

Ample Power EnerMatic Controller, V2 and V4

Installation and Operating Instructions

Ample Power Models for 12/24/32/36/48 Volts

February 15, 2010



Introduction

This manual covers the V2 and V4 EnerMatic Controllers. (Note: there was no V3 EnerMatic.) The V4 incorporates new features:

- Electrical isolation between engine circuits and energy battery;
- Electrical isolation between serial ports and engine/energy circuits, and electrical isolation between the two serial ports;
- Circuits to measure solar and wind current; and,
- data flash for parameter data logging.

The EnerMatic Controller, EMC, is a complex combination of hardware and software functions designed to manage an energy system comprised of battery storage and charge sources. Managed charge sources include an engine with one or more alternators, optional solar panels and wind generators.

User interface to the EMC is via a PC running a terminal emulator program such as HyperTerminal for Windows¹. A second communications port is provided for interface to a *host* computer.

Because the EMC has been designed to manage energy systems with a wide variety of batteries, charge sources and engine control features, (no one-size fits all), initial set-up of operating parameters is non-trivial.

Adding to the complexity is the simple fact that batteries themselves are not as simple as most people believe. The EMC can be configured to optimally manage various battery technologies, but many people don't have the background experience to optimize battery performance. The major challenge that the designers of the EMC faced was how to provide the information and controls

necessary to properly manage the energy system, but also provide protection against mis-use.

One result is a system of permissions or privileges. All who use the terminal interface can view all of the data, (except passwords), but changing operational parameters require *privilege*. This level of protection permits experienced users to change operating conditions, without the fear that those conditions will be modified by someone not authorized to do so.

The EMC has evolved as different customers have asked for special features. Some of those have become standard features. Because the EMC will continue to evolve, keeping detailed printed documentation up to date would be a major challenge. Instead the user interface is designed to be very self explanatory once the organizational foundation of the user interface is understood.

This document should be used with an EMC and terminal emulator. It'll make a lot more sense.

Behind the User Interface, UI

Most experienced programmers know that the User Interface for a given project takes the majority of programming time. Part of that is due to the way the UI is represented in computer code.

Early in the development it was decided to describe the UI in a simple text based set of files. A parser program would be written to read the text files and generate tables in C. The parser was written in Python, an elegant and easy to use high level language. It does error checking, making sure that every menu choice has a member to display or execute, and every element has a menu choice.

Managing permissions on operations was also greatly simplified, as was the process of saving configuration variables in external files and being able to read them back in.

The system allows new features to be added into the UI relatively painlessly. There's still the backend programming that needs to be done to implement any new feature, but at least the UI part of it is managed easily.

While the EMC is already highly functional, from time to time users may wish it did something new or different. Many new features are being evaluated or developed. Some of these will become standard if there is enough interest in them

User Interface Elements

There are three basic user interface elements.

- menus;
- panels; and
- forms.

A menu is a list of selections that can be made by entering the selection number followed by the ENTER key. Some menu selections implement a function directly, while other menu selections lead to another menu, or select a panel to be displayed.

Panels are aggregates of data items. For instance fields showing battery voltage, battery current, etc.

¹Windows may be a trademark of Microsoft Corp.

A form accepts entry of operating setpoints and configurations.

Form elements are used to configure and program setpoints. Each item on a form has a name, and when the cursor is on the line of a name, an *Info* line provides a description of that parameter.

User Interface Organization

As noted earlier, the EMC is a combination of hardware and software. At the user interface level it helps to think of individual devices that co-exist in or with the EMC.

For instance, the alternator regulator is a functional device that exists inside the EMC and is comprised of hardware and software elements. Batteries are a functional device, obviously external to the EMC. The engine is a functional device. So are solar panels and wind generators.

As much as possible, the user interface elements are designed to configure and operate the devices that make up the EMC system.

There is also a *system* device that has tentacles into some or all of the other devices. This is the pseudo device that manages passwords, privileges and global configuration parameters.

Categories of Device Controls

By necessity there are different levels of interaction with any functional device. The levels generally fit in the categories below:

- operating options configuration;
- reset of stored historical data;
- setpoint programming;
- calibration programming;
- data display;
- operating controls; and
- configuration and alarm setpoints.

Not all devices have every one of these categories. Default factory settings suffice in many instances.

Options Configurations

Configuration applies to optional modes of operation. For instance, stopping the engine when the batteries reach full charge is a configuration option which can be selected as *True* or *False*.

Another configuration option selects how the remote start/stop input works ... as a momentary switch action to start/stop, or as a toggle switch type action where engine run is one position and engine stop is another.

Reset of Historical Data

So-called historical data is accumulations of long-term value, for instance engine running hours, or lifetime Amp-hours consumed from the batteries. There are menu selections to reset these values.

Setpoint Programming

An example of a setpoint is the absorption voltage that the alternator regulator needs to achieve. In a 12V application that setpoint might be 14.40 Volts.

Calibration Programming

Analog data can be scaled and conditioned before being read by an *analog-to-digital converter*. Precision components are used in all analog circuits so calibration beyond factory default values are rarely needed.

However, it is possible to program calibration constants to achieve correlation with an external meter.

Shunt size can be changed via calibration constants. That is the EMC can be told that an external shunt is 200 Amps, rather than the standard 400 Amps.

Data Display

Operational data is displayed in so-called panels. The aggregation of parameters into panels is generally related to a device, however, some panels may show data values for more than one device.

Operating Controls

While the EMC is expected to operate automatically most of the time there are occasions when devices need to be tested or exercised manually. Menu selections are provided to allow manual control, for instance to start or stop the engine.

Alarm Configuration and Setpoints

Some devices, for instance the battery bank and the engine have *alarm detectors* which are software functions that look for out-of-band parameters. For instance battery voltage above a given value is too high and an error should be raised, or other actions taken to resolve the condition.

Alarm detectors have to be enabled, and their limits must be set. In systems which incorporate external GSM modems or wired access to outside resources, alarm detectors may trigger text messages or emails to a responsible party who will take appropriate action.

Configuration and Setpoint Preservation

Configurations and setpoints are saved in non-volatile memory so they are preserved in the event power is removed from the EMC. Those values can also be saved externally in a file on a PC, and restored from that file if necessary.

It is highly recommended that the configuration and setpoint values be saved externally. In the event that an EMC is replaced by another unit, the file can be read into the new EMC to establish the same operating conditions.

Privilege Levels

As mentioned, privilege levels are used to protect configurations, setpoints, etc., from being changed by unauthorized users. Every item in the user interface has an associated privilege level.

There are currently eight privilege levels. Each level can also perform all actions of privilege levels below it. The EMC verifies that a user is authorized by privilege level to modify operation setpoints.

Privilege levels are:

- Root
- Distributor
- Dealer
- Administrator
- System Configurator
- Technician
- Operator
- User

Root can do anything. Only the engineers at Ample Power Company have root permissions.

A distributor of the EMC has the next highest privilege level, followed by the dealer.

The Administrator has privileges of those below it. One thing it can do is view and change the passwords of the levels below it.

The Systems Configurator can do all of the functions below it, but most importantly it can decide on the configuration of the system, for instance selecting how the engine will start and stop.

The Technician is allowed to change setpoints such as the alternator regulator absorption or float voltage.

It takes Operator privilege level to start/stop the engine.

At the User level all data can be viewed, but nothing can be changed.

There are no default passwords installed at the factory for System Configurator and below. Pressing the *ENTER* key suffices for password entry for these levels. The owner of the system should enter passwords for these levels to provide protection.

There is an intersection between privileges and programming forms for functional devices. That is, a form may have some values that require Configurator privilege, and others that only require Technician privilege. Items in the form can only be changed with an appropriate privilege. A Technician can see how the system is configured, but is not allowed to change it.

Engine Configuration and Setpoints

There are many ways the engine can be configured to start and stop. In the process of starting and stopping, various signals are actuated in timed sequences, for example starting the fuel pump and glow plugs before activating the starter motor. There are several menus devoted to engine start/stop configuration and timing of the various signals.

There are also engine alarms to be enabled and alarm actions to be configured.

As an example, the form with the title of *Program Start Configurations/Conditions* has an element named *start_vt_cfg*. The *Info* line says, *Enables Start on Volts/Time Conditions*. Below the configuration variable are two more lines with names of *start_vt_volts* and *start_vt_time*.

The *Info* lines for these two variables are respectively, *Start if Voltage Less Than this Trip Point* and *Start if Voltage Less Than Trip-Point for this Time (seconds)*.

These three variables are related by virtue of the shared *start_vt_* in their names. One variable allows the engine to start if enabled, and the others provide the voltage below which to start, after the voltage stays below the setpoint for the specified time in seconds.

Engine Start Timing Configuration

Engine Start Timing Parameters are those that control the length of time that certain events take. For instance there is a fuel start time and a fuel stop time. All times are based on receipt of a start command. That is, a start time of 12 seconds means that the event should be started 12 seconds after being told to start. This is typical of the time when engine cranking begins. Prior to this, the glow plugs should have been activated, and typically the glow plugs continue to be activated during the cranking period. All the parameters can be programmed as necessary.

One noteworthy set of parameters are the fuel pump start and stop. Typically the fuel pump will be started early in the starting process. Some engines, such as the EA-330 don't have an internal fuel pump, so the fuel must either be gravity fed, or continuously fed

by a fuel pump.

Other engines have a cam driven diaphragm fuel pump, but an electric fuel pump is often included in the system to be used as a primer pump. It's handy to use after changing fuel filters to bleed the system of air. It can also be used as a primer pump each time the engine is run. If left running all the time the engine operates then lots of fuel would be pumped through the system, probably wearing out the pump. (With a vane pump this feature can be used as a diesel *polishing* system.)

The fuel pump can be stopped by programming a time into the fuel stop variable. If it is 0, then it will run continuously.

Engine Warm-Up and Cool-Down Cycles

The engine should be warmed up for 1-5 minutes before turning on the alternator. This is controlled via the programmable value for engine warm up time.

There is a corresponding cool down period that can be programmed in the engine stop timing form. The alternator is turned off and the engine is allowed to run during the cool down period.

(Sometimes, however, the engine must be stopped immediately. Thus there is a *stop now* command.)

Executing Functions

As you walk through the menus and select some items a response will be displayed. The name of a function will be displayed along with the response from it.

The same functions can be called via the RAP interface. They do have to be embedded in the RAP protocol to be executed. The RAP Protocol document can be found at <http://www.amplepower.com/wire/index.html>. The RAP port is designed to be used with a computer operating a special interface program, but, however tedious, can be used with a terminal interface by copying and pasting from a file with RAP strings.

As of July, 2008, the executable functions are:

- Engine Operations
 - f_eng_start ... Start the Engine
 - f_eng_stop ... Stop the Engine with Cool Down
 - f_eng_stop_now ... Stop the Engine Immediately
- Solar Panel Operations
 - f_solar_automate ... Automate Solar On/Off
 - f_solar_connect ... Connect Solar
 - f_solar_disconnect ... Disconnect Solar
- Wind/Hydro Generator Operations
 - f_wind_automate ... Automate Wind/Hydro On/Off
 - f_wind_connect ... Connect Wind/Hydro
 - f_wind_disconnect ... Disconnect Wind/Hydro
- Parallel Solenoid Operations
 - f_parallel_automate ... Automate Parallel
 - f_parallel_connect ... Connect Parallel
 - f_parallel_disconnect ... Disconnect Parallel
- Alternator Regulator Operations
 - f_reg_assert_automatic ... Reset/Automatic
 - f_reg_assert_float ... Lock at Float
 - f_reg_assert_gassing ... Lock at Gassing
 - f_reg_assert_absorption ... Lock at Absorption
 - f_reg_assert_eq1 ... Start Equalization
- Reset History Data

- f_reset_b1_ah ... Reset Battery 1 Amp Hours
- f_reset_b1_trip ... Reset Battery 1 Trip Amp Hours
- f_reset_b1_life ... Reset Battery 1 Lifetime Amp Hours
- f_reset_b1_chrg ... Reset Battery 1 Charge Amp Hours
- f_reset_b1_eff_ah ... Reset Battery 1 Efficiency Amp Hours
- f_reset_b1_full ... Reset battery 1 to full state
- f_reset_eng_hist ... Reset Engine Historical values
- f_reset_all ... Reset all of the above

- System Operations

- f_read_err_msg ... Read Last Fatal Error Message
- f_clear_alarm ... Turn off the Alarm

Fatal Error Messages

Bug free software of any complexity beyond a few pages of code doesn't exist. While firmware is not shipped with known bugs, there is still a possibility that some unexpected edge case of operation will stop the processor from functioning normally ... a fatal error.

If the processor were to remain halted, many damaging and compounding events might occur. Rather than let this happen, fatal errors are first trapped and outputs are put at a default state. Next, information identifying the trap is written into non-volatile memory.

After the location of the fatal error is preserved, the processor re-boots ... starts at the beginning of code as if power was just applied.

Fatal error messages can be read by any user and reported to the Ample Power support team.

WatchDog Output

The EMC provides an output signal, named *THROB* that alternates on and off at a one second interval. Loss of this signal indicates that the EMC processor has stopped running.

It is possible to monitor this signal with an external watchdog power control circuit that would de-power the EMC for a few seconds if the *throb* signal stop.

When the watchdog circuit re-powers the EMC it should wait a few seconds before monitoring the *throb* signal again.

Wiring

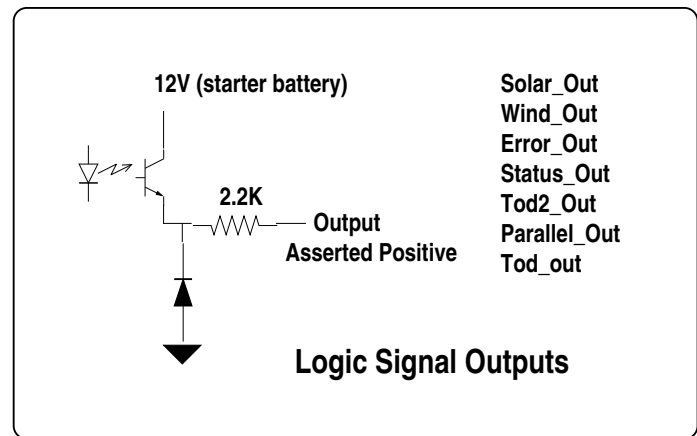
The EMC can be used to run the User Interface with just power and the serial ports wired.

Refer to the proper wiring diagram based on the EnerMatic version. The latest version is the V4.

Logic Signal Levels

There are signals to control external devices. These signals have been isolated from ground in the EnerMatic Controller to eliminate signal switching from interfering with analog measurements, such as battery current.

A simplified circuit diagram is shown below. The list on the right side of the circuit has the names of the signals using the circuit.



Overall Wiring Diagram

Many users incorporate the EnerMatic Controller in their own systems, using different subsets of features and connections. A drawing showing all possible systems would be an undecipherable set of dashed lines for options and many notes and clauses.

There is an overall drawing made for a specific system that can be used with the drawings presented above as a basis of a custom diagram. See Overall Wiring Diagram available from the website.