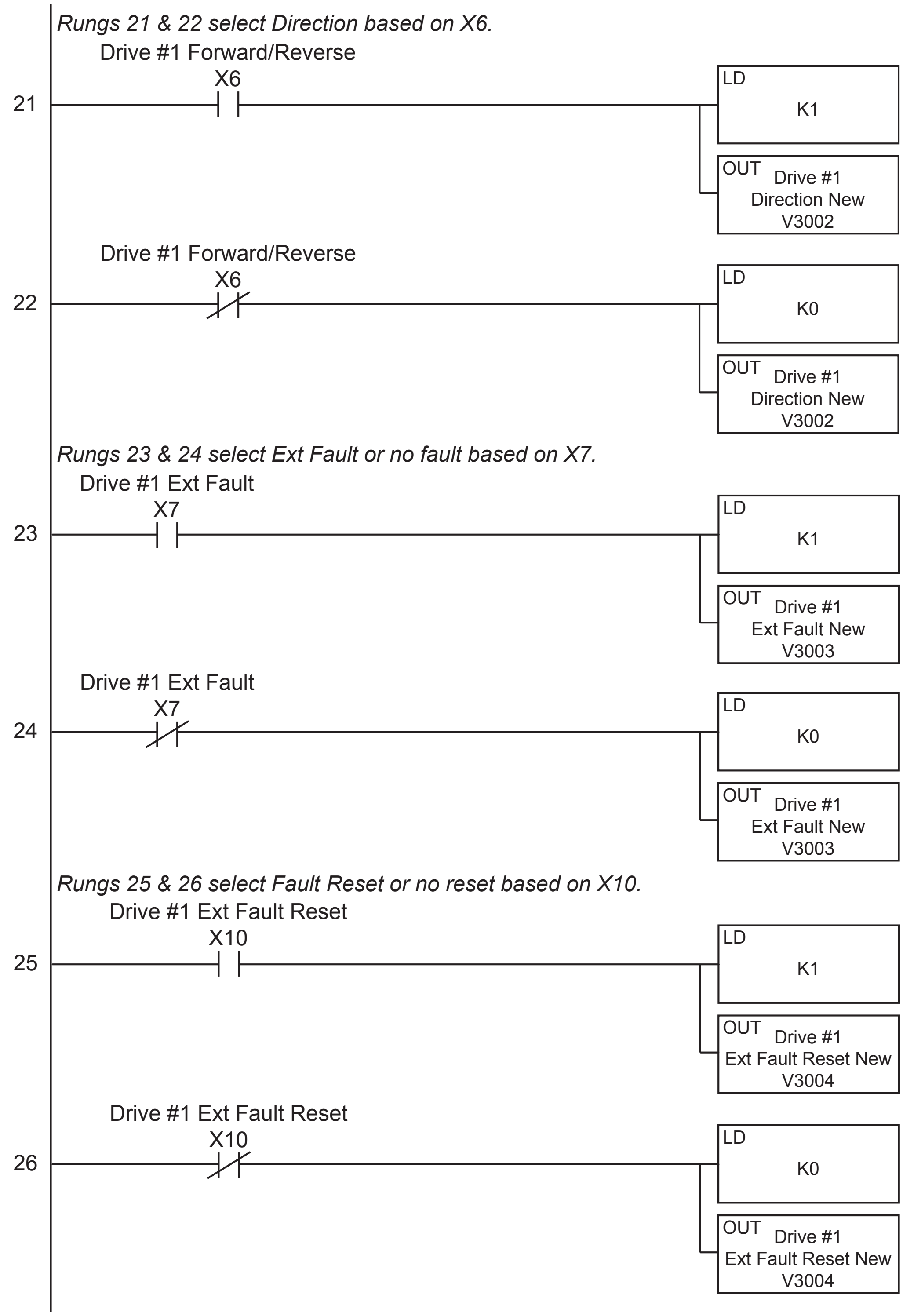
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

*(continued from previous page – **DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450**)*

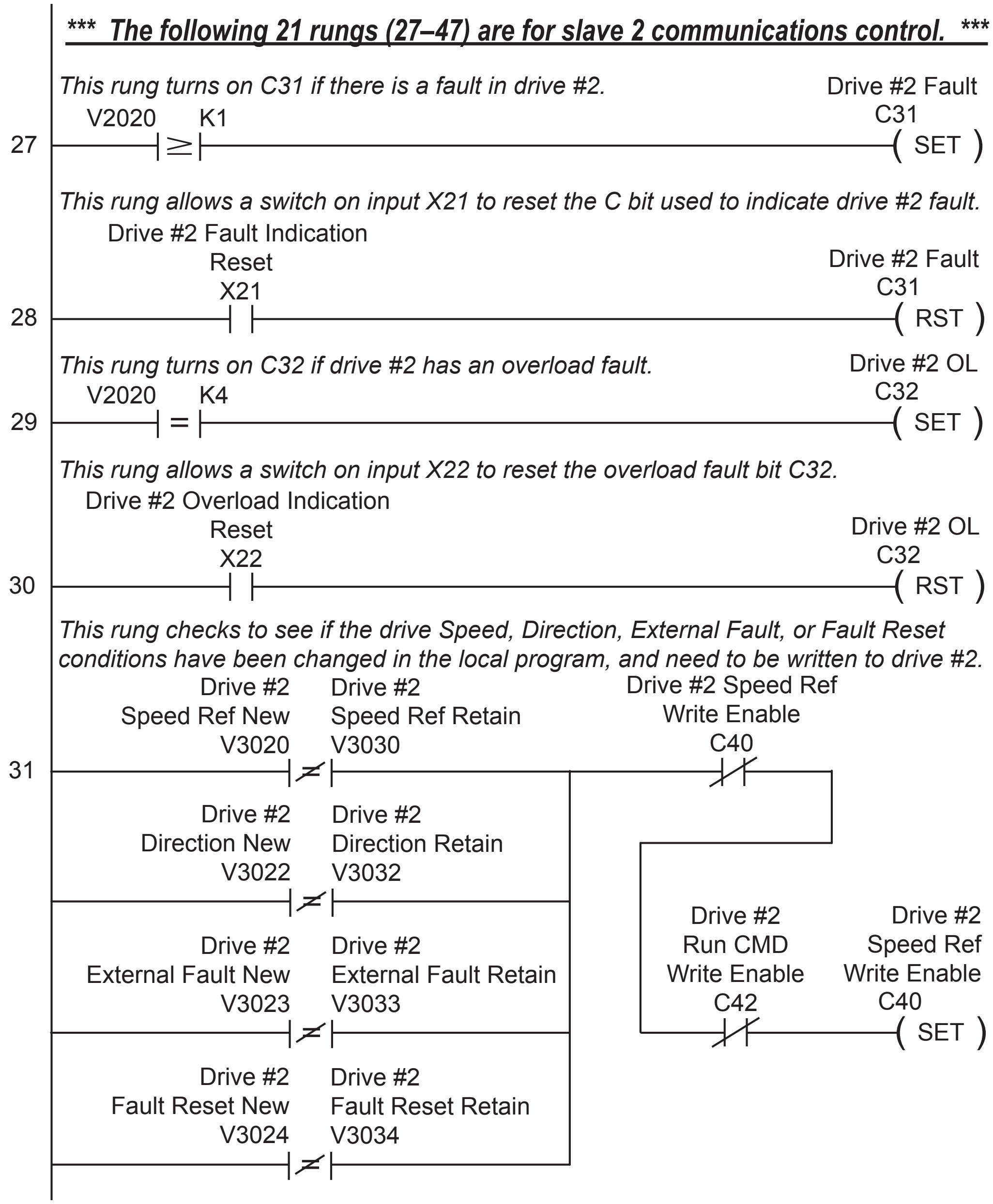
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

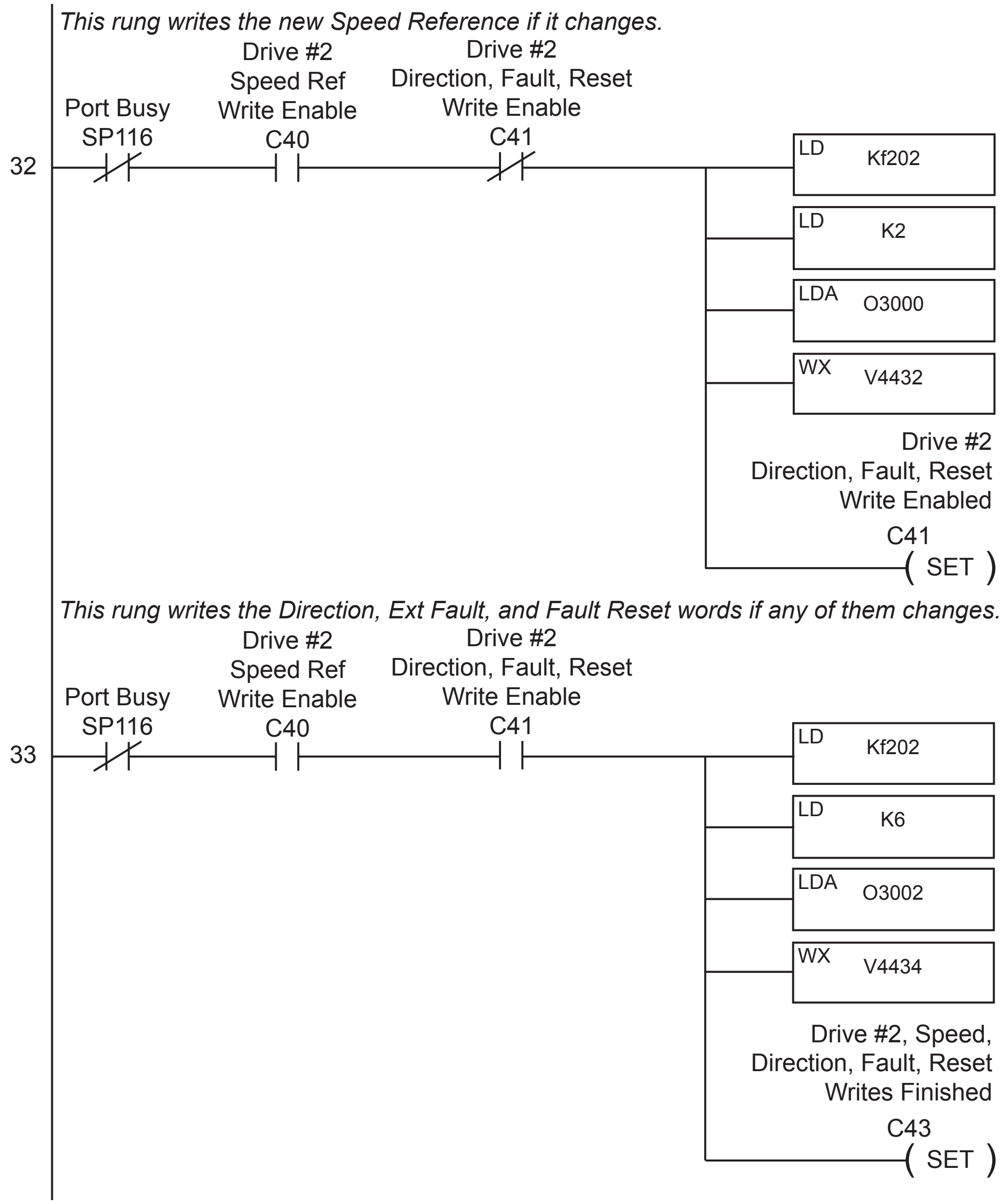
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

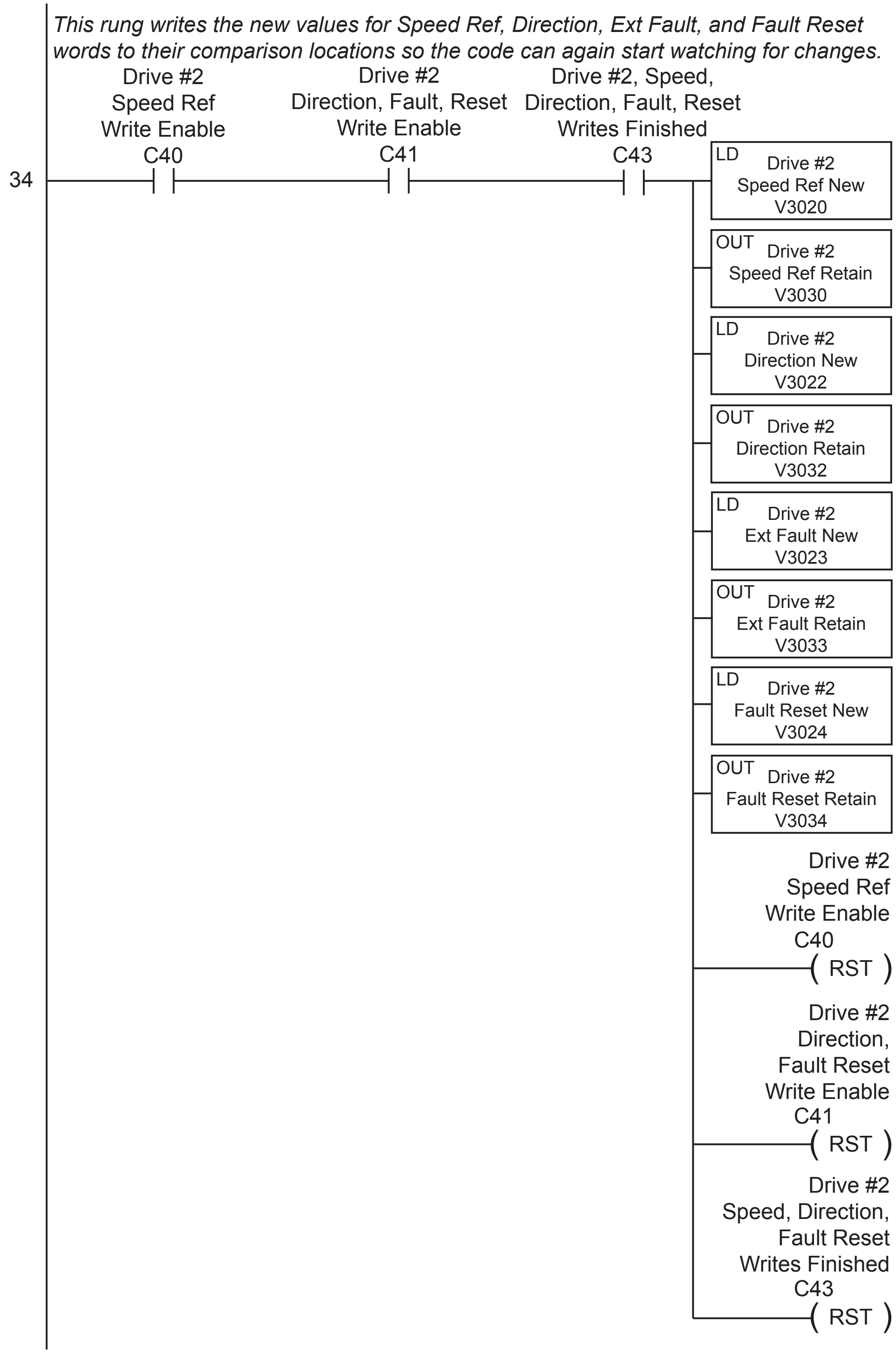
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

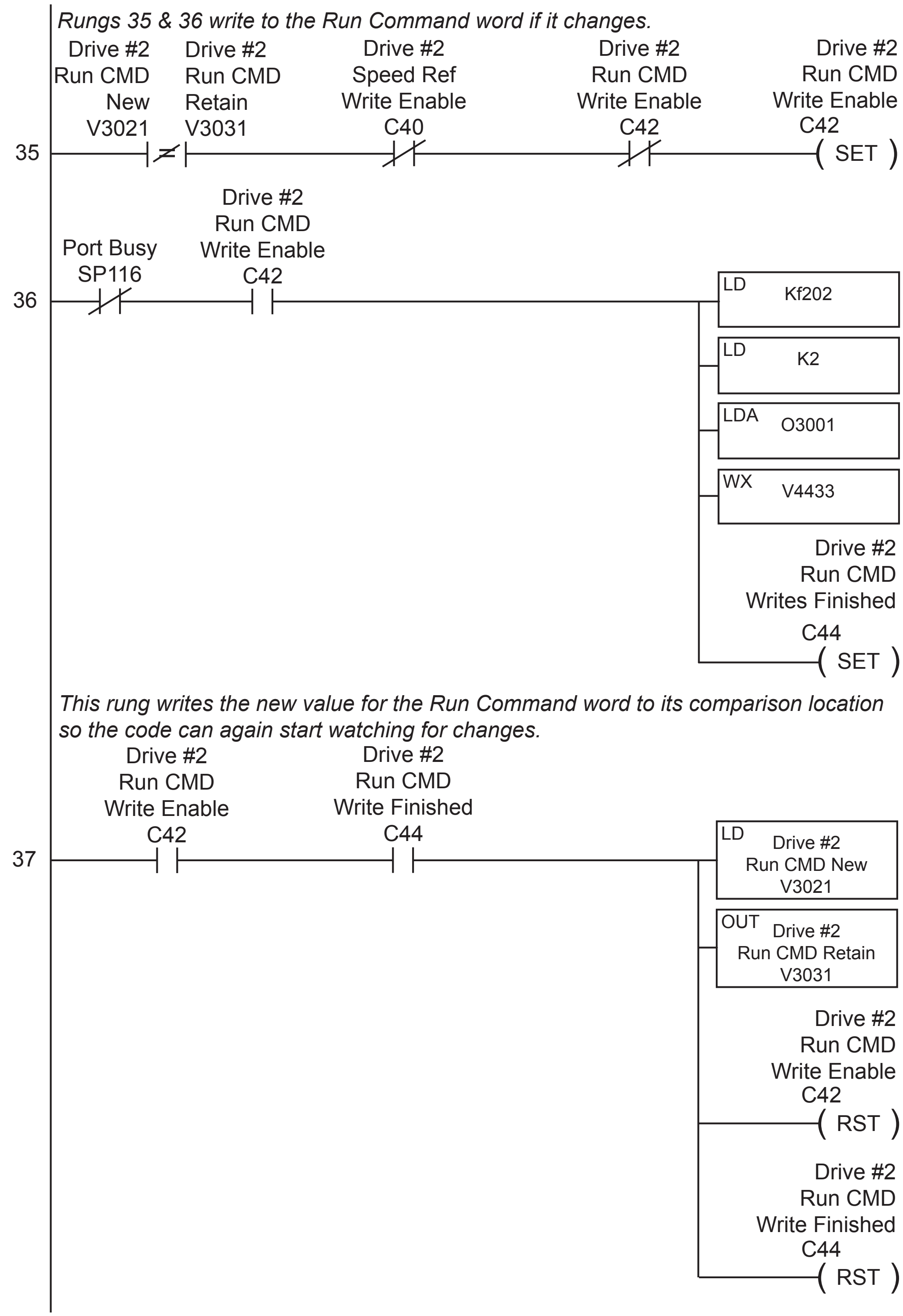
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

*(CONTINUED FROM PREVIOUS PAGE – **DL RX/WX COMMUNICATION PROGRAM – FOR DL05, D2-250(-1), D4-450**)*

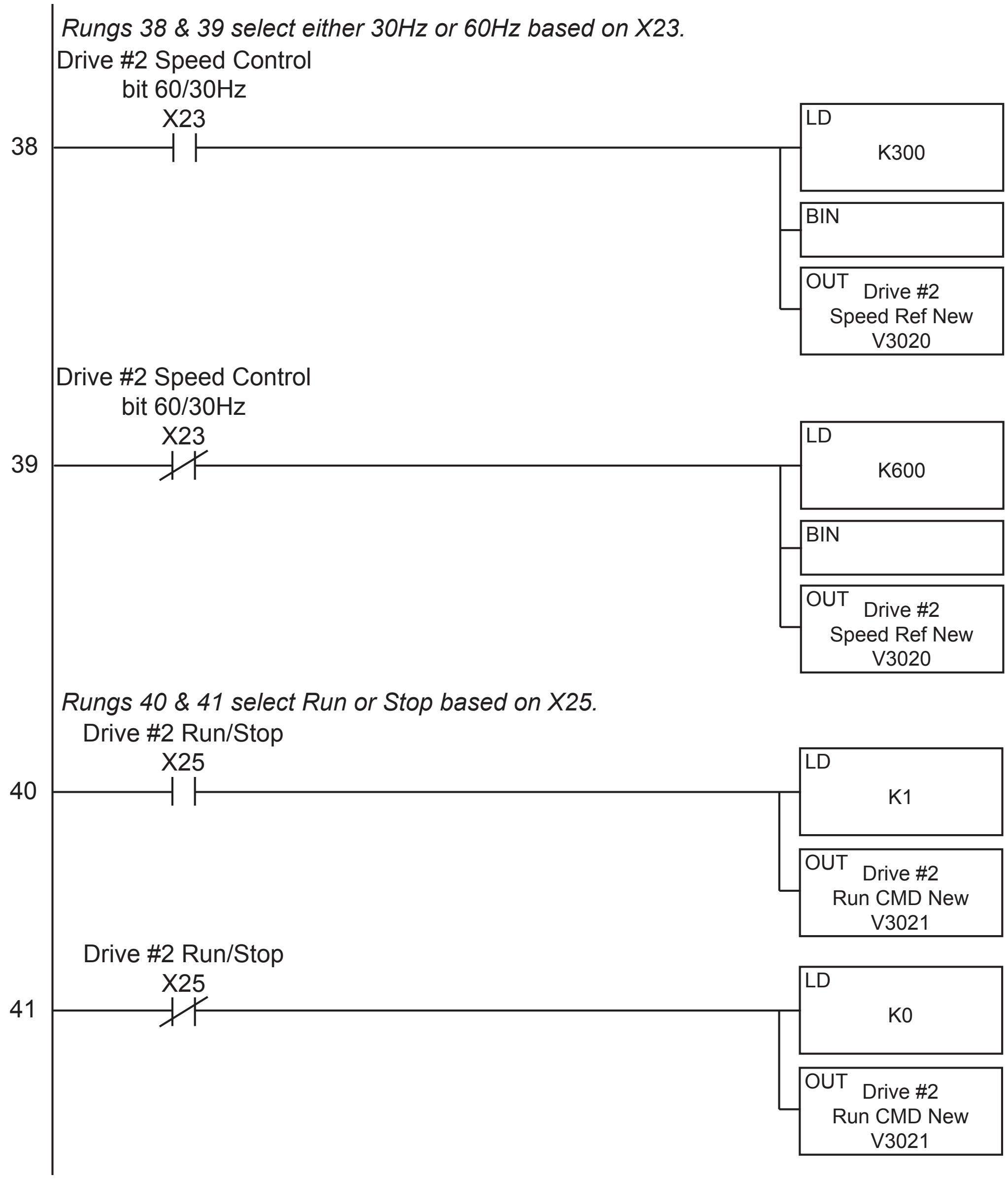
This program is for illustrative purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

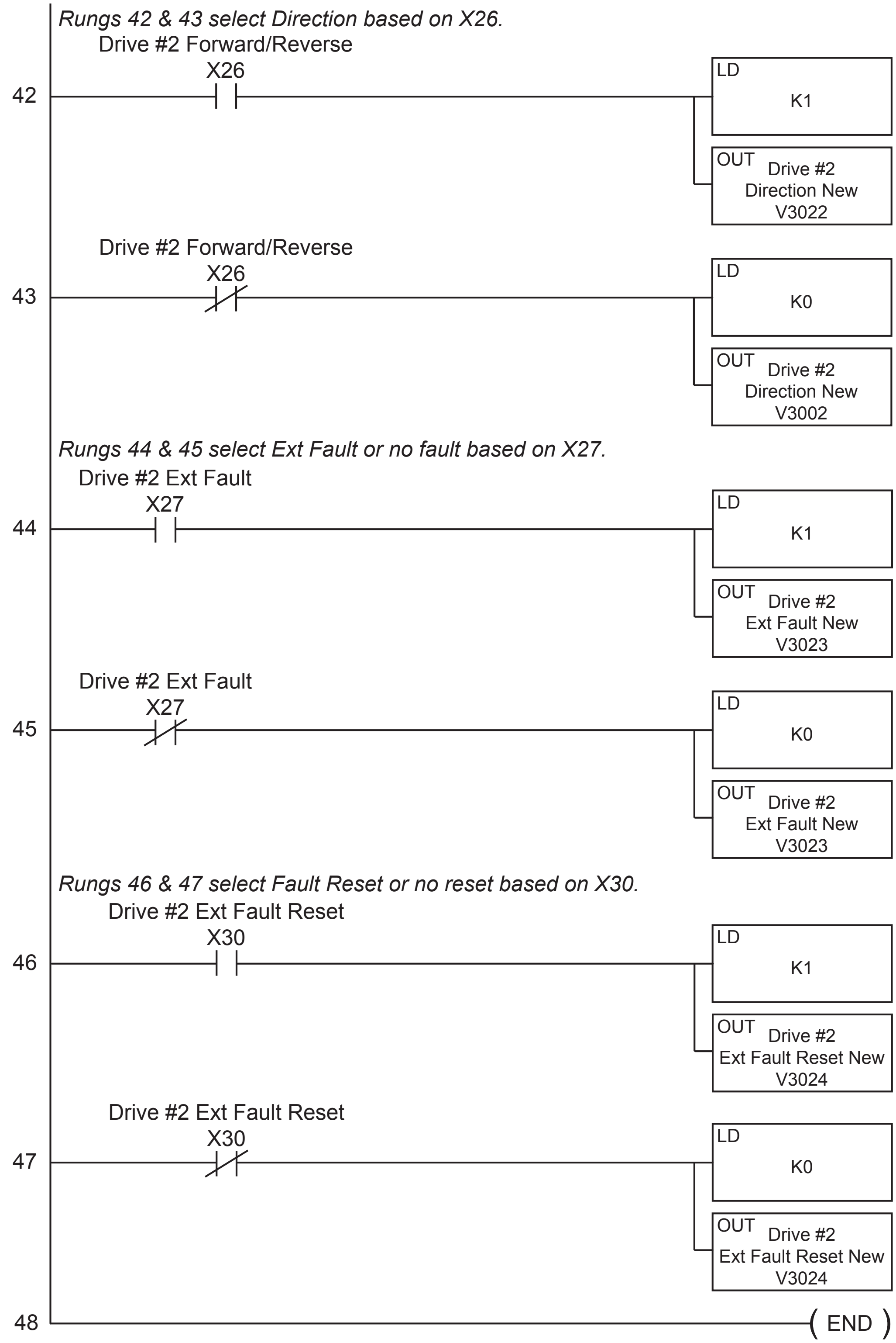
This program is for illustrational purposes only, and is not intended for a true application.



(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)

(continued from previous page – DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450)

This program is for illustrational purposes only, and is not intended for a true application.



Communicating with Third-Party Devices

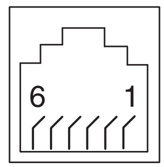
The GS1 Serial Comm Port will accommodate an RS-485 connection.

An RS-485 network cable can span up to 1000 meters (3280 feet). The GS1 AC drive communication address is specified by P9.00. The third party device then controls each AC drive according to its communication address.

The GS1 series AC drive can be set up to communicate on standard MODBUS networks using the following transmission modes: ASCII or RTU. Using the Communication Protocol parameter (P9.02), you can select the desired mode, data bits, parity, and stop bits. The mode and serial parameters must be the same for all devices on a MODBUS network.

GS1 RS-485 Serial Comm Port

GS1 Serial Comm Port
RS-485 Interface
RJ12 (6P4C)



1: +17V
2: GND
3: SG-
4: SG+
5: nc
6: reserved

GS1 drives have a provision for shutting down control or power to the inverter in the event of a communications time out. This feature can be set up through parameters P9.03, P9.04, and P9.05.

Common Third-Party MODBUS RTU Masters

- *MODSCAN from www.wintech.com*
- *KEPSERVER EX 4.0 from www.kepware.com*
- *Entivity Studio 7.2*
- *Think & Do Live 5.5.1*

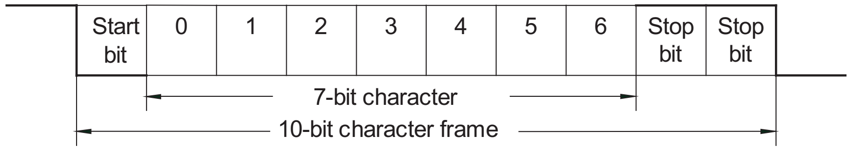
For additional technical assistance, go to our Technical support home page at:
http://support.automationdirect.com/technotes.html

Using Modbus ASCII

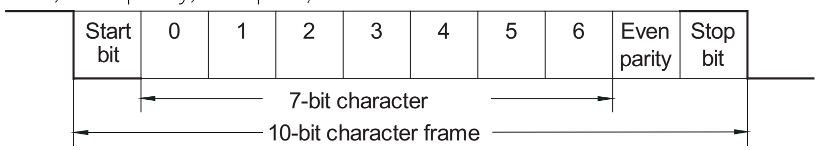
Data Format

ASCII Mode: 10-bit character frame (For 7-bit character):

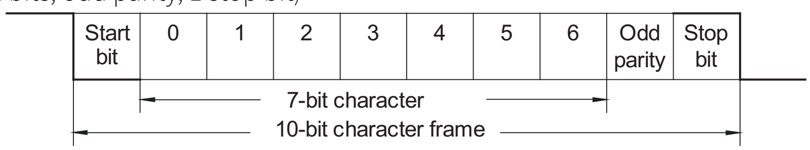
P9.02 = 00 (7 data bits, no parity, 2 stop bits)



P9.02 = 01 (7 data bits, even parity, 1 stop bit)

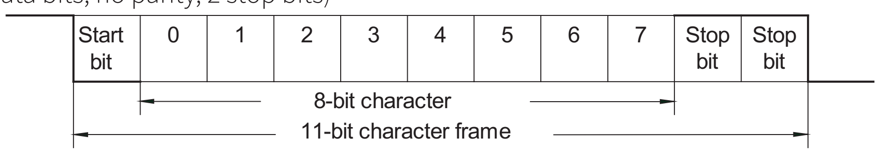


P9.02 = 02 (7 data bits, odd parity, 1 stop bit)

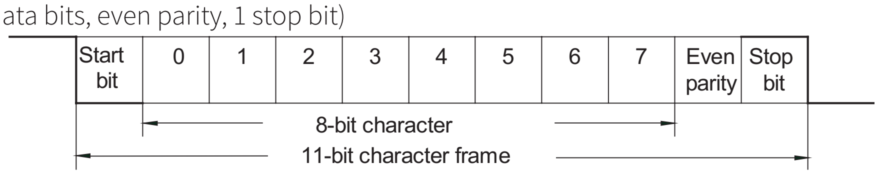


RTU Mode: 11-bit character frame (For 8-bit character):

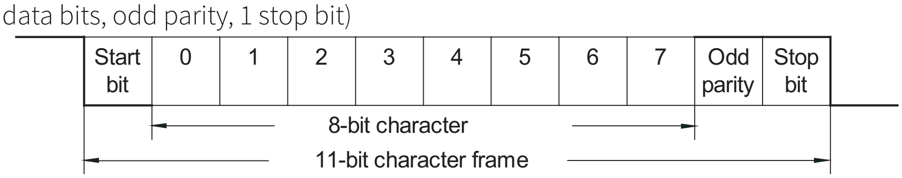
P9.02 = 03 (8 data bits, no parity, 2 stop bits)



P9.02 = 04 (8 data bits, even parity, 1 stop bit)



P9.02 = 05 (8 data bits, odd parity, 1 stop bit)



Communication Protocol

ASCII Mode:

STX	Start Character: (3AH)
ADR 1	Communication Address: 8-bit address consists of 2 ASCII codes
ADR 0	
CMD 1	
CMD 0	
DATA (n-1)	Contents of data: n x 8-bit data consists of 2n ASCII codes. n ≤ 25 maximum of 50 ASCII codes
....	
DATA 0	
LRC CHK 1	LRC check sum: 8-bit check sum consists of 2 ASCII codes
LRC CHK 0	
END 1	END characters: END 1 = CR (0DH); END 0 = LF (0AH)
END 0	

RTU Mode:

START	A silent interval of more than 10 ms
ADR	Communication Address: 8-bit address
CMD	Command Code: 8-bit command
DATA (n-1)	Contents of data: n x 8-bit data, n ≤ 25
....	
DATA 0	
CRC CHK Low	CRC check sum: 16-bit check sum consists of 2 8-bit characters
CRC CHK High	
END	A silent interval of more than 10 ms

ADR (Communication Address)

Valid communication addresses are in the range of 0 to 254. A communication address equal to 0 means broadcast to all AC drives, in which case the drives will not reply any message to the master device.

For example, communication to AC drive with address 16 decimal:

- *ASCII mode: (ADR 1, ADR 0)='1','0' => '1'=31H, '0'=30H*
- *RTU mode: (ADR)=10H*

CMD (Command Code) and DATA (Data Characters)

The format of data characters depends on the command code. The available command codes are described as follows: Command code: 03H, read N words. The maximum value of N is 12. For example, reading continuous 2 words from starting address 2102H of the AC drive with address 01H.

ASCII mode:

Command Message	
STX	':'
ADR 1	'0'
ADR 0	'1'
CMD 1	'0'
CMD 0	'3'
Starting data address	'2'
	'1'
	'0'
	'2'
Number of data (Count by word)	'0'
	'0'
	'0'
	'2'
LRC CHK 1	'D'
LRC CHK 0	'7'
END 1	CR
END 0	LF

Response Message	
STX ':'	':'
ADR 1	'0'
ADR 0	'1'
CMD 1	'0'
CMD 0	'3'
Number of data (Count by byte)	'0'
	'4'
Content of starting data address 2102H	'1'
	'7'
	'7'
	'0'
Content data address 2103H	'0'
	'0'
	'0'
	'0'
LRC CHK 1	'7'
LRC CHK 0	'1'
END 1	CR
END 0	LF

RTU mode:

Command Message	
ADR	01H
CMD	03H
Starting data address	21H
	02H
Number of data (Count by word)	00H
	02H
CRC CHK Low	6FH
CRC CHK High	F7H

Response Message	
ADR	01H
CMD	03H
Number of data (Count by byte)	04H
	'0'
Content of data address 2102H	17H
	70H
Content of data address 2103H	00H
	02H
CRC CHK Low	FEH
CRC CHK High	5CH

Command code: 06H, write 1 word

For example, writing 6000(1770H) to address 0100H of the AC drive with address 01H.

ASCII mode:

Command Message			Response Message	
STX	':'		STX ':'	':'
ADR 1 ADR 0	'0'		ADR 1 ADR 0	'0'
	'1'			'1'
CMD 1 CMD 0	'0'		CMD 1 CMD 0	'0'
	'6'			'6'
Data Address	'0'		Data Address	'0'
	'1'			'1'
	'0'			'0'
	'0'			'0'
	'1'		Data Content	'1'
	'7'			'7'
	'7'			'7'
	'0'			'0'
LRC CHK 1 LRC CHK 0	'7'		LRC CHK 1 LRC CHK 0	'7'
	'1'			'1'
END 1 END 0	CR		END 1 END 0	CR
	LF			LF

RTU mode:

This is an example of using function code 16 for writing to multiple registers.

Command Message			Response Message	
ADR	01H		ADR	01H
CMD	10H		CMD	10H
Starting data address	20H		Starting data address	20H
	00H			00H
Number of registers	00H		Number of data (Count by word)	00H
	02H			02H
Byte count	04H		CRC CHK Low CRC CHK High	4AH
Content of data address 2000H	00H			08H
	02H			
Content of data address 2001H	02H			
	58H			
CRC CHK Low CRC CHK High	CBH			
	34H			

CHK (check sum)

ASCII Mode:

LRC (Longitudinal Redundancy Check) is calculated by summing up module 256, the values of the bytes from ADR1 to last data character, then calculating the hexadecimal representation of the 2's-complement negation of the sum.

For example, reading 1 word from address 0401H of the AC drive with address 01H.

<table>
<tr><th colspan="2">Command Message</th><td rowspan="17">01H+03H+04H+01H+00H+01H=0AH;
the 2's complement negation of 0AH is F6H.</td></tr>
<tr><td>STX</td><td>':'</td></tr>
<tr><td rowspan="2">ADR 1
ADR 0</td><td>'0'</td></tr>
<tr><td>'1'</td></tr>
<tr><td rowspan="2">CMD 1
CMD 0</td><td>'0'</td></tr>
<tr><td>'3'</td></tr>
<tr><td rowspan="4">Starting data address</td><td>'0'</td></tr>
<tr><td>'4'</td></tr>
<tr><td>'0'</td></tr>
<tr><td>'1'</td></tr>
<tr><td rowspan="4">Number of data (Count by word)</td><td>'0'</td></tr>
<tr><td>'0'</td></tr>
<tr><td>'0'</td></tr>
<tr><td>'1'</td></tr>
<tr><td rowspan="2">LRC CHK 1
LRC CHK 0</td><td>'F'</td></tr>
<tr><td>'6'</td></tr>
<tr><td rowspan="2">END 1
END 0</td><td>CR</td></tr>
<tr><td>LF</td></tr>
</table>

RTU Mode:

Response Message	
ADR	01H
CMD	03H
Starting data address	21H
	02H
Number of data (Count by word)	00H
	02H
CRC CHK Low	6FH
CRC CHK High	F7H

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

1) Load a 16-bit register (called CRC register) with FFFFH.
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 CRC register is 0, repeat step 3; else Exclusive or the CRC register with the polynomial value A001H.
5) Repeat step 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 to 5 for the next 8-bit byte of the command message.

Continue doing this until all bytes have been processed. The final contents of the CRC register are 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.

The following is an example of CRC generation using C language. The function takes two arguments:

Unsigned char* data ← a pointer to the message buffer

Unsigned char length ← the quantity of bytes in the message buffer

The function returns the CRC value as a type of unsigned integer.

```
Unsigned int crc_chk(unsigned char* data, unsigned char length){
    int j;
    unsigned int reg_crc=0xFFFF;
    while(length--){
            reg_crc ^= *data++;
            for(j=0;j<8;j++){
                if(reg_crc & 0x01){  /* LSB(b0)=1 */
                    reg_crc=(reg_crc>>1) ^ 0xA001;
            }else{
                    reg_crc=reg_crc >>1;
            }
        }
    }
    return reg_crc;
}
```

RTU mode is preferred. Limited support is available to ASCII users.

Comm Delay – Optimizing Communications

Optimizing Communications to GS Drives

In most cases, optimizing communications to GS Drives MAY NOT BE NECESSARY.

If you are only communicating to one or two drives and reading or writing only a few parameters, the communication speed will most likely be sufficient for your application.

However, in the case that the communication speed (reaction time from reading or writing an event to a given drive) is too slow, you may need to take a more detailed look at how your code is designed to communicate to the GS Drives in your application.

To properly design the system, it is necessary to understand all of the propagation delays that are incurred when triggering the event to send a Modbus message to the point of receiving the data or status of the reply into the PLC or Modbus master.

To determine the time necessary to transmit a message from the Master to the Slave and vice versa, we must first determine the “Bit Time” and the “Character Time”. This is calculated by using the following formulas:

- *Bit Time:*
  The value one divided by the baud rate. A baud rate of 19,200 equals a bit rate of 0.0000528 (1/19200) or 52 μs (microseconds).
- *Character Time:*
  Bit Time multiplied by the number of bits. With Modbus this is typically 10–12 bits per character [1 start bit (fixed), 1 or 2 stop bits (usually configurable), 0 or 1 parity bit (Odd & Even = 1 bit; None = 0), & 8 data bits]. For a setting of Odd parity and 1 Stop bit, this would be 11 bits. So at 19200, Odd parity and 1 stop bit, a character time would be 0.000573 or 573 μs (0.0000528 · 11).

Now that we know the byte time, we can multiply that time by the number of characters in each message.

Types of Messages Sent to GS Drives

There are three different types of messages typically be sent to GS Drives:

1) Read Registers (Function Code 3).
2) Write Multiple Registers (Function Code 16).
3) Write Single Register (Function Code 6).

Format of "Read Registers" Messages:

Request:	Reply:
XX = Node Address (1 Char) 03 = Function Code (1 Char) XXXX = Starting Address to read (2 Chars) XXXX = Number of Registers to read (2 Chars) XXXX = 16 Bit CRC (2 Chars)	XX = Node Address (1 Char) 03 = Function Code (1 Char) XX = Byte count of data being sent from Slave (1 Char) XXXX... = Depends upon Request (2 Chars per Register requested) XXXX = 16 Bit CRC (2 Chars)

Format of "Write Multiple Registers" Messages:

Request:	Reply:
XX = Node Address (1 Char) 10 = Function Code (Hex format) (1 Char) XXXX = Starting Address to write to (2 Chars) XXXX = Number of Registers to write to (2 Chars) XX = Number of bytes of data to write (1 Char) XXXX... = Depends upon Request (2 Chars per Register requested) XXXX = 16 Bit CRC (2 Chars)	XX = Node Address (1 Char) 10 = Function Code (Hex format)(1 Char) XXXX = Starting Address to write to (2 Chars) XXXX = Number of Registers to write to (2 Chars) XXXX = 16 Bit CRC (2 Chars)

Format of "Write Single Register" Messages:

Request:	Reply:
XX = Node Address (1 Char) 06 = Function Code (1 Char) XXXX = Register to Write to (2 Chars) XXXX = Data to Write (2 Chars) XXXX = 16 Bit CRC (2 Chars)	XX = Node Address (1 Char) 06 = Function Code (1 Char) XXXX = Register to Write to (2 Chars) XXXX = Data to Write (2 Chars) XXXX = 16 Bit CRC (2 Chars)

Example Message:

Write a value of 60Hz to P9.26 and a value of 1 to P9.27 = **01 10 09 1b 00 02 04 02 58 00 01 5a 66**

We receive a good reply = **01 10 09 1b 00 02 a3 9f**

Sending message (13 characters from above) = **7.4 ms (0.00744796)**

Reply message (8 characters from above) = **4.6 ms (0.004583)**

For more specific information on how Modbus messages are formed, refer to the Modbus specifications found at www.modbus.org.

Additional Message Delay Times

So we have the total transmission time for sending a message and receiving a reply but this does not include all of the delays for a given message. The receiving device must have time to process the receipt of a message and formulate a reply. The amount of time that the receiving device needs will vary greatly depending upon the hardware platform and other processes that the device is running.

For the previous example message, the GS Drive responds in 4ms when the drive is stopped and will respond in 5ms when the drive is running. This may vary somewhat depending upon the specific parameter values and the size of the request.

Modbus-specified Delays Between Messages

There is one additional time delay required in the Modbus protocol. The protocol specifies at least a 3.5 character delay between messages. For the settings above, a 3.5 character time in our example would be about 2ms.

So the total time required for the message sent above would be:

7.4 ms	(Transmission time for sending message)
5.0 ms	(response delay from GS Drive when drive running)
4.6 ms	(Transmission time for reply message)
+ 2.0 ms	(Modbus message wait delay)
19.0 ms	(approximately)

Remember, from our description, that this is purely the time from when the message leaves the serial port to when the reply is received back in to the serial port.

Other Delays

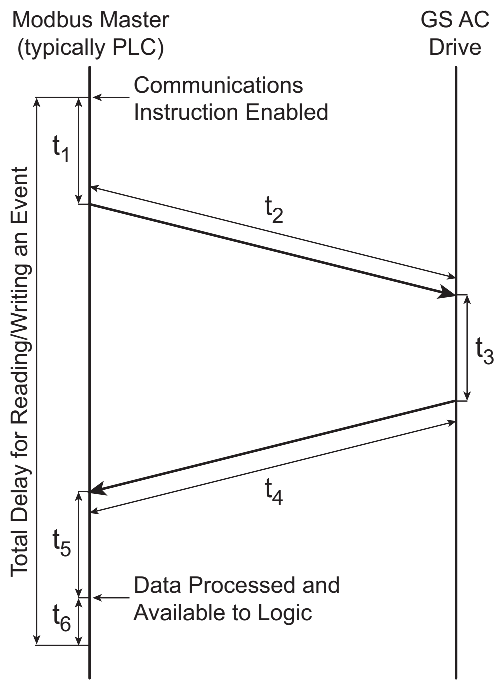
Depending upon the master device, there may be additional delays. For example:

In the DirectLogic PLC, the serial communications are serviced in the housekeeping portion of the PLC scan. So if the communications instruction is in rung #1 of a ladder program, the serial communications message does not get sent until the end of the total PLC scan. Likewise, if the reply message was received into the serial port at the beginning of the PLC scan, it would not be serviced until the end of the PLC scan.

So you would need to add an additional possible two PLC scan times to the number above to truly calculate the time necessary to read or write an event to the GS drive.

These delays are shown in the following Communication Delay Timing Diagram.

Communication Delay Timing Diagram



t_1 = Scan delay from the point of turning on a communications instruction to when it actually goes out of the serial port.

t_2 = Transmission time to send Message request (read or write).

t_3 = Response delay from GS drive to receive the reply and formulate the response.

t_4 = Transmission time to send Reply message.

t_5 = Scan delay from the point of receiving reply, processing it and placing in PLC memory for Logic usage.

t_6 = Wait time required by Modbus spec (3.5 byte times). This may or may not be present depending upon the Scan delay, but safer to factor in.

Communication Delay Summary

Now that you know how to calculate the time required for one message to one GS drive, you would simply multiply this value per message to each GS drive on the network, since only one message can be sent at a time.

As you can deduce from the statement above, the more messages being sent to GS drives, the longer it takes to communicate to an individual drive as each message has to take its turn.

So how do you optimize your communications to get messages faster to your GS drives?

There is no way to make a message go faster than what is specified above, but what you can affect is the amount of messages being sent to any given GS drive in two ways.

1) Group together messages into Block requests whenever possible. For example, if you wanted to read Status Monitor 1 and the Output Frequency status register from the drive, read the two together as a block (Status Monitor 1, Status Monitor 2, Frequency Command and Output Frequency), and ignore the other two status registers that you don't need instead of sending two separate read commands. If you do the calculations above, you will see that is much faster to take the additional hit from four extra bytes in the reply message than it would be to send a separate message. NOTE that you cannot read across non-contiguous Modbus addresses, so this typically only works when reading within the Status registers or in a Parameter category (P9.xx, P1.xx, etc…).
2) Only send a write message when the value changes in the Master device. It is simpler to setup your communications instructions to read and write all the time, but it wastes precious network time to write the same value to the GS drive over and over if that value is not changing. Write some simple logic that only triggers a write command when the value to be sent has changed.

For more specific instructions on how to configure and/or interlock, in detail, the individual communications instructions, consult your PLC or Modbus Master Device user manual. If using DirectLogic PLCs as the Modbus Master, consult the Dx-USER-M manuals for specifics on configuring the individual communications instructions and look at the Hx-ECOM-M manual for information on interlocking communications instructions.