Chapter 5

GS1 Modbus Communications

Contents of this Chapter

Communications Parameters Summary (P9.xx) ................................................................. 5–2
GS1 Parameter Memory Addresses ......................................................................................... 5–3
GS1 Status Addresses ............................................................................................................ 5–7
Block Transfer Parameters for Modbus Programs ................................................................. 5–9
Communicating with AutomationDirect PLCs ..................................................................... 5–9
  Step 1: Choose the Appropriate CPU ................................................................................. 5–9
  Step 2: Make the Connections ......................................................................................... 5–9
  Step 3: Set AC Drive Parameters ..................................................................................... 5–13
  Step 4: Configure the PLC CPU ....................................................................................... 5–13
CLICK Modbus Ladder Programming .................................................................................. 5–17
  Separate Run Command Write Instruction ....................................................................... 5–17
  CLICK Communication Program Example – (for CLICK PLCs) ......................................... 5–18
DirectLOGIC Modbus Ladder Programming ...................................................................... 5–32
  Separate Run Command Write Instruction ....................................................................... 5–32
  Block Transfer Parameters for Modbus Programs .............................................................. 5–32
  DirectLOGIC Basic Communication Program – start with this code ................................. 5–33
  Programming Differences for DirectLOGIC PLCs .......................................................... 5–34
  DL MRX/MWX Communication Program – for DL06 & D2-260 PLCs ............................. 5–35
  DL RX/WX Communication Program – for DL05, D2-250(-1), D4-450 PLCs .................. 5–48
Communicating with Third-Party Devices .......................................................................... 5–61
  Common Third-Party MODBUS RTU Masters ................................................................. 5–61
  Using Modbus ASCII ....................................................................................................... 5–62
Comm Delay – Optimizing Communications ...................................................................... 5–68
  Optimizing Communications to GS Drives ....................................................................... 5–68
  Types of Messages Sent to GS Drives .............................................................................. 5–69
  Format of “Read Registers” Messages: ............................................................................ 5–69
  Format of “Write Multiple Registers” Messages: .............................................................. 5–69
  Format of “Write Single Register” Messages: .................................................................... 5–69
  Example Message: ........................................................................................................... 5–69
  Additional Message Delay Times ..................................................................................... 5–70
  Communication Delay Summary: ...................................................................................... 5–72
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>P9.00</td>
<td>Communication Address</td>
<td>1 to 254</td>
<td>1</td>
</tr>
<tr>
<td>P9.01</td>
<td>Transmission Speed</td>
<td>0: 4800 baud, 1: 9600 baud, 2: 19200 baud</td>
<td>1</td>
</tr>
<tr>
<td>P9.02</td>
<td>Communication Protocol</td>
<td>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</td>
<td>0</td>
</tr>
<tr>
<td>P9.03</td>
<td>Transmission Fault Treatment</td>
<td>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</td>
<td>0</td>
</tr>
<tr>
<td>P9.04</td>
<td>Time Out Detection</td>
<td>0: Disable, 1: Enable</td>
<td>0</td>
</tr>
<tr>
<td>P9.05</td>
<td>Time Out Duration</td>
<td>0.1 to 60.0 seconds</td>
<td>0.5</td>
</tr>
<tr>
<td>♦ P9.07</td>
<td>Parameter Lock</td>
<td>0: All parameters can be set and read, 1: All parameters are read-only</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.08</td>
<td>Restore to Default</td>
<td>99: Restores all parameters to factory defaults</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.11</td>
<td>Block Transfer Parameter 1</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.12</td>
<td>Block Transfer Parameter 2</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.13</td>
<td>Block Transfer Parameter 3</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.14</td>
<td>Block Transfer Parameter 4</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.15</td>
<td>Block Transfer Parameter 5</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.16</td>
<td>Block Transfer Parameter 6</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.17</td>
<td>Block Transfer Parameter 7</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.18</td>
<td>Block Transfer Parameter 8</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.19</td>
<td>Block Transfer Parameter 9</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.20</td>
<td>Block Transfer Parameter 10</td>
<td>Parameters 0.00 to 8.01, 9.99</td>
<td>9.99</td>
</tr>
<tr>
<td>♦ P9.26</td>
<td>Serial Comm Speed Reference</td>
<td>0.0 to 400.0 Hz</td>
<td>60.0</td>
</tr>
<tr>
<td>♦ P9.27</td>
<td>Serial Comm RUN Command</td>
<td>0: Stop, 1: Run</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.28</td>
<td>Serial Comm Direction Command</td>
<td>0: Forward, 1: Reverse</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.29</td>
<td>Serial Comm External Fault</td>
<td>0: No fault, 1: External fault</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.30</td>
<td>Serial Comm Fault Reset</td>
<td>0: No action, 1: Fault Reset</td>
<td>0</td>
</tr>
<tr>
<td>♦ P9.31</td>
<td>Serial Comm JOG Command</td>
<td>0: Stop, 1: Jog</td>
<td>0</td>
</tr>
<tr>
<td>P9.39 *</td>
<td>Firmware Version</td>
<td>#.#</td>
<td>#.#</td>
</tr>
<tr>
<td>P9.41</td>
<td>GS Series Number</td>
<td>1: GS1, 2: GS2, 3: GS3, 4: GS4</td>
<td>#</td>
</tr>
<tr>
<td>P9.42</td>
<td>Manufacturer Model Information</td>
<td>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)</td>
<td>#</td>
</tr>
</tbody>
</table>

*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)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0.00</td>
<td>Motor Nameplate Voltage</td>
<td>0000</td>
<td>40001</td>
<td>0</td>
</tr>
<tr>
<td>P0.01</td>
<td>Motor Nameplate Amps</td>
<td>0001</td>
<td>40002</td>
<td>1</td>
</tr>
<tr>
<td>P0.02</td>
<td>Motor Base Frequency</td>
<td>0002</td>
<td>40003</td>
<td>2</td>
</tr>
<tr>
<td>P0.03</td>
<td>Motor Base RPM</td>
<td>0003</td>
<td>40004</td>
<td>3</td>
</tr>
<tr>
<td>P0.04</td>
<td>Motor Maximum RPM</td>
<td>0004</td>
<td>40005</td>
<td>4</td>
</tr>
</tbody>
</table>

* 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)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.00</td>
<td>Stop Methods</td>
<td>0100</td>
<td>40257</td>
<td>400</td>
</tr>
<tr>
<td>♦ P1.01</td>
<td>Acceleration Time 1</td>
<td>0101</td>
<td>40258</td>
<td>401</td>
</tr>
<tr>
<td>♦ P1.02</td>
<td>Deceleration Time 1</td>
<td>0102</td>
<td>40259</td>
<td>402</td>
</tr>
<tr>
<td>P1.03</td>
<td>Accel S-curve</td>
<td>0103</td>
<td>40260</td>
<td>403</td>
</tr>
<tr>
<td>P1.04</td>
<td>Decel S-curve</td>
<td>0104</td>
<td>40261</td>
<td>404</td>
</tr>
<tr>
<td>♦ P1.05</td>
<td>Acceleration Time 2</td>
<td>0105</td>
<td>40262</td>
<td>405</td>
</tr>
<tr>
<td>♦ P1.06</td>
<td>Deceleration Time 2</td>
<td>0106</td>
<td>40263</td>
<td>406</td>
</tr>
<tr>
<td>P1.07</td>
<td>Select method to use 2nd Accel/Decel</td>
<td>0107</td>
<td>40264</td>
<td>407</td>
</tr>
<tr>
<td>P1.08</td>
<td>Accel 1 to Accel 2 frequency transition</td>
<td>0108</td>
<td>40265</td>
<td>410</td>
</tr>
<tr>
<td>P1.09</td>
<td>Decel 1 to Decel 2 frequency transition</td>
<td>0109</td>
<td>40266</td>
<td>411</td>
</tr>
<tr>
<td>P1.10</td>
<td>Skip Frequency 1</td>
<td>010A</td>
<td>40267</td>
<td>412</td>
</tr>
<tr>
<td>P1.11</td>
<td>Skip Frequency 2</td>
<td>010B</td>
<td>40268</td>
<td>413</td>
</tr>
<tr>
<td>P1.12</td>
<td>Skip Frequency 3</td>
<td>010C</td>
<td>40269</td>
<td>414</td>
</tr>
<tr>
<td>P1.17</td>
<td>Skip Frequency Band</td>
<td>0111</td>
<td>40274</td>
<td>421</td>
</tr>
<tr>
<td>P1.19</td>
<td>DC Injection Voltage Level</td>
<td>0113</td>
<td>40276</td>
<td>423</td>
</tr>
<tr>
<td>P1.20</td>
<td>DC Injection during Start-up</td>
<td>0114</td>
<td>40277</td>
<td>424</td>
</tr>
<tr>
<td>P1.21</td>
<td>DC Injection during Stopping</td>
<td>0115</td>
<td>40278</td>
<td>425</td>
</tr>
<tr>
<td>P1.22</td>
<td>Start-point for DC Injection</td>
<td>0116</td>
<td>40279</td>
<td>426</td>
</tr>
</tbody>
</table>

♦ 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)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2.00</td>
<td>Volts/Hertz Settings</td>
<td>0200</td>
<td>40513</td>
<td>1000</td>
</tr>
<tr>
<td>♦ P2.01</td>
<td>Slip Compensation</td>
<td>0201</td>
<td>40514</td>
<td>1001</td>
</tr>
<tr>
<td>♦ P2.03</td>
<td>Manual Torque Boost</td>
<td>0203</td>
<td>40516</td>
<td>1003</td>
</tr>
<tr>
<td>P2.04</td>
<td>Mid-point Frequency</td>
<td>0204</td>
<td>40517</td>
<td>1004</td>
</tr>
<tr>
<td>P2.05</td>
<td>Mid-point Voltage</td>
<td>0205</td>
<td>40518</td>
<td>1005</td>
</tr>
<tr>
<td>P2.06</td>
<td>Min. Output Frequency</td>
<td>0206</td>
<td>40519</td>
<td>1006</td>
</tr>
<tr>
<td>P2.07</td>
<td>Min. Output Voltage</td>
<td>0207</td>
<td>40520</td>
<td>1007</td>
</tr>
<tr>
<td>P2.08</td>
<td>PWM Carrier Frequency</td>
<td>0208</td>
<td>40521</td>
<td>1010</td>
</tr>
</tbody>
</table>

♦ 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.
### Chapter 5: Communications

#### Parameter Memory Addresses – Digital Parameters (P3.xx)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3.00</td>
<td>Source of Operation Command</td>
<td>0300</td>
<td>40769</td>
<td>1400</td>
</tr>
<tr>
<td>P3.01</td>
<td>Multi-function Inputs 1 &amp; 2 (DI1 &amp; DI2)</td>
<td>0301</td>
<td>40770</td>
<td>1401</td>
</tr>
<tr>
<td>P3.02</td>
<td>Multi-function Input 3 (DI3)</td>
<td>0302</td>
<td>40771</td>
<td>1402</td>
</tr>
<tr>
<td>P3.03</td>
<td>Multi-function Input 4 (DI4)</td>
<td>0303</td>
<td>40772</td>
<td>1403</td>
</tr>
<tr>
<td>P3.11</td>
<td>Multi-Function Output Terminal</td>
<td>030B</td>
<td>40780</td>
<td>1413</td>
</tr>
<tr>
<td>♦ P3.16</td>
<td>Desired Frequency</td>
<td>0310</td>
<td>40785</td>
<td>1420</td>
</tr>
<tr>
<td>♦ P3.17</td>
<td>Desired Current</td>
<td>0311</td>
<td>40786</td>
<td>1421</td>
</tr>
</tbody>
</table>

* 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)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4.00</td>
<td>Source of Frequency Command</td>
<td>0400</td>
<td>41025</td>
<td>2000</td>
</tr>
<tr>
<td>P4.01</td>
<td>Analog Input Offset Polarity</td>
<td>0401</td>
<td>41026</td>
<td>2001</td>
</tr>
<tr>
<td>♦ P4.02</td>
<td>Analog Input Offset</td>
<td>0402</td>
<td>41027</td>
<td>2002</td>
</tr>
<tr>
<td>♦ P4.03</td>
<td>Analog Input Gain</td>
<td>0403</td>
<td>41028</td>
<td>2003</td>
</tr>
<tr>
<td>P4.04</td>
<td>Analog Input Reverse Motion Enable</td>
<td>0404</td>
<td>41029</td>
<td>2004</td>
</tr>
<tr>
<td>P4.05</td>
<td>Loss of ACI Signal (4–20mA)</td>
<td>0405</td>
<td>41030</td>
<td>2005</td>
</tr>
</tbody>
</table>

* 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)

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>♦ P5.00</td>
<td>Jog</td>
<td>0500</td>
<td>41281</td>
<td>2400</td>
</tr>
<tr>
<td>♦ P5.01</td>
<td>Multi-Speed 1</td>
<td>0501</td>
<td>41282</td>
<td>2401</td>
</tr>
<tr>
<td>♦ P5.02</td>
<td>Multi-Speed 2</td>
<td>0502</td>
<td>41283</td>
<td>2402</td>
</tr>
<tr>
<td>♦ P5.03</td>
<td>Multi-Speed 3</td>
<td>0503</td>
<td>41284</td>
<td>2403</td>
</tr>
</tbody>
</table>

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

♦ 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).

<table>
<thead>
<tr>
<th>GS1 Parameter</th>
<th>Description</th>
<th>Hexadecimal</th>
<th>Modbus Decimal *</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>♦ P8.00</td>
<td>User Defined Display Function</td>
<td>0800</td>
<td>42049</td>
<td>4000</td>
</tr>
<tr>
<td>♦ P8.01</td>
<td>Frequency Scale Factor</td>
<td>0801</td>
<td>42050</td>
<td>4001</td>
</tr>
</tbody>
</table>

♦ 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)

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

- 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)

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

Status Monitor 1 – Error Codes

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

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

**GS1 Memory Address (hexadecimal)**

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

**Frequency Command F (xxx.x)**

Status location for the frequency setting of the AC drive.

**Output Frequency H (xxx.x)**

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

**Output Current A (xxx.x)**

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

**DC BUS Voltage d (xxx.x)**

Status location for the DC Bus Voltage.

**Output Voltage U (xxx.x)**

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

**Motor RPM**

Status location for the present estimated speed of the motor.

**Scale Frequency (Low word)**

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

**Scale Frequency (High word)**

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

**% Load**

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

**Firmware Version**

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.

<table>
<thead>
<tr>
<th>Choose Your CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary Choices</strong></td>
</tr>
<tr>
<td>CLICK Analog CPU with Send/Receive instructions &amp; RS-485 comm port</td>
</tr>
<tr>
<td>D2-260 or DL06 with MRX / MWX instructions &amp; RS-485 comm port</td>
</tr>
<tr>
<td><strong>Secondary Choices</strong></td>
</tr>
<tr>
<td>CLICK Basic CPU with Send/Receive instructions &amp; RS-232 comm port</td>
</tr>
<tr>
<td>DL05, D2-250(-1), or D4-450 with RX / WX instructions &amp; RS-232 comm port</td>
</tr>
</tbody>
</table>

**Step 2: Make the Connections**

**GS1 RS-485 Serial Comm Port**

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

<p>| PLC Connections for RS-485 Modbus RTU Control of GS1 Drive |
|-------------|-------------|-------------|-------------|-------------|</p>
<table>
<thead>
<tr>
<th><strong>Drive</strong></th>
<th><strong>PLC Type</strong></th>
<th><strong>PLC Port</strong></th>
<th><strong>Communication</strong></th>
<th><strong>Direct Cable</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GS1 CLICK</td>
<td>3</td>
<td>RS-485</td>
<td>ZL-RJ12-CBL-2P</td>
<td>2m [6.6 ft]</td>
</tr>
<tr>
<td>DL05</td>
<td>2 **</td>
<td>RS-232 – RS-485 **</td>
<td>N/A **</td>
<td></td>
</tr>
<tr>
<td>DL06 D0-DCM</td>
<td>2</td>
<td>RS-485</td>
<td>GS-485HD15-CBL-2</td>
<td>2m [6.6 ft]</td>
</tr>
<tr>
<td>D2-DCM D2-250(-1)</td>
<td>2 **</td>
<td>RS-232 – RS-485 **</td>
<td>N/A **</td>
<td></td>
</tr>
<tr>
<td>D2-260</td>
<td>3 **</td>
<td>RS-485</td>
<td>GS-485HD15-CBL-2</td>
<td>2m [6.6 ft]</td>
</tr>
<tr>
<td>D4-450</td>
<td>3 **</td>
<td>RS-232 – RS-485 **</td>
<td>N/A **</td>
<td></td>
</tr>
</tbody>
</table>

* 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 500Ω).

Recommended RS-485 cable: Belden 9842 or equivalent.

**CLICK C0-02: RS-485 Connection Wiring**

![Click C0-02 Diagram]

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

![D0-DCM/DL06/DL260 Connection Diagram]

* 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)

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**

---

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.
Chapter 5: Communications

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

![Diagram of RS-232C to RS-485 connection wiring for D0-DCM/DL250(-1)]

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**

![Diagram of RS-232C to RS-485 connection wiring for DL350/DL450](image)

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 setup 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 setup 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.

---

1. **_Port_3_Ready_Flag**
   - **SC102**
   - **Up**
   - **CT1**
   - **Complete**

2. **_Port_3_Error_Flag**
   - **SC103**
   - **Up**
   - **CT2**
   - **Complete**

3. **_Port_3_Ready_Flag**
   - **SC102**
   - **Up**
   - **CT3**
   - **Complete**

---

*This rung counts the number of times the PLC attempts to communicate with the drives.*

*This rung counts the number of comm attempts that failed.*

*This rung acts as an alternator, allowing the following logic to alternate between Drive #1 and Drive #2. If there were additional drives, the Setpoint for the counter would simply be increased to match the number of drives.*

*(continued next 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.

```
Port_3_Ready_Flag  =  CTD3  =  0

Drive #1 Direction,
  Fault, Reset,
  Write-Enable
SC102  C11  C12

Drive #1 CMD
Write-Enable
C10

Receive (Port3)  MODBUS
Slave ID
1
Modbus Function Code
03
Slave Addr
408449
NO. of Master Addresses
12
Word Swap
OFF

Status from Drive #1
DS1
```

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.

```
Port_3_Ready_Flag  =  CTD3  =  1

Drive #2 Direction,
  Fault, Reset,
  Write-Enable
SC102  C31  C32

Drive #2 Run CMD
Write-Enable
C30

Receive (Port3)  MODBUS
Slave ID
2
Modbus Function Code
03
Slave Addr
408449
NO. of Master Addresses
12
Word Swap
OFF

Status from Drive #2
DS20
```

(continued next page – CLICK PLC communication program example)
This program is for illustrational purposes only, and is not intended for a true application.

This rung resets all the Receive status coils if either comm event is successful.

Read Drive #1 Success

Read Drive #2 Success

** The following rungs are used for Drive #1 communications, through rung #27 **

Status from Drive #1

Drive #1 Fault Indication

Drive #1 Fault

Drive #1 Overload Indicator

Drive #1 Overload

This rung determines if the Speed, Direction, Ext Fault, or Fault Reset words have changed and need to be written.

Drive #1 Speed Ref New

Drive #1 Speed Ref New

Drive #1 Speed Ref New

Drive #1 Speed Ref Write Enable

Drive #1 Run CMD Write Enable

Drive #1 Speed Ref Write Enable

Drive #1 Fault

Drive #1 Fault

Drive #1 Fault

(continued next page – CLICK PLC communication program example)
This program is for illustrational purposes only, and is not intended for a true application.

This rung writes the new Speed Reference if it changes.

This rung writes the Direction, Ext Fault, and Fault Reset words if any of them changes.

(continued next page – CLICK PLC communication program example)
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.

Drive #1 Speed Ref Write Enable

Drive #1 Direction, Fault, Reset Write-Enable

Copy

Src

Des

Drive #1 Speed Ref New

Drive #1 Speed Ref Retain

Drive #1 Direction New

Drive #1 Direction Retain

Drive #1 Ext Fault New

Drive #1 Ext Fault Retain

Drive #1 Fault Reset New

Drive #1 Fault Reset Retain

Drive #1 Speed Ref Write Enable

Drive #1 Direction, Fault, Reset Write Enable

Drive #1 Speed Direction, Fault, Reset writes finished

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

Rungs 15 & 16 write to the Run Command word if it changes.

Drive #1 Run CMD New
DS301

Drive #1 Run CMD Retain
DS311

Drive #1 Speed Ref Write Enable
C10

Drive #2 Run CMD New

Drive #1 Run CMD Write-Enable
C12

Drive #2 Speed Ref Write Enable
C12

_Port_3_Ready_Flag
SC102

Drive #1 Run CMD Write-Enable
C12

Send(Port3)
MODBUS
Slave ID
1
Modbus Function Code
16
Slave Adder
402332

Master
Drive #1 Run CMD New
DS301

Drive #1 Run CMD Write finished
C14

(set)
(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 writes the new value for the Run Command word to its comparison location so the code can again start watching for changes.

Rung 17

Drive #1 Run CMD Write-Enable

Drive #1 Run CMD Write finished

Copy

Single

Src

Drive #1 Run CMD New

DS301

Des

Drive #1 Run CMD Retain

DS311

Drive #1 Run CMD Write-Enable

RST

Drive #1 Run CMD Write finished

RST

Rungs 18 & 19 select either 30Hz or 60Hz based on C102.

Rung 18

Drive #1 Speed Control 60/30 Hz

Copy

Single

Src

300

Des

Drive #1 Speed Ref New

DS300

Rung 19

Drive #1 Speed Control 60/30 Hz

Copy

Single

Src

600

Des

Drive #1 Speed Ref New

DS300

Rungs 20 & 21 select Run or Stop based on C103.

Rung 20

Drive #1 Run Stop

Copy

Single

Src

1

Des

Drive #1 Run CMD New

DS301

Rung 21

Drive #1 Run Stop

Copy

Single

Src

0

Des

Drive #1 Run CMD New

DS301

(continued next page – CLICK PLC communication program example)
Rungs 22 & 23 select Direction based on C104.

Drive #1 Fwd Rev

22

Copy
Src 1
Des Drive #1 Direction New DS302

Drive #1 Fwd Rev

23

Copy
Src 0
Des Drive #1 Direction New DS302

Rungs 24 & 25 select Ext Fault or no fault based on C105.

Drive #1 Fault

24

Copy
Src 1
Des Drive #1 Ext Fault New DS303

Drive #1 Fault

25

Copy
Src 0
Des Drive #1 Ext Fault New DS303

Rungs 26 & 27 select Fault Reset or no reset based on C106.

Drive #1 Ext Fault Reset

26

Copy
Src 1
Des Drive #1 Fault Reset New DS304

Drive #1 Ext Fault Reset

27

Copy
Src 0
Des Drive #1 Fault Reset New DS304
This program is for illustrational purposes only, and is not intended for a true application.

**The remaining rungs are for Drive #2 communications.**

Status from Drive #2

28

DS20 ≥ 1

Drive #2 Fault Indication

29

C107

Drive #2 Fault

DS20 ≥ 1

Status from Drive #2

30

DS20 ≥ 4

Drive #2 Overload Indication

31

C108

Drive #2 Overload

This rung determines if the Speed, Direction, Ext Fault, or Fault Reset words have changed and need to be written.

Drive #2 Speed Ref New

32

DS320

Drive #2 Speed Ref Retain

DS330

Drive #2 Speed Ref Write Enable

DS323

Drive #2 Overload Indicator

DS322

Drive #2 Ext Fault New

DS323

Drive #2 Fault Reset New

DS324

Drive #2 Run CMD Write Enable

DS312

Drive #2 Speed Ref Write Enable

DS313

(continued next page – CLICK PLC communication program example)
This rung writes the new Speed Reference if it changes.

Drive #2 Direction, Fault, Reset, Write Enable

Drive #2 Speed Ref Write Enable

Send(Port3) MODBUS
Slave ID 2
Modbus Function Code 06
Slave Addr 402331

Master

Drive #2 Speed Ref New

DS320

Drive #2 Direction, Fault, Reset, Write Enable

Drive #2 Speed Ref Write Enable

Send(Port3) MODBUS
Slave ID 2
Modbus Function Code 16
Slave Addr 402331
NO. of Master Addresses 3
Word Swap OFF

Master

Drive #2 Direction New

DS322

Drive #2 Speed, Direction, Fault, Reset, writes finished

(continued next page – CLICK PLC communication program example)
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.

Drive #2 Speed Ref Write Enable  \( \text{RST} \)

Drive #2 Speed, Direction, Fault, Reset, Write-Enable

Drive #2 Speed, Direction, Fault, Reset, writes finished

Drive #1 Speed Ref Write Enable  \( \text{RST} \)

Drive #1 Speed, Direction, Fault, Reset writes finished

Drive #2 Speed, Direction, Fault, Reset Write-Enable

Drive #2 Speed Ref New  \( \text{DS320} \)

Drive #2 Speed Ref Retain  \( \text{DS330} \)

Drive #2 Direction New  \( \text{DS322} \)

Drive #2 Direction Retain  \( \text{DS332} \)

Drive #2 Ext Fault New  \( \text{DS323} \)

Drive #2 Ext Fault Retain  \( \text{DS333} \)

Drive #2 Fault Reset New  \( \text{DS324} \)

Drive #2 Fault Reset Retain  \( \text{DS334} \)

Drive #2 Speed Ref Write-Enable  \( \text{RST} \)

Drive #1 Speed Ref Write Enable  \( \text{RST} \)

Drive #1 Speed, Direction, Fault, Reset, Write-Enable

Drive #1 Speed, Direction, Fault, Reset writes finished

Drive #1 Ext Fault New  \( \text{DS313} \)

Drive #1 Ext Fault Retain  \( \text{DS330} \)

Drive #1 Speed Ref New  \( \text{DS312} \)

Drive #1 Speed Ref Retain  \( \text{DS333} \)

(continued next page – CLICK PLC communication program example)
This program is for illustrational purposes only, and is not intended for a true application.

(Continued from previous page – CLICK PLC Communication Program Example)

Rungs 36 & 37 write to the Run Command word if it changes.

Drive #2 Run CMD New

DS321

Drive #2 Run CMD Retain

DS331

Drive #2 Run CMD Write-Enable

C30

Drive #2 Run CMD Write-Enable

C32

Send(Port3) MODBUS
Slave ID

2

Slave Adder

402332

Master

Drive #2 Run CMD New

DS321

Drive #2 Run CMD Write finished

C34

Copy

Single

Src

Drive #2 Run CMD New

DS321

Des

Drive #2 Run CMD Retain

DS331

Drive #2 Run CMD Write-Enable

C32

Drive #2 Run CMD write finished

C34

(continued next page – CLICK PLC communication program example)
Rungs 39 & 40 select either 30Hz or 60Hz based on C109.

Drive #2 Speed Control 60/30 Hz

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>300</td>
</tr>
<tr>
<td>Des</td>
<td>DS320</td>
</tr>
</tbody>
</table>

Drive #2 Speed Ref Write Enable

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>600</td>
</tr>
<tr>
<td>Des</td>
<td>DS320</td>
</tr>
</tbody>
</table>

Rungs 41 & 42 select Run or Stop based on C110.

Drive #2 Run Stop

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>1</td>
</tr>
<tr>
<td>Des</td>
<td>DS321</td>
</tr>
</tbody>
</table>

Drive #2 Run CMD Write-Enable

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>0</td>
</tr>
<tr>
<td>Des</td>
<td>DS321</td>
</tr>
</tbody>
</table>

Rungs 43 & 44 select Direction based on C111.

Drive #2 Fwd Rev

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>1</td>
</tr>
<tr>
<td>Des</td>
<td>DS322</td>
</tr>
</tbody>
</table>

Drive #2 Direction Write-Enable

<table>
<thead>
<tr>
<th>Copy</th>
<th>Single</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>0</td>
</tr>
<tr>
<td>Des</td>
<td>DS322</td>
</tr>
</tbody>
</table>

This program is for illustrational purposes only, and is not intended for a true application.
This program is for illustrational purposes only, and is not intended for a true application.
**DirectLOGIC Modbus Ladder Programming**

The set up for all of the DirectLOGIC 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 DirectLOGIC 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 illustrative 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.*

---

This rung counts the number of times the PLC attempts to communicate to the drive.

```
1
  SP116
  CT0
```

This rung counts the number of times an attempted communication to the drive fails.

```
2
  SP117
  CT1
```

This rung reads the ‘Status Addresses’ information from the drive. Use this code to test communication to each of your drives before writing more advanced code that polls multiple drives. To target different drives, change the value ‘Kf201’ to ‘Kf202’ for slave 2, ‘Kf203’ for slave 3, etc.

```
3
  SP116
  LD Kf201
  LD K24
  LDA O2000
  RX V20400
```

4

(END)

---

*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.
This rung counts the number of times the PLC attempts to communicate to the drive.

```
1
Port Busy
SP116
CT0
CNT
CT0
K9999
```

This rung counts the number of times an attempted communication to the drive fails.

```
2
Port Comm Fail
SP117
CT1
CNT
CT1
K9999
```

This rung acts as an alternator, allowing the following logic to alternate between communicating to slave 1 or slave 2. If there were additional slaves, the 'K' number for the counter would simply be increased to match the number of slaves in the system.

```
3
Port Busy
SP116
CT2
CNT
CT2
K2
```

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

```
4
Port Busy
SP116
CTA2
K0
C10
Drive #1
Speed Ref
Write Enable
MRX
CPU/DCM Slot:  CPU
Port Number:  K2
Slave Address:  K1
Function Code:  03 - Read Holding Registers
Start Slave Memory Address:  K48449
Start Master Memory Address:  V2000
Number of Elements:  K12
Modbus Data type:  584/984 Mode
Exception Response Buffer:  V5000
```

(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)
This rung checks to see if it is time to communicate to slave 2, and also if there are no current write requests to that drive. If not, it reads data from slave 2.

Port Busy
SP116 = CTA2

Drive #2
Speed Ref
Write Enable
K1
C40

Drive #2
Direction, Fault, Reset
Write Enable
C41
C42

*** The following 21 rungs (6–26) are for slave 1 communications control. ***

This rung turns on C1 if there is a fault in drive #1.

Drive #1 Fault
C1

This rung allows a switch on input X1 to reset the C bit used to indicate a drive #1 fault.

Drive #1 Fault Indication
Reset
X1

This rung turns on C2 if drive #1 has an overload fault.

Drive #1 OL
C2

This rung allows a switch on input X2 to reset the overload fault bit C2.

Drive #1 Overload Indication
Reset
X2
This rung checks to see if the drive Speed, Direction, External Fault, or Fault Reset conditions have been changed in the local program, and need to be written to drive #1.

Drive #1 Speed Ref New
V3000

Drive #1 Speed Ref Retain
V3010

Drive #1 Direction New
V3002

Drive #1 Direction Retain
V3012

Drive #1 External Fault New
V3003

Drive #1 Ext Fault Retain
V3013

Drive #1 Fault Reset New
V3004

Drive #1 Fault Reset Retain
V3014

This rung writes the new Speed Reference if it changes.

Drive #1 Speed Ref
Write Enable
C10

Drive #1 Direction, Fault, Reset
Write Enable
C11

This rung writes the Direction, Ext Fault, and Fault Reset words if any of them changes.

Drive #1 Speed Ref
Write Enable
C10

Drive #1 Direction, Fault, Reset
Write Enable
C11

(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.

<table>
<thead>
<tr>
<th>Rung</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>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.</td>
</tr>
</tbody>
</table>

- **Drive #1 Speed Control**
  - Rungs 17 & 18 select either 30Hz or 60Hz based on X3.
  - 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.

- **Drive #2 Speed Control**
  - This rung allows a switch on input X1 to reset the C bit used to indicate a drive #1 fault.
  - This rung counts the number of times the PLC attempts to communicate to the drive.
  - This rung counts the number of times an attempted communication to the drive fails.
  - This rung checks to see if the drive Speed, Direction, External Fault, or Fault Reset.
This rung writes the new value for the Run Command word to its comparison location so the code can again start watching for changes.
(continuing 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.

Rungs 17 & 18 select either 30Hz or 60Hz based on X3.
Drive #1 Speed Control
bit 60/30Hz

Rungs 19 & 20 select Run or Stop based on X5.
Drive #1 Run/Stop

(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)
Rungs 21 & 22 select Direction based on X6.
Drive #1 Forward/Reverse

Rungs 23 & 24 select Ext Fault or no fault based on X7.
Drive #1 Ext Fault

Rungs 25 & 26 select Fault Reset or no reset based on X10.
Drive #1 Ext Fault Reset
*** The following 21 rungs (27–47) are for slave 2 communications control. ***

This rung turns on C31 if there is a fault in drive #2.

\[ \text{V2020} \geq K1 \]  
\[ \text{Drive #2 Fault} \]
\[ \text{C31 (SET)} \]

This rung allows a switch on input X21 to reset the C bit used to indicate drive #2 fault.

\[ \text{Drive #2 Fault Indication} \]
\[ \text{Reset X21} \]
\[ \text{Drive #2 Fault} \]
\[ \text{C31 (RST)} \]

This rung turns on C32 if drive #2 has an overload fault.

\[ \text{V2020} = K4 \]
\[ \text{Drive #2 OL} \]
\[ \text{C32 (SET)} \]

This rung allows a switch on input X22 to reset the overload fault bit C32.

\[ \text{Drive #2 Overload Indication} \]
\[ \text{Reset X22} \]
\[ \text{Drive #2 OL} \]
\[ \text{C32 (RST)} \]

This rung checks to see if the drive Speed, Direction, External Fault, or Fault Reset conditions have been changed in the local program, and need to be written to drive #2.

\[ \text{Drive #2 Speed Ref} \]
\[ \text{Write Enable} \]
\[ \text{C40 (SET)} \]

(continued next page – DL MRX/MWX communication program example 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 Speed Reference if it changes.

Port Busy Speed Ref Write Enable C40 C41

Drive #2 Drive #2

This rung writes the Direction, Ext Fault, and Fault Reset words if any of them changes.

Port Busy Speed Ref Write Enable C40 C41

Drive #2 Drive #2

Drive #2 Speed, Direction, Fault, Reset
Write Enabled

C41 SET

Drive #2, Speed,
Direction, Fault, Reset
Writes Finished

C43 SET

(continued next page – DL MRX/MWX communication program example 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.

Drive #2
Speed Ref
Write Enable
C40

Drive #2
Direction, Fault, Reset
Write Enable
C41

Drive #2, Speed, Direction, Fault, Reset
Write Finished
C43

LD Drive #2 Speed Ref New V3020
OUT Drive #2 Speed Ref Retain V3030
LD Drive #2 Direction New V3022
OUT Drive #2 Direction Retain V3032
LD Drive #2 Ext Fault New V3023
OUT Drive #2 Ext Fault Retain V3033
LD Drive #2 Fault Reset New V3024
OUT Drive #2 Fault Reset Retain V3034

Drive #2 Speed Ref
Write Enable
C40
(RST)

Drive #2 Direction, Fault, Reset
Write Enable
C41
(RST)

Drive #2 Speed, Direction, Fault Reset
Write Finished
C43
(RST)

(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)
Rungs 35 & 36 write to the Run Command word if it changes.

Drive #2  Run CMD  Run CMD
New  Retain  Write Enable
V3021  V3031  C40

35

Port Busy
SP116
C42

36

Drive #2  Run CMD  Write Enable
V3031

This rung writes the new value for the Run Command word to its comparison location so the code can again start watching for changes.

Drive #2  Run CMD  Write Enable
C42

37

Drive #2  Run CMD  Write Enable
C42

(continued next page – DL MRX/MWX communication program example for DL06 & D2-260 PLCs)
Rungs 38 & 39 select either 30Hz or 60Hz based on X23.
Drive #2 Speed Control
   bit 60/30Hz
   X23

Rungs 40 & 41 select Run or Stop based on X25.
Drive #2 Run/Stop
   X25

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

**Drive #2 Speed Control**

Rungs 14 & 15 write to the Run Command word if it changes.

Rungs 25 & 26 select Fault Reset or no reset based on X10.

This rung counts the number of times an attempted communication to the drive fails.

*** The following 21 rungs (6–26) are for slave 1 communications control. ***

This rung turns on C1 if there is a fault in drive #1.

This rung turns on C2 if drive #1 has an overload fault.

This rung writes the new values for Speed Ref, Direction, Ext Fault, and Fault Reset.

Drive #2 Forward/Reverse

Rungs 44 & 45 select Ext Fault or no fault based on X27.

Rungs 46 & 47 select Fault Reset or no reset based on X30.

Drive #2 Forward/Reverse

Rungs 42 & 43 select Direction based on X26.

Run #2 Forward/Reverse

Drive #2 Ext Fault

Rungs 44 & 45 select Ext Fault or no fault based on X27.

Rungs 46 & 47 select Fault Reset or no reset based on X30.

Drive #2 Ext Fault

Rungs 46 & 47 select Fault Reset or no reset based on X30.
Chapter 5: Communications

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.

This rung counts the number of times the PLC attempts to communicate to the drive.

<table>
<thead>
<tr>
<th>Port Busy</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP116</td>
<td>K9999</td>
</tr>
</tbody>
</table>

This rung counts the number of times an attempted communication to the drive fails.

<table>
<thead>
<tr>
<th>Port Comm Fail</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP117</td>
<td>K9999</td>
</tr>
</tbody>
</table>

This rung acts as an alternator, allowing the following logic to alternate between communicating to slave 1 or slave 2. If there were additional slaves, the 'K' number for the counter would simply be increased to match the number of slaves in the system.

<table>
<thead>
<tr>
<th>Port Busy</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP116</td>
<td>K2</td>
</tr>
</tbody>
</table>

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

Port Busy | CTA2 = K0 C10 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SP116</td>
<td>Write Enable</td>
</tr>
</tbody>
</table>

Drive #1 Speed Ref, Direction, Fault, Reset

<table>
<thead>
<tr>
<th>LD</th>
<th>K201</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>24</td>
</tr>
<tr>
<td>RX V20400</td>
<td></td>
</tr>
</tbody>
</table>

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

Port Busy | CTA2 = K1 C40 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SP116</td>
<td>Write Enable</td>
</tr>
</tbody>
</table>

Drive #2 Speed Ref, Direction, Fault, Reset

<table>
<thead>
<tr>
<th>LD</th>
<th>K202</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>24</td>
</tr>
<tr>
<td>RX V20400</td>
<td></td>
</tr>
</tbody>
</table>

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.

*** The following 21 rungs (6–26) are for slave 1 communications control. ***

This rung turns on C1 if there is a fault in drive #1.

```
6
V2000 K1
```

![Diagram of rung 6](image)

This rung allows a switch on input X1 to reset the C bit used to indicate a drive #1 fault.

```
7
Reset
Drive #1 Fault Indication
X1
```

![Diagram of rung 7](image)

This rung turns on C2 if drive #1 has an overload fault.

```
8
V2000 K4
```

![Diagram of rung 8](image)

This rung allows a switch on input X2 to reset the overload fault bit C2.

```
9
Reset
Drive #1 Overload Indication
X2
```

![Diagram of rung 9](image)

This rung checks to see if the drive Speed, Direction, External Fault, or Fault Reset conditions have been changed in the local program, and need to be written to drive #1.

```
10
Drive #1 Speed Ref New V3000
Drive #1 Speed Ref Retain V3010
Run CMD Retain C10
```

![Diagram of rung 10](image)

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.

11

Port Busy
SP116              

This rung writes the new Speed Reference if it changes.

Drive #1 Speed Ref Write Enable
C10

Drive #1 Direction, Fault, Reset Write Enable
C11

12

Port Busy
SP116              

This rung writes the Direction, Ext Fault, and Fault Reset words if any of them changes.

Drive #1 Speed Ref Write Enable
C10

Drive #1 Direction, Fault, Reset Write Enable
C11

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

13

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

Rungs 14 & 15 write to the Run Command word if it changes.

This rung writes the new value for the Run Command word to its comparison location so the code can again start watching for changes.

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.

<table>
<thead>
<tr>
<th>Rung 17</th>
<th>Rung 18</th>
<th>Rung 19</th>
<th>Rung 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rungs 17 &amp; 18 select either 30Hz or 60Hz based on X3.</td>
<td>Drive #1 Speed Control</td>
<td>Drive #1 Run/Stop</td>
<td>Drive #1 Run/Stop</td>
</tr>
<tr>
<td>Drive #1 Speed Control</td>
<td>bit 60/30Hz</td>
<td>X5</td>
<td>X5</td>
</tr>
<tr>
<td>X3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
LD K300
BIN
OUT Drive #1 Speed Ref New V3000
```

```
LD K600
BIN
OUT Drive #1 Speed Ref New V3000
```

```
Rungs 19 & 20 select Run or Stop based on X5.
Drive #1 Run/Stop
```

```
LD K1
OUT Drive #1 Run CMD New V3001
```

```
LD K0
OUT Drive #1 Run CMD New V3001
```

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.

Rungs 21 & 22 select Direction based on X6.
Drive #1 Forward/Reverse
X6

21

Rungs 23 & 24 select Ext Fault or no fault based on X7.
Drive #1 Ext Fault
X7

23

Rungs 25 & 26 select Fault Reset or no reset based on X10.
Drive #1 Ext Fault Reset
X10

25

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
*** The following 21 rungs (27–47) are for slave 2 communications control. ***

This rung turns on C31 if there is a fault in drive #2.

27

\[ V2020 \geq K1 \]

This rung allows a switch on input X21 to reset the C bit used to indicate drive #2 fault.

\[ \text{Drive #2 Fault Indication} \]

\[ \text{Reset} \]

\[ X21 \]

\[ \text{Drive #2 Fault} \]

\[ C31 \]

\( \text{SET} \) 

This rung turns on C32 if drive #2 has an overload fault.

29

\[ V2020 = K4 \]

This rung allows a switch on input X22 to reset the overload fault bit C32.

\[ \text{Drive #2 Overload Indication} \]

\[ \text{Reset} \]

\[ X22 \]

\[ \text{Drive #2 OL} \]

\[ C32 \]

\( \text{SET} \) 

This rung checks to see if the drive Speed, Direction, External Fault, or Fault Reset conditions have been changed in the local program, and need to be written to drive #2.

31

\[ \text{Drive #2 Speed Ref New} \]

\[ V3020 \]

\[ \text{Speed Ref Retain} \]

\[ V3030 \]

\[ \text{Write Enable} \]

\[ C40 \]

\[ \text{Drive #2} \]

\[ \text{Direction New} \]

\[ V3022 \]

\[ \text{Direction Retain} \]

\[ V3032 \]

\[ \text{Drive #2} \]

\[ \text{External Fault New} \]

\[ V3023 \]

\[ \text{External Fault Retain} \]

\[ V3033 \]

\[ \text{Drive #2} \]

\[ \text{Fault Reset New} \]

\[ V3024 \]

\[ \text{Fault Reset Retain} \]

\[ V3034 \]

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
Chapter 5: Communications

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

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

This rung writes the new Speed Reference if it changes.

Port Busy
SP116
Write Enable
C40

Drive #2
Speed Ref
Write Enable
C41

LD  Kf202
LD  K2
LDA  O3000
WX  V4432

Drive #2
Direction, Fault, Reset
Write Enabled
C41  (SET )

This rung writes the Direction, Ext Fault, and Fault Reset words if any of them changes.

Port Busy
SP116
Write Enable
C40

Drive #2
Direction, Fault, Reset
Write Enable
C41

LD  Kf202
LD  K6
LDA  O3002
WX  V4434

Drive #2, Speed,
Direction, Fault, Reset
Writes Finished
C43  (SET )

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

Drive #2
Speed Ref
Write Enable
C40

Drive #2
Direction, Fault, Reset
Write Enable
C41

Drive #2, Speed,
Direction, Fault, Reset
Writes Finished
C43

LD Drive #2 Speed Ref New V3020

OUT Drive #2 Speed Ref Retain V3030

LD Drive #2 Direction New V3022

OUT Drive #2 Direction Retain V3032

LD Drive #2 Ext Fault New V3023

OUT Drive #2 Ext Fault Retain V3033

LD Drive #2 Fault Reset New V3024

OUT Drive #2 Fault Reset Retain V3034

Drive #2
Speed Ref
Write Enable
C40

( )RST

Drive #2
Direction,
Fault Reset
Write Enable
C41

( )RST

Drive #2
Speed, Direction,
Fault Reset
Writes Finished
C43

( )RST

(continued next page – DL RX/WX Communication Program example for DL05, D2-250(-1), D4-450 PLCs)
This program is for illustrational purposes only, and is not intended for a true application.
Rungs 38 & 39 select either 30Hz or 60Hz based on X23.
Drive #2 Speed Control
bit 60/30Hz
X23

Rungs 40 & 41 select Run or Stop based on X25.
Drive #2 Run/Stop
X25

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

- **Rungs 42 & 43 select Direction based on X26.**
  - Drive #2 Forward/Reverse
    - X26
    - LD K1
    - OUT Drive #2 Direction New V3022

- **Rungs 44 & 45 select Ext Fault or no fault based on X27.**
  - Drive #2 Ext Fault
    - X27
    - LD K1
    - OUT Drive #2 Ext Fault New V3023

- **Rungs 46 & 47 select Fault Reset or no reset based on X30.**
  - Drive #2 Ext Fault Reset
    - X30
    - LD K1
    - OUT Drive #2 Ext Fault Reset New V3024

- **Rungs 21 & 22 select Direction based on X6.**
- **Rungs 23 & 24 select Ext Fault or no fault based on X7.**
- **Rungs 25 & 26 select Fault Reset or no reset based on X10.**
- **Rungs 40 & 41 select Run or Stop based on X25.**
- **Rungs 44 & 45 select Ext Fault or no fault based on X27.**
- **Rungs 46 & 47 select Fault Reset or no reset based on X30.**
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**

<table>
<thead>
<tr>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+17V</td>
<td>GND</td>
<td>SG-</td>
<td>SG+</td>
<td>nc</td>
<td>reserved</td>
</tr>
</tbody>
</table>

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
**Chapter 5: Communications**

**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)

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Stop bit</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10-bit character frame</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Even parity</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10-bit character frame</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Odd parity</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10-bit character frame</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Stop bit</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11-bit character frame</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Even parity</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11-bit character frame</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Start bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Odd parity</th>
<th>Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit character</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11-bit character frame</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Communication Protocol

#### ASCII Mode:

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

#### RTU Mode:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>A silent interval of more than 10 ms</td>
</tr>
<tr>
<td>ADR</td>
<td>Communication Address: 8-bit address</td>
</tr>
<tr>
<td>CMD</td>
<td>Command Code: 8-bit command</td>
</tr>
<tr>
<td>DATA (n-1)</td>
<td>Contents of data: n x 8-bit data, n ≤ 25</td>
</tr>
<tr>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>CRC CHK Low</td>
<td>CRC check sum: 16-bit check sum consists of 2 8-bit characters</td>
</tr>
<tr>
<td>CRC CHK High</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>A silent interval of more than 10 ms</td>
</tr>
</tbody>
</table>

#### 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:**

<table>
<thead>
<tr>
<th>Command Message</th>
<th>Response Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>STX <code>:</code></td>
</tr>
<tr>
<td>ADR 1</td>
<td>ADR 1</td>
</tr>
<tr>
<td>ADR 0</td>
<td>ADR 0</td>
</tr>
<tr>
<td>CMD 1</td>
<td>CMD 1</td>
</tr>
<tr>
<td>CMD 0</td>
<td>CMD 0</td>
</tr>
<tr>
<td><code>0</code></td>
<td><code>0</code></td>
</tr>
<tr>
<td><code>1</code></td>
<td><code>1</code></td>
</tr>
<tr>
<td><code>2</code></td>
<td><code>2</code></td>
</tr>
<tr>
<td><code>3</code></td>
<td><code>3</code></td>
</tr>
<tr>
<td>Number of data</td>
<td>Number of data</td>
</tr>
<tr>
<td>(Count by byte)</td>
<td>(Count by byte)</td>
</tr>
<tr>
<td><code>0</code></td>
<td><code>0</code></td>
</tr>
<tr>
<td><code>1</code></td>
<td><code>1</code></td>
</tr>
<tr>
<td><code>2</code></td>
<td><code>2</code></td>
</tr>
<tr>
<td><code>3</code></td>
<td><code>3</code></td>
</tr>
<tr>
<td>LRC CHK 1</td>
<td>LRC CHK 1</td>
</tr>
<tr>
<td>LRC CHK 0</td>
<td>LRC CHK 0</td>
</tr>
<tr>
<td>CR</td>
<td>CR</td>
</tr>
<tr>
<td>LF</td>
<td>LF</td>
</tr>
</tbody>
</table>

**RTU mode:**

<table>
<thead>
<tr>
<th>Command Message</th>
<th>Response Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR</td>
<td>ADR</td>
</tr>
<tr>
<td>CMD</td>
<td>CMD</td>
</tr>
<tr>
<td>Starting data</td>
<td>Starting data</td>
</tr>
<tr>
<td>address</td>
<td>address</td>
</tr>
<tr>
<td><code>01H</code></td>
<td><code>01H</code></td>
</tr>
<tr>
<td><code>03H</code></td>
<td><code>03H</code></td>
</tr>
<tr>
<td><code>21H</code></td>
<td><code>21H</code></td>
</tr>
<tr>
<td><code>02H</code></td>
<td><code>02H</code></td>
</tr>
<tr>
<td>Number of data</td>
<td>Number of data</td>
</tr>
<tr>
<td>(Count by byte)</td>
<td>(Count by byte)</td>
</tr>
<tr>
<td><code>00H</code></td>
<td><code>00H</code></td>
</tr>
<tr>
<td><code>02H</code></td>
<td><code>02H</code></td>
</tr>
<tr>
<td>Content of data</td>
<td>Content of data</td>
</tr>
<tr>
<td>address</td>
<td>address</td>
</tr>
<tr>
<td><code>2102H</code></td>
<td><code>2102H</code></td>
</tr>
<tr>
<td><code>2103H</code></td>
<td><code>2103H</code></td>
</tr>
<tr>
<td>CRC CHK Low</td>
<td>CRC CHK Low</td>
</tr>
<tr>
<td>CRC CHK High</td>
<td>CRC CHK High</td>
</tr>
<tr>
<td>06H</td>
<td>06H</td>
</tr>
<tr>
<td>0FH</td>
<td>0FH</td>
</tr>
<tr>
<td>07H</td>
<td>07H</td>
</tr>
<tr>
<td>0E</td>
<td>0E</td>
</tr>
<tr>
<td>CRC CHK Low</td>
<td>CRC CHK Low</td>
</tr>
<tr>
<td>CRC CHK High</td>
<td>CRC CHK High</td>
</tr>
<tr>
<td>FEH</td>
<td>FEH</td>
</tr>
<tr>
<td>5CH</td>
<td>5CH</td>
</tr>
</tbody>
</table>
Command code: 06H, write 1 word
For example, writing 6000(1770H) to address 0100H of the AC drive with address 01H.

**ASCII mode:**

<table>
<thead>
<tr>
<th>Command Message</th>
<th>Response Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>STX</td>
</tr>
<tr>
<td>ADR 1</td>
<td>ADR 1</td>
</tr>
<tr>
<td>ADR 0</td>
<td>ADR 0</td>
</tr>
<tr>
<td>CMD 1</td>
<td>CMD 1</td>
</tr>
<tr>
<td>CMD 0</td>
<td>CMD 0</td>
</tr>
<tr>
<td>Data Address</td>
<td></td>
</tr>
<tr>
<td>LRC CHK 1</td>
<td>LRC CHK 1</td>
</tr>
<tr>
<td>LRC CHK 0</td>
<td>LRC CHK 0</td>
</tr>
<tr>
<td>END 1</td>
<td>END 1</td>
</tr>
<tr>
<td>END 0</td>
<td>END 0</td>
</tr>
</tbody>
</table>

**RTU mode:**

This is an example of using function code 16 for writing to multiple registers.
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>
<thead>
<tr>
<th>Command Message</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>‘ ‘</td>
</tr>
<tr>
<td>ADR 1</td>
<td>‘0’</td>
</tr>
<tr>
<td>ADR 0</td>
<td>‘1’</td>
</tr>
<tr>
<td>CMD 1</td>
<td>‘0’</td>
</tr>
<tr>
<td>CMD 0</td>
<td>‘3’</td>
</tr>
<tr>
<td>Starting data address</td>
<td>01H+03H+04H+01H+00H+01H=0AH; the 2’s complement negation of 0AH is F6H.</td>
</tr>
<tr>
<td>Number of data (Count by word)</td>
<td>0</td>
</tr>
<tr>
<td>LRC CHK 1</td>
<td>‘F’</td>
</tr>
<tr>
<td>LRC CHK 0</td>
<td>‘6’</td>
</tr>
<tr>
<td>END 1</td>
<td>CR</td>
</tr>
<tr>
<td>END 0</td>
<td>LF</td>
</tr>
</tbody>
</table>

**RTU Mode:**

<table>
<thead>
<tr>
<th>Response Message</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR</td>
<td>01H</td>
</tr>
<tr>
<td>CMD</td>
<td>03H</td>
</tr>
<tr>
<td>Starting data address</td>
<td>21H</td>
</tr>
<tr>
<td>Number of data (Count by word)</td>
<td>00H</td>
</tr>
<tr>
<td>CRC CHK Low</td>
<td>6FH</td>
</tr>
<tr>
<td>CRC CHK High</td>
<td>F7H</td>
</tr>
</tbody>
</table>
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.
Chapter 5: Communications

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:**

<table>
<thead>
<tr>
<th>Request</th>
<th>Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX = Node Address (1 Char)</td>
<td>XX = Node Address (1 Char)</td>
</tr>
<tr>
<td>03 = Function Code (1 Char)</td>
<td>03 = Function Code (1 Char)</td>
</tr>
<tr>
<td>XXXX = Starting Address to read (2 Chars)</td>
<td>XX = Byte count of data being sent from Slave (1 Char)</td>
</tr>
<tr>
<td>XXXX = Number of Registers to read (2 Chars)</td>
<td>XXXX... = Depends upon Request (2 Chars per Register requested)</td>
</tr>
<tr>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
</tr>
</tbody>
</table>

**Format of “Write Multiple Registers” Messages:**

<table>
<thead>
<tr>
<th>Request</th>
<th>Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX = Node Address (1 Char)</td>
<td>XX = Node Address (1 Char)</td>
</tr>
<tr>
<td>10 = Function Code (Hex format) (1 Char)</td>
<td>10 = Function Code (Hex format) (1 Char)</td>
</tr>
<tr>
<td>XXXX = Starting Address to write to (2 Chars)</td>
<td>XXXX = Starting Address to write to (2 Chars)</td>
</tr>
<tr>
<td>XXXX = Number of Registers to write to (2 Chars)</td>
<td>XXXX = Number of Registers to write to (2 Chars)</td>
</tr>
<tr>
<td>XX = Number of bytes of data to write (1 Char)</td>
<td>XXXX = Number of Registers to write to (2 Chars)</td>
</tr>
<tr>
<td>XXXX... = Depends upon Request (2 Chars per Register requested)</td>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
</tr>
<tr>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
<td></td>
</tr>
</tbody>
</table>

**Format of “Write Single Register” Messages:**

<table>
<thead>
<tr>
<th>Request</th>
<th>Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX = Node Address (1 Char)</td>
<td>XX = Node Address (1 Char)</td>
</tr>
<tr>
<td>06 = Function Code (1 Char)</td>
<td>06 = Function Code (1 Char)</td>
</tr>
<tr>
<td>XXXX = Register to Write to (2 Chars)</td>
<td>XXXX = Register to Write to (2 Chars)</td>
</tr>
<tr>
<td>XXXX = Data to Write (2 Chars)</td>
<td>XXXX = Data to Write (2 Chars)</td>
</tr>
<tr>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
<td>XXXX = 16 Bit CRC (2 Chars)</td>
</tr>
</tbody>
</table>

**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**

![Diagram showing delays in the communication process between a Modbus Master (typically a PLC) and a GS AC Drive.](image)

- $t_1$ = 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.