SC12
12-channel solenoid controller/driver

USER'S GUIDE

Rev 1.2
June 2013

RW Automation, LLC

www.rwautomation.com

Table of Contents

1) Legal Notices

1.1) Copyright

All material presented in this user's manual is subject to the copyright laws of the United States of America.

RW Automation, LLC reserves the right to make changes and improvements to its products and to this document without notification.

1.2) License agreement

The software source code and executables packaged with the SC12 solenoid controller board are intended for use only with the SC12 solenoid controller board. Any other use of the software is strictly forbidden without written permission of RW Automation, LLC. The source code and executables packaged with the SC12 solenoid controller board may be altered and recompiled into new executables. Such executables based on the original software packaged with the SC12 solenoid controller board may be freely distributed for resale along with the SC12 solenoid controller board, but may otherwise not be distributed without the SC12 solenoid controller board.

1.3) Limited warranty

RW Automation, LLC warrants the SC12 solenoid controller board and associated software to be free from defects in material and workmanship and to perform to applicable published RW Automation, LLC specifications for a period of one year from the date of shipment to the purchaser.

RW Automation, LLC will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. RW Automation, LLC shall have the right of final determination as to the existence and cause of defects.

In no event shall RW Automation, LLC be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, RW AUTOMATION, LLC MAKE NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. RW AUTOMATION, LLC WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEROF.

1.4) Third-party trademarks

“Microsoft”, “Microsoft Visual C++ 6.0”, “Microsoft Visual Studio” and “Microsoft Windows” are registered trademarks of Microsoft Corporation.

2) Introduction

Congratulations on your purchase of the SC12 solenoid controller board. Please take the time to familiarize yourself with the material and features of the SC12 solenoid controller board package before proceeding to the next section.

2.1) Packing list

When you purchase the SC12 solenoid controller board, you should receive the following items:

a) The SC12 solenoid controller board
b) Twelve 2-position, 3.81mm terminal blocks (one for each solenoid)
c) Two 8-position, 3.81mm terminal block (for TTL inputs)
d) CD-ROM (with solenoid control library and sample application)
e) RS-232 serial cable

2.2) Features

The SC12 solenoid controller board controls twelve independent solenoids. By utilizing the SC12 controller board's peak-and-hold algorithm, the user can energize each solenoid at its full rated voltage for fast actuation and strong pull, and then have the SC12 automatically reduce the average voltage applied to the solenoid in order to save power and prevent over-heating of the solenoid while still holding the plunger in-place.

Other features:

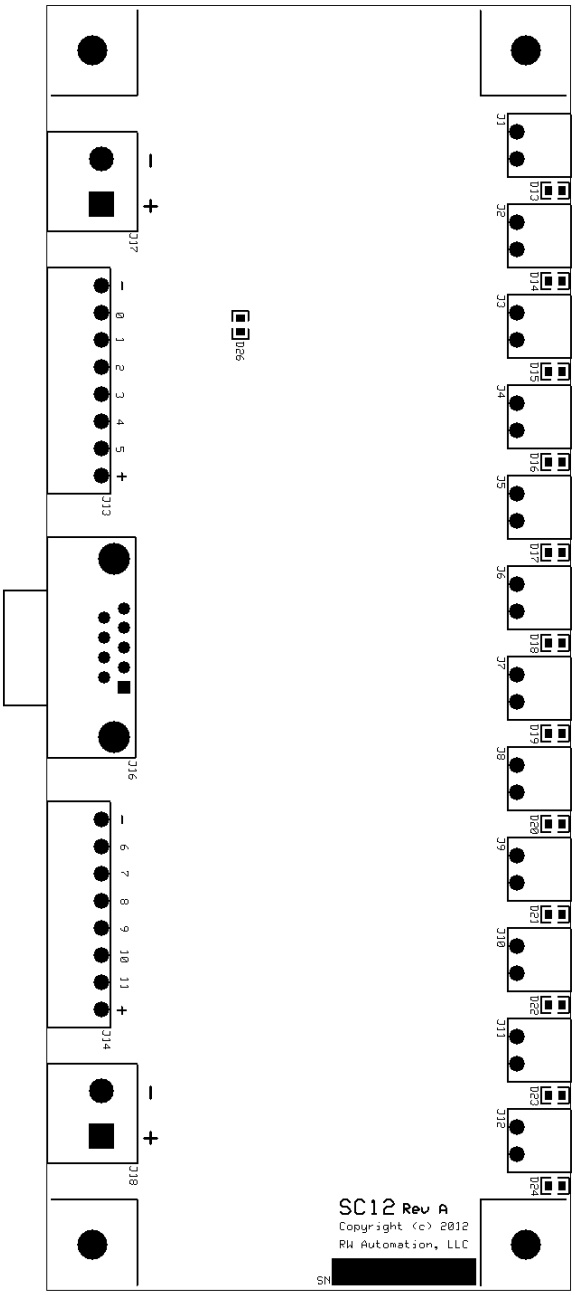
a) RS-232 port to connect to a host computer. The baud rate is fixed at 19200.
b) Concise ASCII command set.
c) Allows both serial and 3.3V LVTTL control of solenoids.
d) User programmable PWM frequency (1KHz to 30KHz).
e) User programmable peak PWM levels (independent for each solenoid).
f) User programmable peak PWM times (independent for each solenoid).
g) User programmable hold PWM levels (independent for each solenoid).
h) User programmable TTL input modes and debouncing.
i) User parameters can be stored in flash memory.
j) Single supply input from 7-40VDC.
k) Detachable terminal block connectors for quick connect/disconnect.
l) Power and status LEDs.

In addition, a software library and solenoid control application program (including source code) that can be built under Microsoft Visual C++ 6.0, Microsoft Visual Studio 2004, Microsoft Visual Studio 2008, and Microsoft Visual Studio 2010 is included.

3) Configuration

This section describes the steps needed to set up the SC12 solenoid controller board before you will be able to use it.

Figure 1 – Overview of connector and status LED locations



Note: connector faces are flush with the perimeter of the board

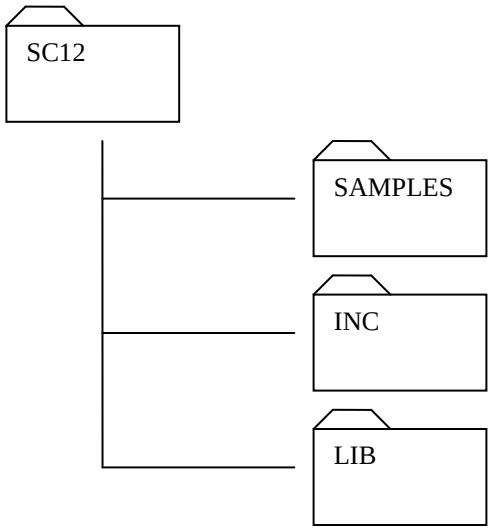
The connectors and LEDs shown in figure 1 are listed in the following table:

Component	Description
J1	Solenoid 0 connector
J2	Solenoid 1 connector
J3	Solenoid 2 connector
J4	Solenoid 3 connector
J5	Solenoid 4 connector
J6	Solenoid 5 connector
J7	Solenoid 6 connector
J8	Solenoid 7 connector
J9	Solenoid 8 connector
J10	Solenoid 9 connector
J11	Solenoid 10 connector
J12	Solenoid 11 connector
J13	TTL input connector (controls solenoids 0 – 5)
J14	TTL input connector (controls solenoids 6 – 12)
J16	RS-232 serial port connector
J17	7 to 40 VDC connector (powers solenoids 0 – 5 and +3.3V regulator)
J18	7 to 40 VDC connector (powers solenoids 6 – 12)
D13	Solenoid 0 status LED
D14	Solenoid 1 status LED
D15	Solenoid 2 status LED
D16	Solenoid 3 status LED
D17	Solenoid 4 status LED
D18	Solenoid 5 status LED
D19	Solenoid 6 status LED
D20	Solenoid 7 status LED
D21	Solenoid 8 status LED
D22	Solenoid 9 status LED
D23	Solenoid 10 status LED
D24	Solenoid 11 status LED
D26	Power LED

3.1) Installing the software

The SC12 software runs under Microsoft Windows 98/2000/XP/Vista/7 operating systems. It is distributed as a single self-extracting setup executable file. Simply run the setup file and follow the instructions. The SC12 library software, sample application, this document, and an uninstall utility are all provided in the distribution file.

Once installed, the software will have the following directory structure:



The SAMPLES sub-directory holds the sample application executable, as well as the full source code and project files for building the application under Microsoft Visual C++ 6.0 and versions of Microsoft Visual Studio.

The INC sub-directory holds the header files user applications will need to include with their projects to use the library.

The LIB sub-directory contains the .lib (static library) source files and project files for the solenoid control library files. Pre-built static libraries for the normal and multi-threaded builds of the solenoid control library are included.

Refer to sections 4 and 5 for more details regarding the use of the software after it has been installed.

3.2) Host computer communications

Figure 2 below illustrates an example of how to connect the SC12 solenoid controller board (the “slave") to a host PC (the “master"). A 9-pin female-male cable wired straight through is appropriate for most recently manufactured PCs, however, older PCs and other types of computers may have different connectors (often 25-pin) and pinouts. In such cases, it is up to the user to purchase whatever additional null modem, gender changer, and adapter cables are needed to suit the requirements of the SC12 solenoid controller board. For use with newer PCs and laptops that do not have an exposed serial port, the user may purchase a USB-to-serial adapter.

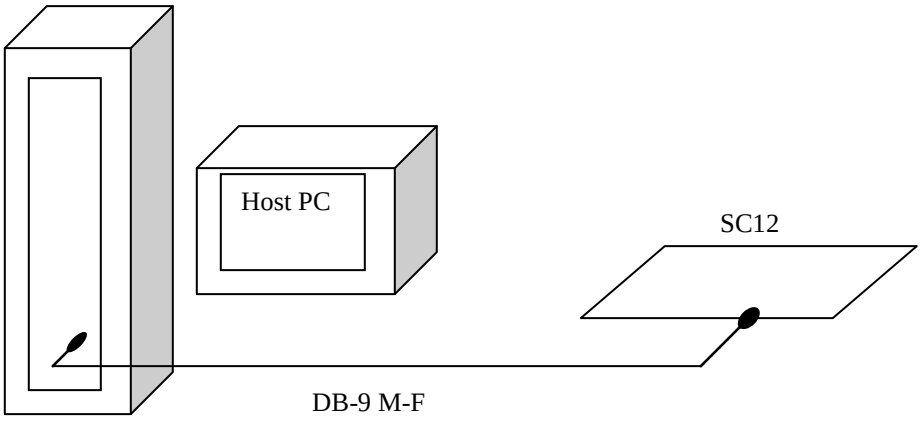
Below is the pinout of the RS-232 connector on the SC12 solenoid controller board used to connect the SC12 solenoid to a host PC. Note that flow control lines are not supported by the SC12 solenoid controller board.

RS-232 “master” connector pinout

Pin	Function
1	Loopback to pins 4 and 6
2	Receive
3	Transmit
4	Loopback to pins 1 and 6
5	Ground
6	Loopback to pins 1 and 4
7	Loopback to pin 8
8	Loopback to pin 7
9	Not connected

The RS-232 parameters are: 19200 baud, 8-bits, no parity, one stop bit.

Figure 2 – Host PC to SC12 connection



3.3) Connecting power

The SC12 solenoid controller board has two power inputs (J17 and J18). The SC12 solenoid controller board requires a single voltage source in the range +7 to +40 VDC connected to both J17 and J18 in parallel. Alternatively, two supplies with a **common ground** may be used, with one supply (connected to J17) powering solenoids 0 – 5 and another supply (connected to J18) powering solenoids 6 – 11. Note that in addition to supplying power to solenoids 0 – 5, connector J17 also powers an on-board +5VDC switching regulator and a +3.3 VDC LDO linear regulator needed for the gate drivers and microprocessor. The +3.3VDC supply is also available to the user (with restrictions on current) via the TTL connectors.

Power connector pinout

Pin	Function
1	Ground
2	+7 to +40 VDC

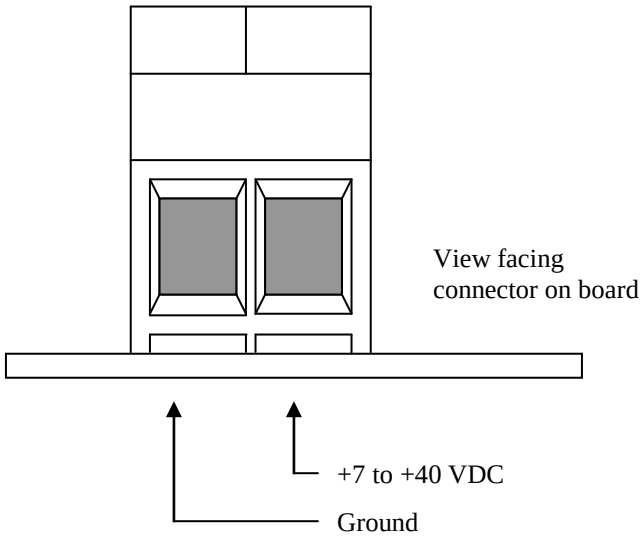
If all twelve solenoids are to be run simultaneously at the maximum average current (2 Amperes each), then the supply must be able to support a minimum of 25 Amperes of continuous current, split evenly between J17 and J18. The supply must also be rated to handle peak currents in excess of 2A/solenoid if the user exceeds the average current rating when turning solenoids on.

WARNING: Please note carefully the required polarity of the power supply connectors. Applying power of the incorrect polarity can damage the SC12 solenoid controller board and will void the warranty.

WARNING: Applying voltages outside of the required 7-40 VDC range can damage the SC12 solenoid controller board and will void the warranty.

Figure 3 below shows the 2-pin power connector illustrating how to connect power to the SC12 solenoid controller board.

Figure 3 – Power connectors



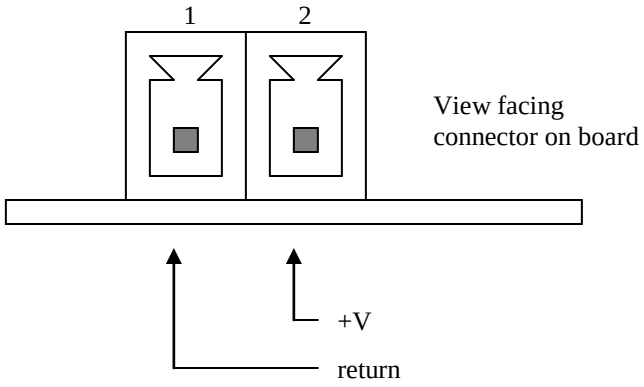
3.4) Connecting solenoids

Each solenoid is assigned a logical number from 0 to 11. All twelve solenoid channels are wired identically. Figure 4 illustrates the solenoid connector and the pinout is shown in the table below.

Solenoid connector pinout

Pin	Function
1	return
2	+V

Figure 4 – Solenoid connector



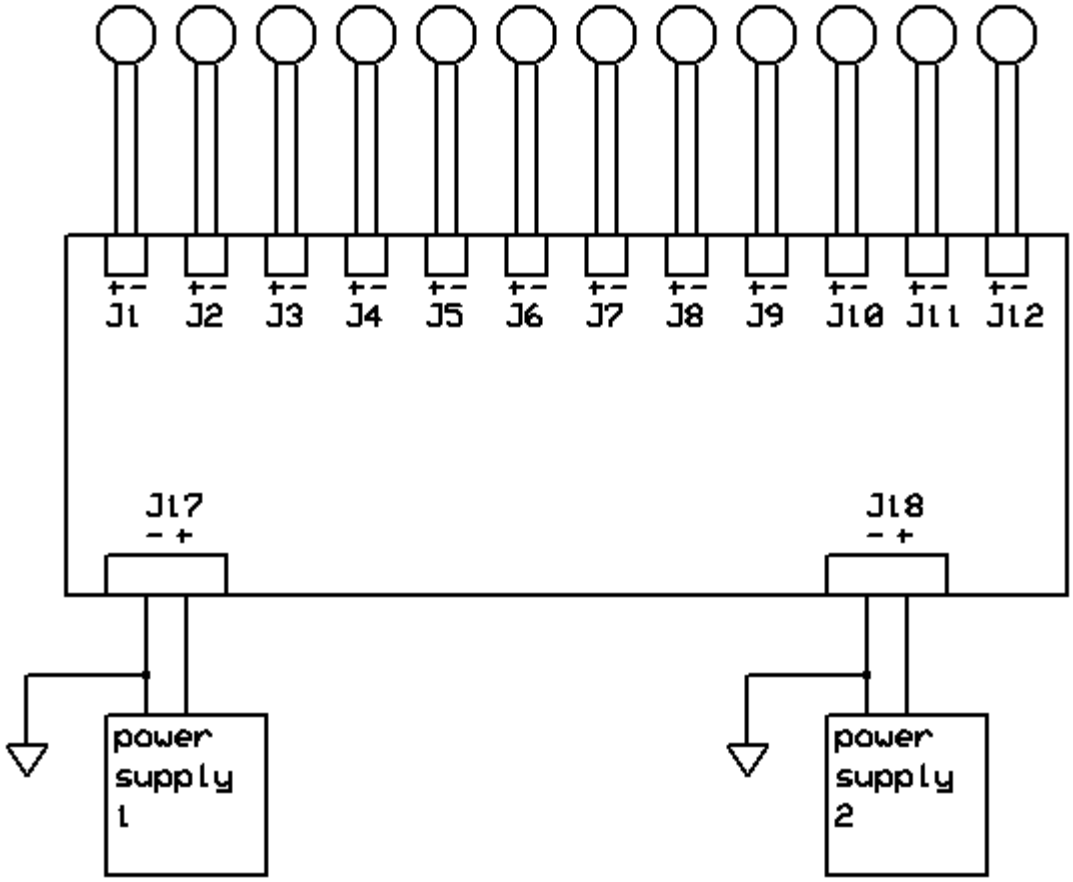
WARNING: Shorting the outputs of the solenoid connector or otherwise driving excessive current can damage the SC12 solenoid controller board and will void the warranty.

WARNING: The components and thermal pads on the SC12 solenoid controller board can get very hot. Avoid touching the board when in operation. It is recommended that forced air be directed at the components and thermal pads in order to increase their ability to dissipate heat.

3.5) Connecting multiple solenoids

The most straightforward way to connect a set of solenoids is to connect each solenoid to its channel independently of the others. This is illustrated in the diagram of figure 5 below. Note how two wires run from the board to each solenoid. Thus, for a full set of twelve solenoids, 24 wires are required. Note: a single power supply can be used for both power inputs, if desired.

Figure 5 – Typical solenoid wiring scheme



WARNING: It is the user's responsibility to select wires of gauge sufficient to handle the current demand of the SC12 board and solenoids. Failure to do so can result in heating of wires and significant voltage drops across the wiring.

In some applications, the wires from the SC12 controller board to the solenoids can be quite long. In order to reduce the wiring required one might be tempted to use a common ground for all solenoids. This reduces the wires from 24 down to 13, a reduction of almost 50%. However, the common ground implementation has a **serious flaw**. The solenoid drivers on the SC12 controller board each use what is referred to as a "low-side" switch. Thus it is the positive supply terminal that is always connected the solenoid and the ground is alternately connected and disconnected by the switch (power transistor). If all solenoids are connected to a common ground they will not be controllable.

Figure 6 below shows a variation of the circuit in figure 5. In this a case it is the positive voltage that is used as a common rail for all solenoids. The individual solenoids are now all controllable since each one has a wire running to one of the low-side switches. However, there is still a minor flaw in this approach. Each contact on the solenoid connectors (J1 – J12) is rated up to 8 amperes. By connecting many solenoids to just one of the connectors, the current rating can easily be exceeded and can potentially damage to the board.

Figure 6 – Common positive voltage for each solenoid bank. This connection is only appropriate if the current remains below the 8A rating for the solenoid connectors.

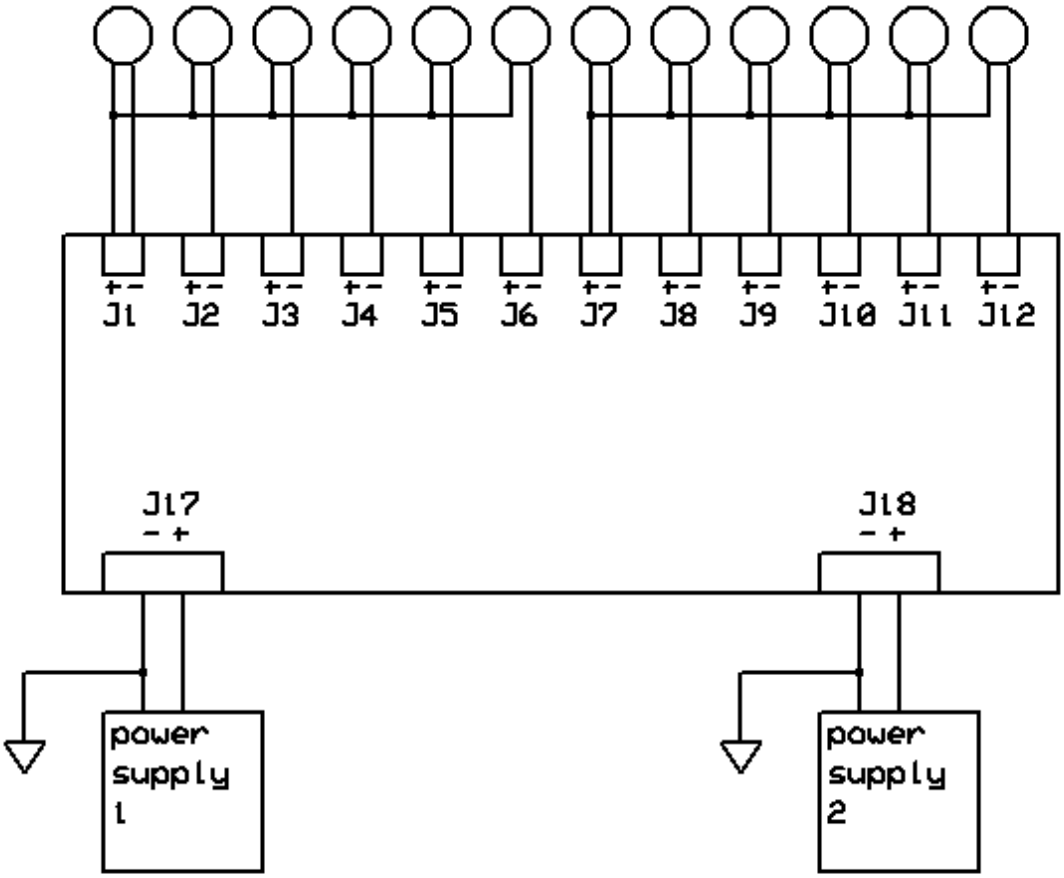
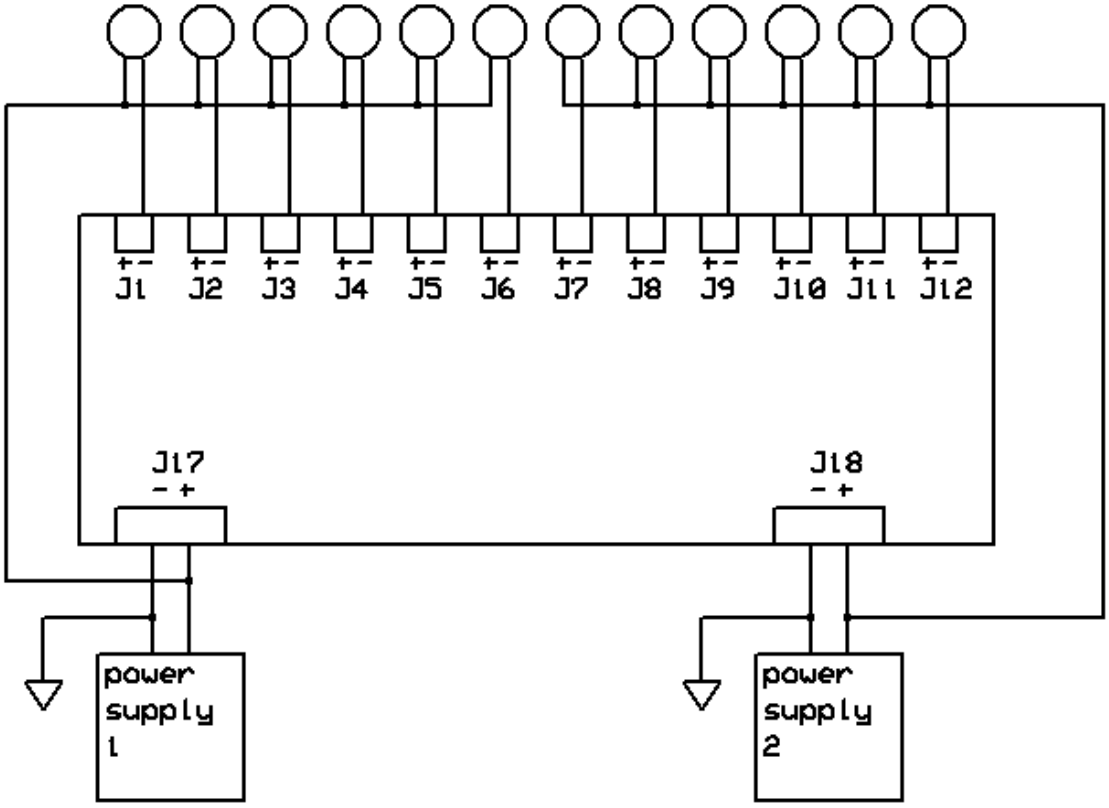


Figure 7 below shows an alternative to the circuit of figure 6. In this case, the common positive voltage for each bank of solenoids is connected directly to the positive output of the bank's power supply, thus bypassing the current restrictions imposed by the solenoid connectors. When such a circuit is implemented, the user should take care to use wires of appropriate current carrying capacity. Note: again, two separate power supplies are shown as the most general case, they can be the same supply if desired.

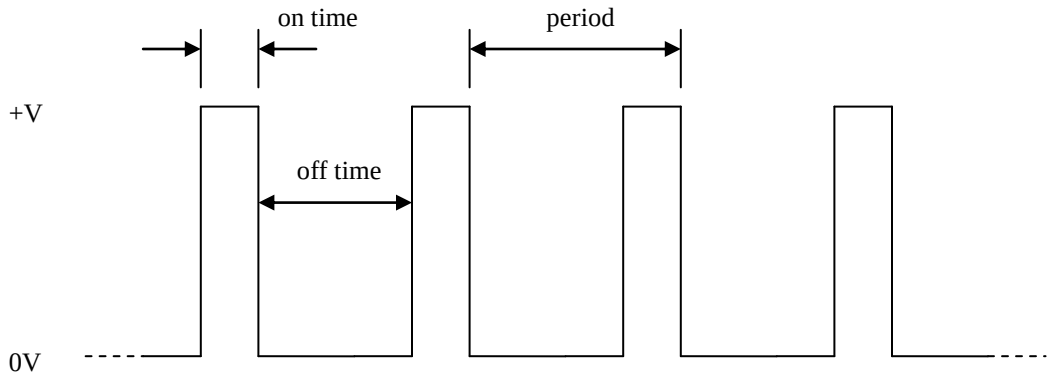


3.6) Controlling solenoids

Power is supplied to each solenoid independently using pulse-width-modulation (PWM) and a microprocessor-controlled peak and hold algorithm.

The PWM signal is a rectangular wave as shown in figure 8.

Figure 8 – example of PWM waveform



The ratio of the "on time" and the period, along with the voltage applied to the controller board and the type of solenoid being used will dictate the average current flowing in the solenoid.

The peak-and-hold algorithm applies one PWM signal to the solenoid when it is first turned on (the peak level). This peak level is held for a period of time, after which a second PWM signal is applied to the solenoid (the hold level).

This allows the user to apply a larger current to the solenoid when it is first turned on, and then hold the solenoid with a smaller current. This results in faster actuation of the solenoid, but generates less heat and uses less power once the solenoid has been actuated.

WARNING: It is the user's responsibility to select the correct operating voltage and PWM levels to ensure the peak current is less than 10A per solenoid and that the average current is maintained at or below 2A per solenoid.

3.7) Connecting TTL control lines

The SC12 solenoid controller board has two 8-input TLL connectors. The pinouts for the connectors are shown in below.

J13 – TTL input connector 1

Pin	Function
1	Ground
2	TTL input 0
3	TTL input 1
4	TTL input 2
5	TTL input 3
6	TTL input 4
7	TTL input 5
8	+3.3V

J14 – TTL input connector 2

Pin	Function
1	Ground
2	TTL input 6
3	TTL input 7
4	TTL input 8
5	TTL input 9
6	TTL input 10
7	TTL input 11
8	+3.3V

The inputs use LVTTL 3.3V gates. Do not exceed the 0-3.3VDC input range of the inputs. The TTL inputs are pulled high on the SC12 board.

Each input is mapped (logically) to the corresponding solenoid on the SC12 solenoid controller board (for example, TTL input 0 is mapped to solenoid 0). The TTL inputs may be independently programmed to control their corresponding solenoids under a variety of modes. Each TTL input has 5 user-programmable modes of operation listed in the table below. The user may also program the amount of debouncing used for each TTL input.

TTL input mode	Action(s)
Disabled	TTL input has no effect on state of solenoid
Active on low-to-high transition	On a low-to-high transition the solenoid is turned on. On a high-to-low transition the solenoid is turned off. The peak-and-hold algorithm timing is reset on each transition. When the system is powered up, no action will occur.
Active on high-to-low transition	On a high-to-low transition the solenoid is turned on. On a low-to-high transition the solenoid is turned off. The peak-and-hold algorithm timing is reset on each transition. When the system is powered up, no action will occur.
Active on high level	When the TTL input is high, the solenoid will be turned on. When the TTL input is low, the solenoid will be turned off. The peak-and-hold timing is reset each time a state is encountered in which the solenoid is off and the TTL level is high. **Warning**: when using this TTL input mode, the solenoid can be activated immediately when the system is powered up.
Active on low level	When the TTL input is low, the solenoid will be turned on. When the TTL input is high, the solenoid will be turned off. The peak-and-hold timing is reset each time a state is encountered in which the solenoid is off and the TTL level is low. **Warning**: when using this TTL input mode, the solenoid can be activated immediately when the system is powered up.

A limited +3.3 VDC supply is available on these connectors for users that need to power circuits used with the TTL interface. **Note**: no more than 100 milliamperes (total for both connectors) should be drawn from the +3.3 VDC regulator.

3.8) Status LEDs

The SC12 solenoid controller board has thirteen status LEDs (refer to figure 1).

The power LED (D26) will be on whenever the on-board +3.3 VDC regulator is active.

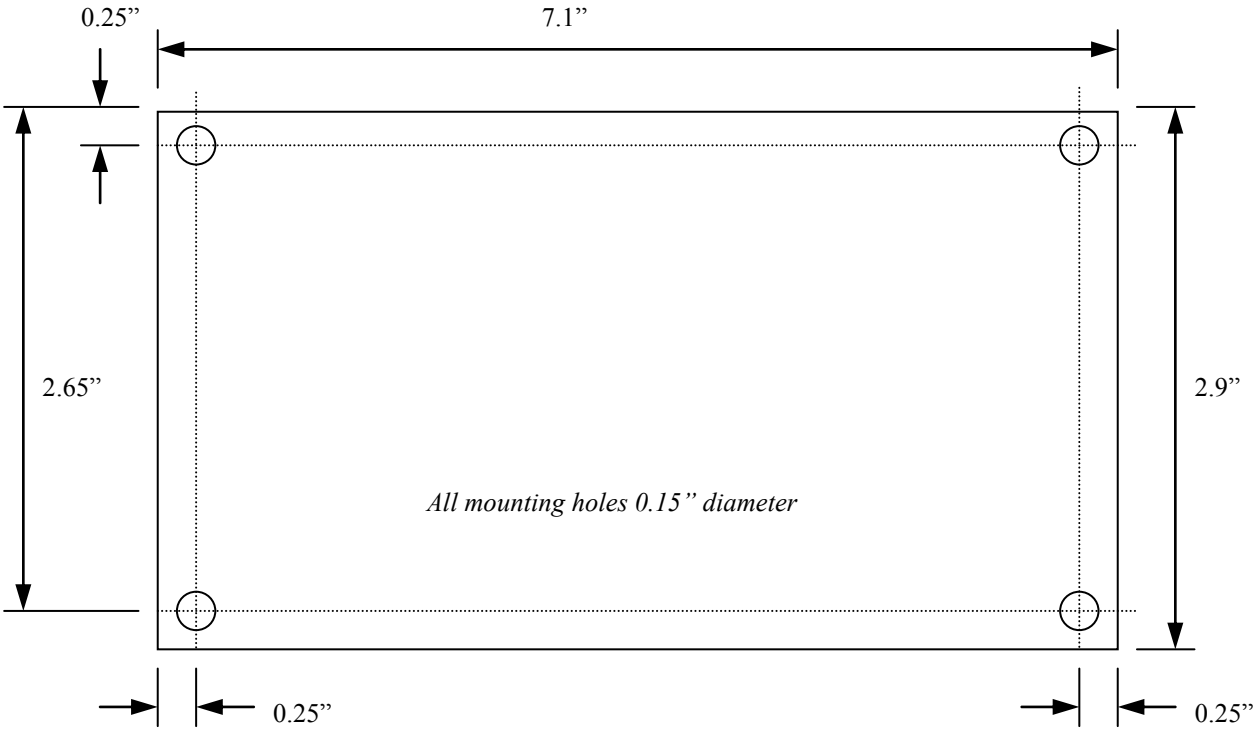
The solenoid status LEDs (D13 through D24, corresponding to solenoids 0 through 11) will be on whenever there is a current being applied to the corresponding solenoid. The solenoid state LEDs are off when no current is being applied to the corresponding solenoid.

3.9) Mechanical specifications

The SC12 solenoid controller board is 7.1” x 2.9” (180.3mm x 73.7mm). The height of the tallest component is 0.984” (25.0 mm).

Mounting hole locations are shown in figure 10 below (not draw to scale).

Figure 10 – Mounting hole locations



4) Solenoid Control Library

The solenoid control library is a software library of C functions that can be compiled under Microsoft Visual C++ 6.0 and Microsoft Visual Studio.

These functions provide an application programming interface (API) for the solenoid controller without the need to learn the SC12 solenoid controller board serial communication protocol or the API for the PC's serial port.

The source code is included to allow the interested user to more fully understand how to use the serial communication protocol (see section 6). As well, the source code makes it possible to port the library to platforms other than those supported under Microsoft development tools.

The subsections that follow give details of each of the library API calls and their usage. Each API function will return an integer status code. The codes are enumerated as follows:

0 NO ERROR

This is the status normally returned by an API call. It indicates that the API call was completed successfully.

1 SERIAL PORT ERROR

This will occur when a low-level serial port error occurs. For instance, when an API call specifies the use of a serial port that does not exist, or is in use by another application.

2 NOT INITIALIZED ERROR

This will occur when calling an API function before the library is initialized by calling the `scl_initialize()` function.

3 TIMEOUT ERROR

This will occur when an attempt to communicate with the SC12 solenoid controller board fails. This will normally only occur if there is a problem with the connection between the host computer and the SC12 solenoid controller board.

4 PARAMETER ERROR

This indicates that a value for a parameter has been passed to an API function that is outside of the range expected for that function.

4.1) Get library version

```
int scl_get_library_version(int *version)
```

The calling function must pass an integer pointer to hold the version number. The version number of the library is put into memory at the address held by the pointer. The two least significant digits of the version value are used for the minor version number, the rest of the digits are used for the major version number. For example, library version 1.00 is represented by a value of 100.

4.2) Get controller model

```
int scl_get_controller_model(int *model)
```

The solenoid control library supports both the SC5 solenoid controller and the SC12 solenoid controller boards. This function may be used at run time to verify the expected solenoid board is connected. The calling function must pass an integer pointer to hold the model ID. When an SC5 controller is connected, the model ID is 0, when an SC12 controller is connected, the model ID is 1.

4.3) Initialize

```
int scl_initialize(int com_port)
```

The user should call this initialization function only once at the start of the program to initialize the solenoid controller library. Most of the other API functions will return a "not initialized" status if this function is not called first.

The `com_port` parameter specifies the RS-232 serial port, where 1 is used for COM1, 2 for COM2, etc.

The `scl_initialize()` function will attempt to open the specified serial port and will then send out an initialization command to the SC12 solenoid controller board connected to the port. If the solenoid controller board is currently running, it will be reset to its default conditions (see section 6).

Once the `scl_initialize()` function is called, calling it again will re-initialize the SC12 board, but will not change the communication port. If the user wants to specify a different com port, then the `scl_deinitialize()` function must be called prior to calling the `scl_initialize()` function again with the new com port parameter.

4.4) De-initialize

```
int scl_deinitialize(void)
```

The `scl_deinitialize()` API function is complementary to the `scl_initialize()` function. The user's program should call this function when it is being terminated or when it is desired to release the serial port specified in the call to `scl_initialize()`.

When `scl_deinitialize()` is called, the SC12 solenoid controller board will be re-initialized and the library will return to its uninitialized state.

4.5) Set solenoid state

```
int scl_set_solenoid_state(int solenoid_id,
                           int state)
```

This API function sets the state for the specified solenoid (in the range 0 to 11 for the SC12 solenoid controller board).

The state must be either 0 (to turn solenoid off) or 1 (to turn solenoid on). When 1 is specified, the peak-and-hold algorithm timing (see section 3.6) is reset (even if the solenoid was already on).

4.6) Set all solenoid states

```
int scl_set_all_solenoid_states(int state0,
                                int state1,
                                int state2,
                                int state3,
                                int state4,
                                int state5=0,
                                int state6=0,
                                int state7=0,
                                int state8=0,
                                int state9=0,
                                int state10=0,
                                int state11=0);
```

This API function sets the states for all twelve solenoids simultaneously. The state0 parameter corresponds to solenoid 0, the state1 parameter corresponds to solenoid 1, etc.

Each state parameter must be either 0 (to turn solenoid off) or 1 (to turn solenoid on). When 1 is specified, the peak-and-hold algorithm timing for the corresponding solenoid (see section 3.6) is reset (even if the solenoid was already on).

NOTE: in order to preserve backwards compatibility with earlier versions of the solenoid controller library, states5 through state11 may be omitted from the function call and the corresponding solenoids will be turned off.

```
int scl_set_all_solenoid_states_bitfield(int state_bitfield)
```

For those users that prefer to store states in a bitfield, this function has been provided. The least significant bit (bit 0) stores the state of solenoid 0, bit 1 stores the state of solenoid 1, etc.

4.7) Set solenoid parameters

```
int scl_set_solenoid_parameters(int solenoid_id,
                                int peak_level,
                                int peak_time,
                                int hold_level)
```

This API function sets the peak-and-hold parameters for the specified solenoid (in the range 0 to 11 for the SC12 solenoid controller board).

The peak level must be in the range 0 to 256, and represents the portion of time (out of 256) that the voltage is applied to the solenoid for each PWM cycle during the peak period. Thus, for a value of 0, no voltage is applied to the solenoid. For a value of 256, voltage is continuously applied to the solenoid. For a value of 128, a 50% duty cycle square wave is applied to the solenoid.

The peak time parameter must be in the range 0 to 65535 and specifies the number of milliseconds the peak level is to be applied to the solenoid (when it is turned on) before reverting to the hold level.

WARNING: High peak currents (in excess of 4 Amperes) should only be applied for times less than 1000ms.

The hold level must be in the range 0 to 256, and represents the portion of time (out of 256) that the voltage is applied to the solenoid for each PWM cycle when holding.

Typically, the user will set the hold level substantially lower than the peak level.

4.8) Set TTL input parameters

```
int scl_set_ttl_input_parameters(int solenoid_id,
                                 int ttl_mode,
                                 int ttl_debounce_time)
```

This API function sets the TTL control values for the specified solenoid (in the range 0 to 11 for the SC12 solenoid controller board).

The TTL mode parameter must be in the range 0 to 4 and is enumerated as follows in the solenoid control library's header file:

```
#define SOLENOID_TTL_INPUT_DISABLED         0
#define SOLENOID_TTL_INPUT_EDGE_LOW_TO_HIGH 1
#define SOLENOID_TTL_INPUT_EDGE_HIGH_TO_LOW 2
#define SOLENOID_TTL_INPUT_LEVEL_HIGH       3
#define SOLENOID_TTL_INPUT_LEVEL_LOW        4
```

The debounce time parameter must be in the range 1 to 255 and specifies the number of one millisecond debounce increments that a TTL level must be held at in order for a valid level (transition) to be recognized. For instance, if a TTL input transitions from low-to-high and the debounce time is set to 3 milliseconds, then the controller will not activate the corresponding solenoid until the 3 millisecond period has elapsed (and the TTL level has be held steady for the 3 times it is sampled).

4.9) Set PWM mode

```
int scl_set_pwm_mode(int pwm_mode)
```

This API function sets PWM mode (frequency) for all solenoids. For compatibility with the SC5 solenoid controller, the PWM mode is enumerated in the solenoid control library's header file as follows:

```
#define PWM_MODE_24KHZ 0
#define PWM_MODE_8KHZ  1
```

For the SC12 solenoid controller, the frequency may be directly specified in the range 1000Hz to 30000Hz by setting the PWM mode argument to the frequency (in Hz).

Any change to the PWM mode is applied immediately regardless of whether any solenoids are on or not.

4.10) Get solenoid states

```
int scl_get_solenoid_states(int *state0,
                            int *state1,
                            int *state2,
                            int *state3,
                            int *state4,
                            int *state5=0,
                            int *state6=0,
                            int *state7=0,
                            int *state8=0,
                            int *state9=0,
                            int *state10=0,
                            int *state11=0)
```

This API function returns the states of each of the twelve solenoids in the corresponding integer variables at the addresses held in the pointers supplied by the calling function.

Each state will be 0 if the solenoid is off, or 1 if it is on.

NOTE: in order to preserve backwards compatibility with earlier versions of the solenoid controller library, states5 through state11 may be omitted from the function call and the corresponding solenoids states will not be returned.

```
int scl_get_solenoid_states_bitfield(int *state_bitfield)
```

For those users that prefer to store states in a bitfield, this function has been provided. The least significant bit (bit 0) stores the state of solenoid 0, bit 1 stores the state of solenoid 1, etc.

4.11) Get solenoid parameters

```
int scl_get_solenoid_parameters(int solenoid_id,
                                int *peak_level,
                                int *peak_time,
                                int *hold_level)
```

This API function gets the solenoid parameters for the specified solenoid (in the range 0 to 11 for the SC12 solenoid controller board).

The parameters are returned in the corresponding integer variables at the addresses held in the pointers supplied by the calling function. The parameters are identical to those of the complementary `scl_set_solenoid_parameters()` API function (see section 4.7)

4.12) Get TTL input parameters

```
int scl_get_ttl_input_parameters(int solenoid_id,
                                 int *ttl_mode,
                                 int *ttl_debounce_time)
```

This API function gets the TTL input parameters for the specified solenoid (in the range 0 to 11 for the SC12 solenoid controller board).

The parameters are returned in the corresponding integer variables at the addresses held in the pointers supplied by the calling function. The parameters are identical to those of the complementary `scl_set_ttl_input_parameters()` API function (see section 4.8)

4.13) Get TTL input states

```
int scl_get_ttl_input_states(int *ttl0,
                             int *ttl1,
                             int *ttl2,
                             int *ttl3,
                             int *ttl4,
                             int *ttl5=0,
                             int *ttl6=0,
                             int *ttl7=0,
                             int *ttl8=0,
                             int *ttl9=0,
                             int *ttl10=0,
                             int *ttl11=0)
```

This API function gets the instantaneous (non-debounced) TTL input states for all twelve TTL inputs.

The states are returned in the corresponding integer variables at the addresses held in the pointers supplied by the calling function. A state value of 0 represents a TTL low (0V) level and a state value of 1 represents a TTL high (3.3V) level.

NOTE: in order to preserve backwards compatibility with earlier versions of the solenoid controller library, ttl5 through ttl11 may be omitted from the function call and the corresponding TTL inputs will not be reported.

```
int scl_get_ttl_input_states_bitfield(int *ttl_bitfield);
```

For those users that prefer to store states in a bitfield, this function has been provided. The least significant bit (bit 0) stores the state of ttl 0, bit 1 stores the state of ttl 1, etc.

4.14) Get PWM mode

```
int scl_get_pwm_mode(int *pwm_mode)
```

This API function gets the PWM mode.

The PWM mode is returned in the corresponding integer variable at the address held in the pointer supplied by the calling function. The PWM mode is identical to that of the complementary `scl_set_pwm_mode()` API function (see section 4.9)

4.15) Get system voltages

```
int scl_get_system_voltages(int *voltage_1,int *voltage_2)
```

The voltages at the two power input connectors (J17 and J18) are monitored by the SC12 solenoid controller and may be read via this API function. Both voltages are returned in units of millivolts.

4.16) Save parameters

```
int scl_save_parameters(void)
```

This API function will cause the SC12 solenoid controller board to save the currently loaded solenoid parameters (see section 4.7), TTL input parameters (see section 4.8) and PWM mode (see section 4.9) to be stored in FLASH memory on SC12 solenoid controller board. These parameters then become the power-up default parameters for the SC12 solenoid controller board.

This feature is useful for those wishing to use the SC12 solenoid controller board via TTL control only. The parameters may be entered via the RS-232 port and then saved to FLASH memory so the TTL inputs are active on power up.

WARNING: The FLASH memory on the microcontroller used on the SC12 solenoid controller board has a limit of 10,000 erase-write cycles. Excessive calls to the `scl_save_parameters()` API function can wear out the FLASH memory and render it inoperative.

4.17) Restore default parameters

```
int scl_restore_default_parameters(void)
```

This API function restores the default parameters in FLASH memory to the factory default values (for all solenoids):

Peak level	256 (100%)
Peak time	250ms
Hold level	64 (25%)
TTL input	disabled
PWM mode	25.0KHz

WARNING: The FLASH memory on the microcontroller used on the SC12 solenoid controller board has a limit of 10,000 erase-write cycles. Excessive calls to the `scl_restore_default_parameters()` API function can wear out the FLASH memory and render it inoperative.

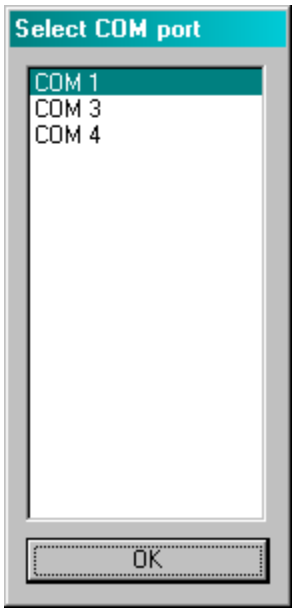
5) Sample Application

A sample application is provided that runs under Microsoft Windows 98/NT/2000/XP/Vista/7. The application provides convenient access to all of the solenoid controller's functions and can be used to manually control the solenoids as an aid to setting up a system for automated control of the solenoids.

The application may be compiled under Microsoft Visual C++ 6.0 and Visual Studio. The source code is provided as an example of how the solenoid control library may be used.

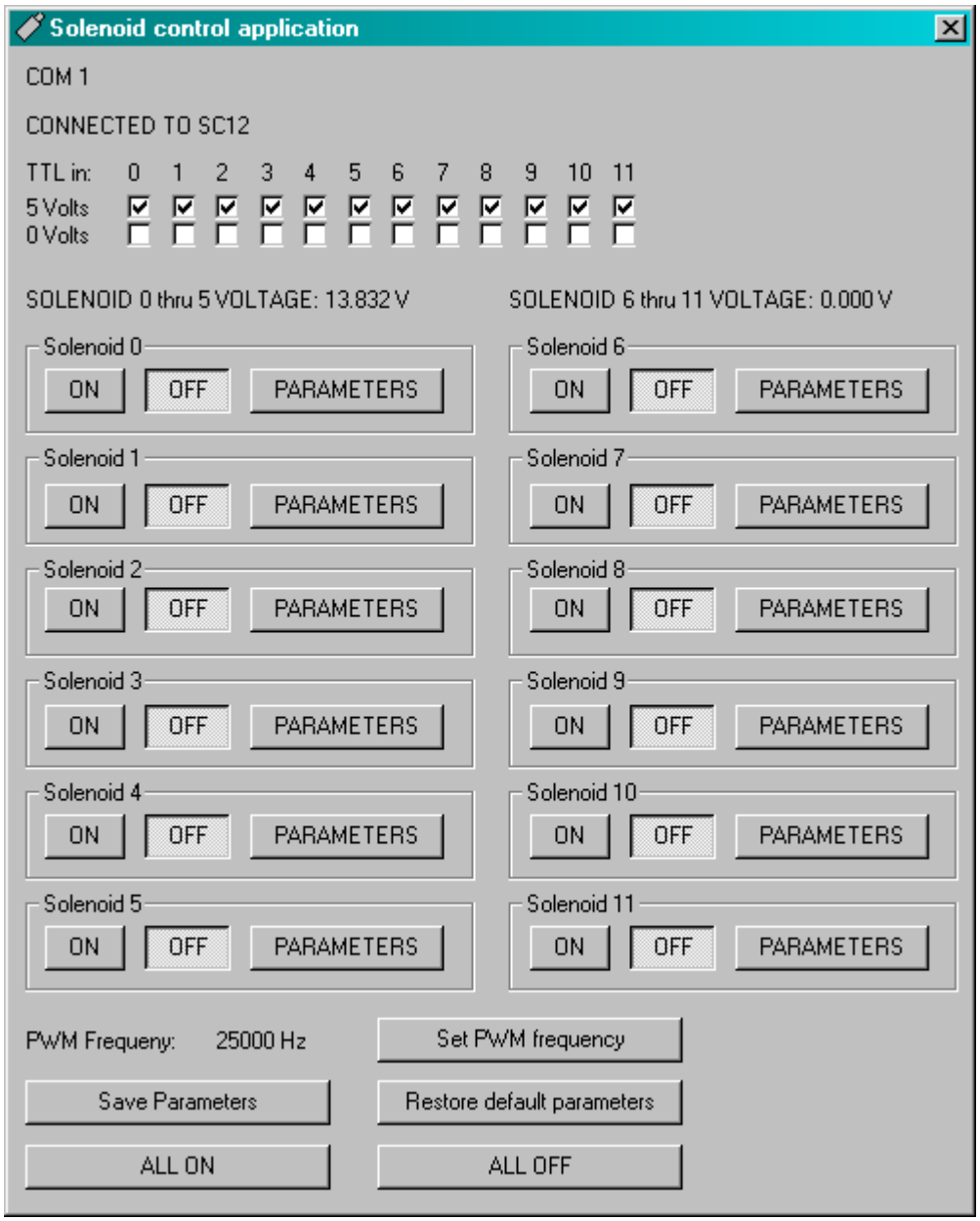
5.1) Communications port selection dialog

When the application is first started the computer is scanned to create a list of unused COM ports. The user is prompted to select a communications port using the following dialog. The user must select the communication port corresponding to the one the SC12 solenoid controller board is connected to, and the press “OK”.



5.2) Control dialog

After the communications port is selected by the user, the application will search for and initialize the SC12 solenoid controller board found on the port. The following control panel dialog is then displayed and remains until the user closes the application.



The top portion of the dialog will display the communications port currently in use, the port status, the instantaneous state of the TTL inputs, and the measured voltage for each bank of six solenoids. The display is updated approximately once per second.

The controls for controlling the solenoids are contained in separate identical group boxes (one per solenoid). The instantaneous solenoid state (on or off) is updated continuously at a rate of approximately once per second. The user may change the solenoid state by pressing the corresponding button, however the user-entered state can be overridden by the TTL inputs (if active).

See section 5.3 for a description of the dialog that appears when the “parameters” button is pressed.

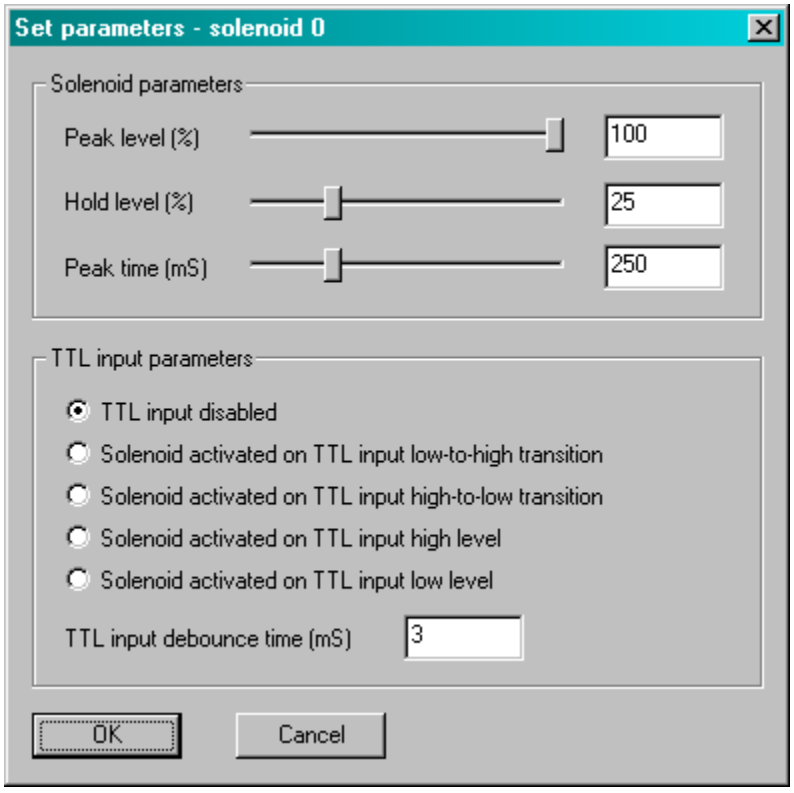
Pressing the “set PWM frequency” button will create a small dialog box that allows the user to change the PWM frequency within the range 1KHz to 30KHz.

The “save parameters” button allows the user to store the solenoid parameters and PWM frequency in FLASH memory on the SC12 solenoid controller board. The “restore default parameters” button reset all parameters to the factory defaults and stores them in FLASH memory.

The “all on” and “all off” buttons allow the user to simultaneously turn on or off all twelve solenoids.

5.3) Set parameters dialog

The following dialog appears when the “parameters” button for a solenoid is pressed. Hence the controls in this dialog apply only to the solenoid corresponding to the “parameter” button pressed.



The user may set the peak and hold level (as percentages, where 100% is when voltage is applied continuously). The peak time is specified in milliseconds.

The user may also select the TTL input mode and debounce count for the TTL input corresponding to the solenoid.

Pressing the “OK” button will cause the parameters to be registered. The TTL parameters will take effect immediately, the peak and hold levels will take effect the next time the solenoid is commanded to its ON state.

Pressing the “cancel” button will discard any changes made to the parameters.

6) Serial Command Set

For those users wishing to control the SC12 directly from their own system without the use of the solenoid control library, the serial command set supported by the SC12 is listed below.

All commands are ASCII text (human readable). Thus, a communications program such as "Hyperterminal" included with most Microsoft Windows operating systems, may be used to directly control the SC12.

Users of newer Windows operating systems that do not include Hyperterminal may download the PuTTY program from http://www.putty.org.

Below are screen shots of the Hyperterminal and PuTTY properties pages. Shown are the settings required to allow communications between the host PC and the SC12 solenoid controller.

The communications protocol is a simple command/response protocol. The host system sends a command to the SC12 and will then receive a response from the SC12. The SC12 will not initiate communications with the host PC.

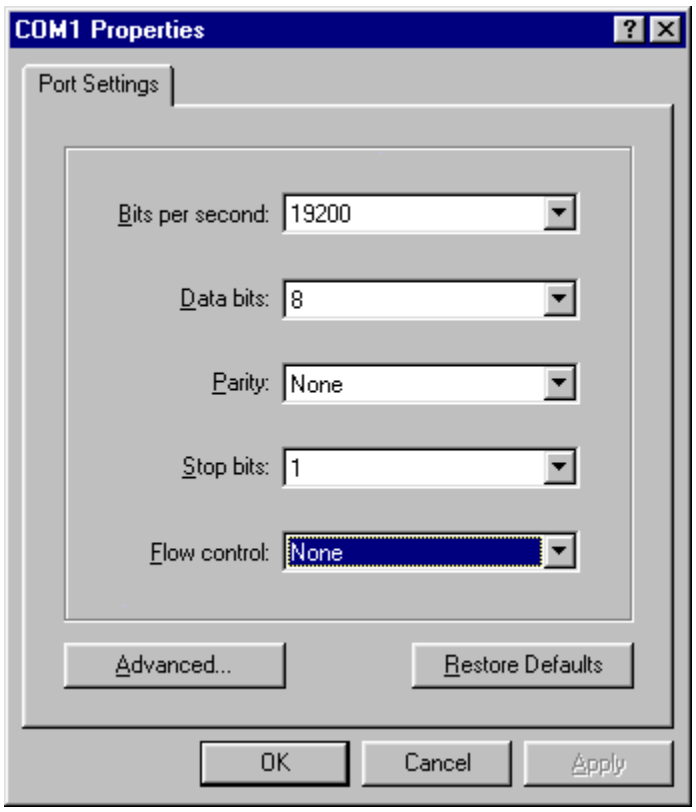
The command syntax is of the form:

<command character>[<parameter 1>][,<parameter 2>]…[,<parameters n>]<CR>

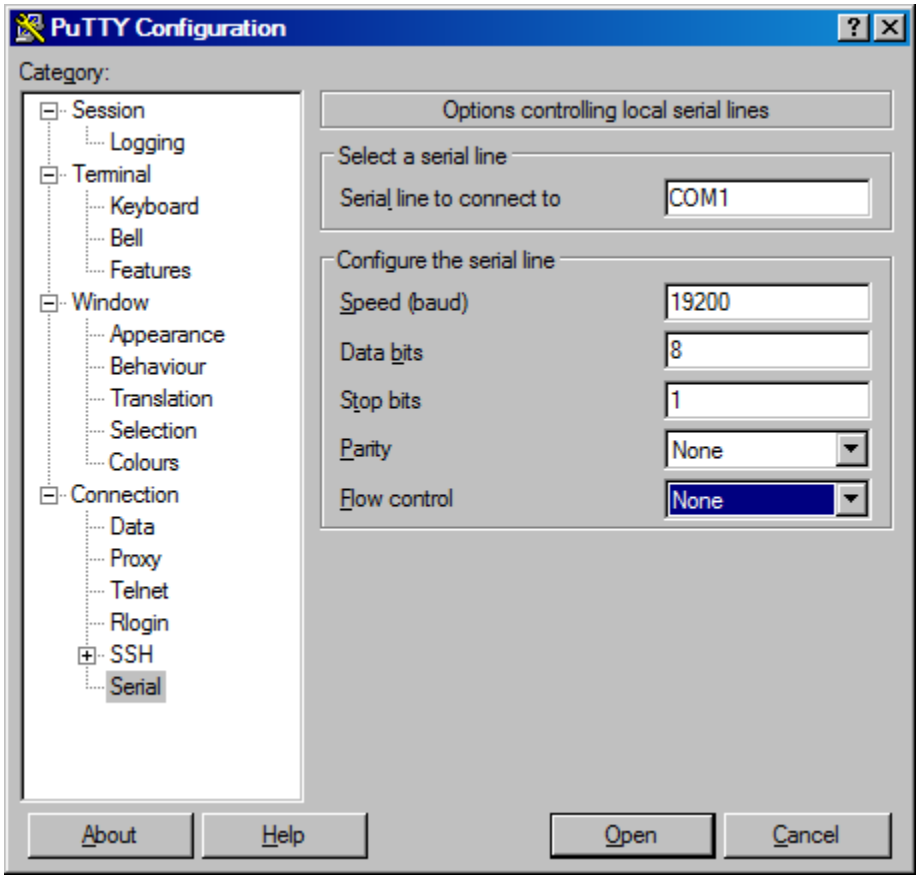
As noted, all commands must start with a single command character. Commands cannot be queued. All commands are terminated with a carriage return (hexidecimal 0x0D). Line feeds (hexidecimal 0x0A) are ignored. Illegal characters will cause the command to be ignored and an error to be returned.

All SC12 controller response strings are formatted in the same fashion as the command strings.

Sample Hyperterminal properties page



Sample PuTTY properties tab



6.1) Initialize

I

Initializes the controller with default parameters loaded from FLASH memory (see sections 4.16 and 4.17). The controller then acknowledges the command by echoing the 'I' command.

6.2) Set solenoid state

S<solenoid>,<state>

Sets the state of the solenoid specified by the <solenoid> parameter (0 through 11). The <state> parameter must be either 0 for OFF, or 1 for ON. When the solenoid is turned on, the peak-and-hold timing is reset. The controller responds with 'S'.

6.3) Set all solenoid states

A<state 0>,<state 1>,<state 2>,….,<state 10>,<state 11>

Set the states of all 12 solenoids simultaneously. Each state must be either 0 for OFF, or 1 for ON. The peak-and-hold timing for each solenoid turned on is reset. The controller responds with 'A'.

6.4) Set solenoid parameters

P<solenoid>,<peak level>,<peak time>,<hold level>

Sets the peak-and-hold parameters of the specified solenoid (0 through 11). The <peak level> and <hold level> parameters must be in the range 0 to 256 and represent the portion (out of 256) of the on-time for each PWM cycle (i.e. 256 is on all the time). The <peak time> parameter must be in the range 0 to 65535 and represents the number of milliseconds that the peak level is applied to the solenoid when it is commanded to the ON state. After this time has elapsed, the PWM level gets set to the hold level. The controller responds with 'P'.

6.5) Set TTL input parameters

T<solenoid>,<TTL mode>,<TTL debounce count>

Each of the twelve TTL inputs (0 through 11) controls the corresponding solenoid. The <TTL mode> must be one of the following:

0	Disabled (TTL does not control the solenoid)
1	Solenoid ON when TTL has a low-to-high transition Solenoid OFF when TTL has a high-to-low transition
2	Solenoid ON when TTL has a high-to-low transition Solenoid OFF when TTL has a low-to-high transition
3	Solenoid ON when TTL high Solenoid OFF when TTL low
4	Solenoid ON when TTL low Solenoid OFF when TTL high

Where LOW is 0V and HIGH is +3.3V. The TTL inputs are debounced at a 1ms rate. A TTL must be at a steady state for <TTL debounce count> milliseconds before a state transition can occur. <TTL debounce count> must be in the range 1 to 255. The controller responds to this command with 'T'.

6.6) Set PWM frequency

F<PWM frequency>

Set the PWM frequency from 1000 to 30000 Hz. The change in frequency takes place immediately. The controller responds to this command with ‘F’.

6.7) Get solenoid parameters

G<solenoid>

Gets the parameters for the specified solenoid (in the range 0 to 11). The controller responds with:

G<peak level>,<peak time>,<hold level>

Where the return parameters complement those of the 'P' command (see section 6.4).

6.8) Get TTL parameters

B<solenoid>

Gets the TTL parameters for the specified solenoid (in the range 0 to 11). The controller responds with:

B<TTL mode>,<TTL debounce count>

Where the return parameters complement those of the 'T' command (see section 6.5).

6.9) Get PWM frequency

H

Gets the PWM frequency mode. The controller responds with:

H<PWM frequency>

Where the return parameter complements that of the 'F' command (see section 6.6).

6.10) Get TTL inputs

R

Reads the TTL inputs. The controller responds with:

R<TTL 0>,<TTL 1>,<TTL 2>,…,<TTL 10>,<TTL 11>

Where <TTL n> is the instantaneous (non-debounced) value of the TTL input.

6.11) Get solenoid states

Q

Queries the state of the solenoids. The controller responds with:

Q<state 0>,<state 1>,<state 2>,…,<state 10>,<state 11>

Where <state n> is 0 for OFF and 1 for ON.

6.12) Get firmware version

V

Gets the firmware version from the SC12 controller. The controller responds with:

V<version>

Where <version> is the integer representation of the firmware version number.

6.13) Get model string

M

Gets the model number of the solenoid controller. The controller responds with:

MSC12

6.14) Get system voltages

N

Gets the instantaneous system voltages. The controller responds with:

N<voltage 1>,<voltage 2>

Where <voltage 1> is the voltage (in millivolts) for solenoids 0 through 5 and <voltage 2> is the voltage (in millivolts) for solenoids 6 through 11.

6.15) Save parameters

W

Saves the current parameters (for solenoids, TTL inputs, and PWM mode) into FLASH memory. This allows the system to power up with user-defined parameters that may be immediately invoked via TTL input or RS-232. The controller responds with ‘W’.

WARNING: The FLASH memory on the microcontroller used on the SC12 solenoid controller board has a limit of 10,000 erase-write cycles. Excessive use of the W command can wear out the FLASH memory and render it inoperative.

6.16) Restore default parameters

D

Sets the current parameters (for solenoids, TTL inputs, and PWM mode) back to the factory defaults and saves them in FLASH memory. The controller responds with ‘D’.

WARNING: The FLASH memory on the microcontroller used on the SC12 solenoid controller board has a limit of 10,000 erase-write cycles. Excessive use of the D command can wear out the FLASH memory and render it inoperative.

6.17) Errors

Errors in sending serial commands to the SC12 solenoid controller board will cause the controller to respond with an error string of the form E<e>, where <e> is the error number. The error numbers are enumerated below:

0 Invalid command – command character not recognized

1 Invalid format – command contained unexpected characters or incorrect number of parameters.

2 Parameter out of range – range checking is applied to most parameters to ensure that they are valid.