Modbus for the DG Series Generators

Models 004311-0
004313-0
INTRODUCTION

The DG series of generators are controlled by “The F Panel.” This module has both an RS232 and an RS485 communication port that can be used to share data. Both ports provide the same function, such that either port is interchangeable in terms of the data and protocol, only the physical signals are different. The RS232 port has facilities to allow the connection of a modem for remote monitoring. Depending upon the particular installation, one or both of these ports may be available for communication with the generator. Some installations may use both ports, for example, a multi-generator site with peak shaving and each generator fitted with a remote annunciator. If you have a linked multi-generator site, then data about each generator can be obtained from the separate master controllers in the same protocol. Examples of connections are described later in this manual.

The method used to communicate with all DG series controllers is the industry standard Modbus protocol. Details about this protocol may be obtained from the following website:

http://www.modicon.com/techpubs/intr7.html

Generac produces a software application known as “GENLINK”, which is a PC based program designed to allow communication with individual generators and linked DG series generators via the master controller. The physical connection between GENLINK and the generators may be a direct serial link or it can be via a modem on the RS232 port.

GENLINK uses the Modbus protocol to continuously monitor and display the state of each generator and master controller. Data is displayed in the form of mimic diagrams showing the generator’s key data (such as voltage, current, power, etc.). For multiple set installations, an overall mimic diagram is displayed. This overall mimic will show the power layout indicating which sections are live and the positions of the transfer switches. GENLINK can also be used to change the operating parameters of the generators and the master controller.

It is not necessary to use GENLINK as the basis for communication with the DG series. Since Modbus has been chosen as the standard protocol, any piece of equipment capable of being programmed as a Modbus master can be used for data exchange.

MODBUS PROTOCOL

Modbus controllers can be set up to communicate in one of two modes — ASCII or RTU. The DG series controllers will only work in RTU modes (where communications are done in binary — not ASCII). The speed of communication (baud rate) is programmable, but limited to be either 9600 or 4800 baud.

The DG controllers act as slaves in a master/slave configuration and depending on the installation, many slaves can be linked together to one master. In order to differentiate units from each other, each unit is assigned a unique address (even if there is only one unit). These addresses are programmed into the slave units from their front panels.

Data is stored and accessed from uniquely defined registers within the F panel. Each register contains specific data such as RPM, HZ, and VOLTS, etc. and can be accessed or changed as defined by the Modbus protocol.

GENERATOR (SLAVE) ADDRESSING

In a single generator installation, it is not necessary to program an address into the controller. This is because each controller will always respond to a “universal address”, thus allowing a GENLINK user to link to any site. Security is performed not by address, but by a password. The universal address is decimal 250. The unique address will default to 1.

In a multiple generator installation, each generator must have a unique address. This can be entered from the front panel under the engine parameters menu. The master controller must also have an address but it will default to 1 which is sufficient for most purposes. If the master controller is linked to a building network, you may need to alter it’s address to avoid conflicts with other equipment connected to the same network. The master controller also needs to be programmed with both the number of generator sets connected to it and their addresses.

SETTING UP COMMUNICATION PARAMETERS

The speed of communication (baud rate), number of stop bits and parity are all programmable from the front panel of the controllers. These should be set up to match each other and the Modbus master (e.g. GENLINK).

We suggest: 4800 baud, 8 data bits, NO parity, 2 stop bits.

This is the default setting for both the “F” panel and the master controller RS485 port. The RS232 port is set as: 9600 baud, 8 data bits, NO parity, 1 stop bit.

CONNECTIONS TO THE “F” PANEL

In order to make physical connections to the ports, it is necessary to open the generator’s enclosure.

--- DANGER ---

OBSERVE ALL SAFETY PRECAUTIONS AS DESCRIBED IN THE GENERATOR MANUAL. ENSURE THAT THE UTILITY VOLTAGE HAS BEEN DISCONNECTED FROM THE UNIT AND THAT THE BATTERY AND THE UTILITY BATTERY CHARGER HAVE BEEN DISCONNECTED. THE EMERGENCY STOP SWITCH SHOULD BE PRESSSED AND THE FRONT PANEL KEY SWITCH SHOULD BE IN THE “OFF” POSITION.
Locate the transfer switch enclosure below or opposite the voltmeter and ammeter at the back of the generator. Remove the cover plate. The communications terminal strip is located on the left hand wall.

The RS485 connections are to the following wires:
- Wire# 392 — RS485+
- Wire# 393 — RS485 –

The RS232 connections are to the following wires:
- Wire# 387 — RS232 RX
- Wire# 388 — RS232 TX
- Wire# 389 — RS232 COMMON

Connection should be via twisted pair shielded cable.

Cable size:
- 24 AWG 0-250 feet
- 18 AWG 0-1200 feet

**CONNECTIONS TO THE MASTER CONTROLLER**

In a multiple generator application, there is a master controller. These are either a peak shave module or an island mode module. There is also the option of having a building management control module. By incorporating a building management control module into the system, it now serves as the master controller in the modbus network, in line with the multiple generators. A RS232 or multi-drop RS485 communicates between the two controllers. The building management control module is a slave in this connection, but is still the master controller over the multiple generator connection. This connection allows interaction between both the master and any of the multiple generators on an independent basis.

Connection to the Power Manager Controller is done by connecting the RS 485 wires to TB2-RS485 #2 positive (+) and TB2-RS485 #2 negative (−).

**STARTING AND STOPPING USING MODBUS COMMANDS**

The F panel can be started and stopped using the Modbus protocol. In order to do this, the key switch must be in the “Auto” position and there must be a good utility supply (otherwise the generator will run in standby mode).

To run the generator without paralleling, the mb_start variable (Modbus register address 0026h) should be set to 0001h. This will start the generator but will not connect the generator to the utility. Setting this variable back to zero will stop the generator again.

To run the generator in parallel using the power and power factor settings on the generator front panel, first make sure that the screen_en variable (Modbus register address 0132h high byte) is set to 0001h. This can also be set from the front panel. Start the generator by setting the mb_start variable to 1 as above and then set the parallel_en variable to 1 (this is the high byte of Modbus register 002a — in other words set this register to 0100h). This will close the transfer switch once the generator has synchronized with the utility, and use the power setting from the front panel display. Note that this power setting (and the power factor setting) can also be changed using Modbus (power setting is register 012bh and power factor is 012ah). These settings are written into EEPROM, meaning that they are retained even after the generator stops. However, it is not advisable to change these values too frequently as the EEPROM has a limited number of write cycles. To disconnect, set the mb_start variable back to zero which will cause the generator to disconnect from the utility and run for the preset cooldown period before stopping.

If the screen_en variable is set to zero, the F panel will use the analog power input instead of the front panel setting. The power factor is still set from the front panel.

The generator can also be run in peak-shave mode by setting the mb_start variable to 0002h instead of 0001h. This will start the generator and automatically parallel. There is no need to set parallel_en in this mode. However, instead of using the front panel settings for power and power factor, it will use the values in ps_kw (Modbus register address 0027h) and ps_pf (Modbus register address 0028h). These values are stored in RAM which means that they are not retained if the F panel is powered down, but they can be written to more frequently than the EEPROM values.

**REGISTER ADDRESSES AND THEIR DATA**

Although data is accessed as 16 bit registers, some data is stored as a byte and therefore one register access will bring back 2 “lots” of data. The following table shows the locations of the available data and is correct as of 8-16-00. The locations should not change but may be added to in the future. The table is in the form of a “C” program header file so you can see which variables

---

Generac reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Generac does not assume any liability arising out of the application or use of this product.
are in what format (signed, unsigned, word or byte). Some variables are accessible for both read and write while others are read only variables. Generac accepts no responsibility for operation as a result of changing these variables.

Editor’s Notes:
Variables with a _l extension are usually the low level flags for a warning. (Non zero flags signify a warning)
Variables with a _h extension are usually the high level flags for a warning. (Non zero flags signify a warning)
Variables with a _s extension are the shutdown flags. (Non zero flags signify a warning)

### Variable declaration

Alarm status variables set if alarm is present: READ ONLY, Stored in RAM

<table>
<thead>
<tr>
<th>Variable</th>
<th>Modbus address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>oil_press_1</td>
<td>0000 Hi low oil pressure warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>oil_press_s</td>
<td>0000 Lo low oil pressure shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>coolant_temp_h</td>
<td>0001 Hi high coolant temperature warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>coolant_temp_s</td>
<td>0001 Lo high coolant temperature shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>coolant_temp_l</td>
<td>0002 Hi low coolant temperature warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>oil_temp_h</td>
<td>0002 Lo high oil temperature warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>oil_temp_s</td>
<td>0003 Hi high oil temperature shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>batt_volts_l</td>
<td>0003 Lo low battery voltage warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>batt_volts_h</td>
<td>0004 Hi high battery voltage warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>rpm_s</td>
<td>0004 Lo overspeed (high rpm) shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>rpm_l</td>
<td>0005 Hi underspeed (low rpm) warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>gen_volts_h</td>
<td>0005 Lo overvoltage shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>gen_volts_l</td>
<td>0006 Hi undervoltage shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>freq_h</td>
<td>0006 Lo over frequency shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>freq_l</td>
<td>0007 Hi under frequency shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>fuel_level_h</td>
<td>0007 Lo high fuel level warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>fuel_level_l</td>
<td>0008 Hi low fuel level warning</td>
</tr>
<tr>
<td>unsigned char</td>
<td>fuel_level_s</td>
<td>0008 Lo low fuel level shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>inter_comm_f</td>
<td>0009 Hi internal error shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>fail_start_s</td>
<td>0009 Lo over-crank shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>water_level_s</td>
<td>000a Hi coolant level sensor failed</td>
</tr>
<tr>
<td>unsigned char</td>
<td>rpm_sens_s</td>
<td>000a Lo rpm sensor failed</td>
</tr>
<tr>
<td>unsigned char</td>
<td>start_inhibit_n</td>
<td>000b Hi start inhibited due to oil press.</td>
</tr>
<tr>
<td>unsigned char</td>
<td>emstop_s</td>
<td>000b Lo emergency stop shutdown</td>
</tr>
<tr>
<td>unsigned char</td>
<td>oil_press_f</td>
<td>000c Hi oil pressure sensor failed</td>
</tr>
<tr>
<td>unsigned char</td>
<td>oil_temp_f</td>
<td>000c Lo oil temperature sensor failed</td>
</tr>
<tr>
<td>unsigned char</td>
<td>coolant_temp_f</td>
<td>000d Hi coolant temperature sensor failed</td>
</tr>
<tr>
<td>unsigned char</td>
<td>*engine_run_status</td>
<td>000d Lo</td>
</tr>
<tr>
<td>unsigned char</td>
<td>*engine_running</td>
<td>000e Hi shows if engine running</td>
</tr>
<tr>
<td>unsigned char</td>
<td>system_status</td>
<td>000e Lo</td>
</tr>
</tbody>
</table>

*See notes on page 5.*
### Variable declaration

#### Analog Values

```
<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>signed int</td>
<td>oil_press</td>
<td>000f</td>
</tr>
<tr>
<td>unsigned int</td>
<td>rpm</td>
<td>0010</td>
</tr>
<tr>
<td>unsigned int</td>
<td>freq</td>
<td>0011</td>
</tr>
<tr>
<td>signed int</td>
<td>oil_temp</td>
<td>0012</td>
</tr>
<tr>
<td>unsigned int</td>
<td>fuel_level</td>
<td>0013</td>
</tr>
<tr>
<td>signed int</td>
<td>coolant_temp</td>
<td>0014</td>
</tr>
<tr>
<td>unsigned int</td>
<td>batt_volts</td>
<td>0015</td>
</tr>
<tr>
<td>unsigned int</td>
<td>gen_voltsA</td>
<td>0016</td>
</tr>
<tr>
<td>unsigned int</td>
<td>gen_voltsB</td>
<td>0017</td>
</tr>
<tr>
<td>unsigned int</td>
<td>gen_voltsC</td>
<td>0018</td>
</tr>
<tr>
<td>unsigned char</td>
<td>power</td>
<td>0019</td>
</tr>
<tr>
<td>unsigned char</td>
<td>power_factor</td>
<td>0019</td>
</tr>
<tr>
<td>unsigned int</td>
<td>load_amps</td>
<td>001a</td>
</tr>
<tr>
<td>unsigned int</td>
<td>util_volts</td>
<td>001b</td>
</tr>
<tr>
<td>unsigned int</td>
<td>util_freq</td>
<td>001c</td>
</tr>
<tr>
<td>unsigned int</td>
<td>gen_amps</td>
<td>001d</td>
</tr>
<tr>
<td>struct Dispctrl_bit</td>
<td>Dispctrl</td>
<td>001e</td>
</tr>
<tr>
<td>char</td>
<td>Transfer_error_no</td>
<td>001e</td>
</tr>
<tr>
<td>struct RS232_bit</td>
<td>RS232_bitstr</td>
<td>001f</td>
</tr>
<tr>
<td>struct Output_bit</td>
<td>Output</td>
<td>001f</td>
</tr>
<tr>
<td>union PORTA_store_bit</td>
<td>PORTA_store</td>
<td>0020</td>
</tr>
<tr>
<td>union PORTB_store_bit</td>
<td>PORTB_store</td>
<td>0020</td>
</tr>
<tr>
<td>union PORTP_store_bit</td>
<td>PORTP_store</td>
<td>0021</td>
</tr>
<tr>
<td>union PORTT_store_bit</td>
<td>PORTT_store</td>
<td>0021</td>
</tr>
<tr>
<td>union Alarm_bitaddr</td>
<td>Alarm</td>
<td>0022</td>
</tr>
<tr>
<td>union AVR_23_bit</td>
<td>AVR_23</td>
<td>0023</td>
</tr>
<tr>
<td>union AVR_01_bit</td>
<td>AVR_01</td>
<td>0023</td>
</tr>
<tr>
<td>unsigned long</td>
<td>hours_run</td>
<td>0024 + 0025</td>
</tr>
<tr>
<td>struct mbstart_bit</td>
<td>mb_start</td>
<td>0026</td>
</tr>
<tr>
<td>unsigned int</td>
<td>ps_kw</td>
<td>0027</td>
</tr>
<tr>
<td>unsigned int</td>
<td>ps_pf</td>
<td>0028</td>
</tr>
<tr>
<td>unsigned char</td>
<td>reset_genlink</td>
<td>0029</td>
</tr>
<tr>
<td>unsigned char</td>
<td>parallel_en</td>
<td>002a</td>
</tr>
<tr>
<td>unsigned int</td>
<td>modem_disconnect</td>
<td>002b</td>
</tr>
</tbody>
</table>
```

### Notes for Page 4

**Note 1:**
- Engine_run_status
  - 0 = Keyswitch in off position
  - 1 = Started from manual keyswitch start
  - 2 = Started from remote start
  - 3 = Started from serial link (Genlink)
  - 4 = Started from exerciser (no transfer)
  - 5 = Started from parallel switch
  - 6 = Started from exerciser (with transfer)
  - 7 = Started due to utility loss
  - 8 = Stopped with keyswitch in auto position
  - 9 = Started from peak shave module

**Note 2:**
- Engine_running
  - Bit 0 = Engine running
  - Bit 1 = Utility breaker status
  - Bit 2 = Generator breaker status

**Note 3:**
- This variable can have the following values:
  - Reset = 0
  - Running Up = 6
  - Cool Down = 11
  - Stopped = 1
  - Warming Up = 7
  - Stopping = 12
  - Preheating = 2
  - Load Accept = 8
  - Alarm Stopping = 13
  - Starting = 3
  - Warming Active = 9
  - Alarm Stopped = 14
  - Starting PAUSE (between cranks) = 5
  - Alarms Active = 10
Editors notes:
The following locations are setpoints in EEPROM and are the stored parameters which are not lost when power is removed. (Settings such as cooldown time etc) The parameters have both READ AND WRITE ACCESS.

The following naming conventions are used.

- Variables with a _lsp extension are low setpoints
- Variables with a _hsp extension are high setpoints
- Variables with a _ssp extension are shutdown setpoints

You should not write to EEPROM MEMORY locations on a continual basis as EEPROM MEMORY has a limited number of write cycles before it “wears out”.

```c
#include "avr.h"

// Variables with a _lsp extension are low setpoints
// Variables with a _hsp extension are high setpoints
// Variables with a _ssp extension are shutdown setpoints

// NOT ACCESSIBLE FROM MODBUS

// EEPROM unsigned long hours_run_ep[6]
// EEPROM unsigned int start_attempts
// EEPROM unsigned int pause_time
// EEPROM unsigned int cool_time
// EEPROM unsigned int warm_time
// EEPROM unsigned int hold_off_time
// EEPROM unsigned int mbus_spare1
// EEPROM unsigned int mbus_spare2
// EEPROM unsigned int start_time
// EEPROM unsigned int rpm_started
// EEPROM unsigned int preheat_time
// EEPROM unsigned int fuel_level_lsp
// EEPROM unsigned int fuel_level_hsp
// EEPROM unsigned int gen_volts_lsp
// EEPROM unsigned int gen_volts_hsp
// EEPROM unsigned int rpm_lsp
// EEPROM unsigned int rpm_ssp
// EEPROM unsigned int batt_volts_hsp
// EEPROM unsigned int batt_volts_lsp
// EEPROM signed int oil_temp_ssp
// EEPROM signed int oil_temp_hsp
// EEPROM signed int coolant_temp_ssp
// EEPROM signed int coolant_temp_hsp
// EEPROM signed int coolant_temp_lsp
// EEPROM signed int oil_press_ssp
// EEPROM signed int oil_press_hsp
// EEPROM signed int flywheel_teeth
// EEPROM unsigned char userpwd[8]
// EEPROM unsigned char panel_id[8]
// EEPROM unsigned char avr_droop_gain
// EEPROM unsigned char avr_droop_select
// EEPROM unsigned char avr_stability
// EEPROM unsigned char avr_gain
// EEPROM unsigned char avr_sensing
// EEPROM unsigned char avr_freq
// EEPROM unsigned char avr_kw
// EEPROM unsigned char avr_pf
// EEPROM unsigned char modem_sel
// EEPROM unsigned char gov_stability
// EEPROM unsigned char preheat_en
// EEPROM unsigned char transfer_en

// ifdef CT_RATIO_CODE_REQUIRED
// @eeprom char ct_ratio
// @eeprom unsigned long hours_run_ep
// @eeprom unsigned int start_attempts
// @eeprom unsigned int pause_time
// @eeprom unsigned int cool_time
// @eeprom unsigned int warm_time
// @eeprom unsigned int hold_off_time
// @eeprom unsigned int mbus_spare1
// @eeprom unsigned int mbus_spare2
// @eeprom unsigned int start_time
// @eeprom unsigned int rpm_started
// @eeprom unsigned int preheat_time
// @eeprom unsigned int fuel_level_lsp
// @eeprom unsigned int fuel_level_hsp
// @eeprom unsigned int gen_volts_lsp
// @eeprom unsigned int gen_volts_hsp
// @eeprom unsigned int rpm_lsp
// @eeprom unsigned int rpm_ssp
// @eeprom unsigned int batt_volts_hsp
// @eeprom unsigned int batt_volts_lsp
// @eeprom signed int oil_temp_ssp
// @eeprom signed int oil_temp_hsp
// @eeprom signed int coolant_temp_ssp
// @eeprom signed int coolant_temp_hsp
// @eeprom signed int coolant_temp_lsp
// @eeprom signed int oil_press_ssp
// @eeprom signed int oil_press_hsp
// @eeprom signed int flywheel_teeth
// @eeprom unsigned char userpwd[8]
// @eeprom unsigned char panel_id[8]
// @eeprom unsigned char avr_droop_gain
// @eeprom unsigned char avr_droop_select
// @eeprom unsigned char avr_stability
// @eeprom unsigned char avr_gain
// @eeprom unsigned char avr_sensing
// @eeprom unsigned char avr_freq
// @eeprom unsigned char avr_kw
// @eeprom unsigned char avr_pf
// @eeprom unsigned char modem_sel
// @eeprom unsigned char gov_stability
// @eeprom unsigned char preheat_en
// @eeprom unsigned char transfer_en
// #endif
```
DG Series Modbus

**EXAMPLES**

To get registers 0x000 to 0x00e the request string (in decimal) would be:

50  address=50, say
3   function (fetch registers)
0   Start address high byte
0   Start address low byte
0   Number of registers to get high byte
15  Number of registers to get low byte
crc low
crc high

The response will be:

50  address
16  function (load registers)
0   start address, high byte
0   start address, low byte
0   number of registers to write to, high byte
15  number of registers to write to, low byte
30  byte count
xx  data
xx  data
..  data
crc low
crc high

to write to registers the string would be:

60  address
16  function (load registers)
0   start address, high byte
0   start address, low byte
15  number of registers to write to, high byte
30  byte count
xx  data
xx  data
..  data
crc low
crc high