

RW Automation, LLC

SC5
5-channel solenoid controller/driver

USER'S GUIDE

Rev 1.2
October 2009



www.rwautomation.com

Table of Contents

1) Legal Notices 3
 1.1) Copyright 3
 1.2) License agreement 3
 1.3) Limited warranty 3
 1.4) Third-party trademarks 4
 2) Introduction 5
 2.1) Packing list 5
 2.2) Features 5
 3) Configuration 7
 3.1) Installing the software 9
 3.2) Host computer communications 10
 3.4) Connecting power 11
 3.5) Connecting solenoids 12
 3.6) Connecting TTL control lines 14
 3.7) Status LEDs 16
 3.8) Mechanical specifications 17
 4) Solenoid Control Library 18
 4.1) Get library version 19
 4.2) Initialize 20
 4.3) De-initialize 21
 4.4) Set solenoid state 22
 4.5) Set all solenoid states 23
 4.6) Set solenoid parameters 24
 4.7) Set TTL input parameters 25
 4.8) Set PWM mode 26
 4.9) Get solenoid states 27
 4.10) Get solenoid parameters 28
 4.11) Get TTL input parameters 29
 4.12) Get TTL input states 30
 4.13) Get PWM mode 31
 4.14) Save parameters 32
 4.15) Restore default parameters 33
 5) Sample Application 34
 5.1) Communications port selection dialog 35
 5.2) Control dialog 36
 5.3) Set parameters dialog 38
 6) Serial Command Set 39
 6.1) Initialize 41
 6.2) Set solenoid state 42
 6.3) Set all solenoid states 43
 6.4) Set solenoid parameters 44
 6.5) Set TTL input parameters 45
 6.6) Set PWM mode 46
 6.7) Get solenoid parameters 47
 6.8) Get TTL parameters 48
 6.9) Get PWM mode 49
 6.10) Get TTL inputs 50
 6.11) Get solenoid states 51
 6.12) Get firmware version 52
 6.13) Save parameters 53
 6.14) Restore default parameters 54
 6.15) Errors 55

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.

Copyright © 2009 RW Automation, LLC - All rights reserved

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 SC5 solenoid controller board are intended for use only with the SC5 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 SC5 solenoid controller board may be altered and recompiled into new executables. Such executables based on the original software packaged with the SC5 solenoid controller board may be freely distributed for resale along with the SC5 solenoid controller board, but may otherwise not be distributed without the SC5 solenoid controller board.

1.3) Limited warranty

RW Automation, LLC warrants the SC5 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 the existence and cause of defect.

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” and “Microsoft Windows” are registered trademarks of Microsoft Corporation.

2) Introduction

Congratulations on your purchase of the SC5 solenoid controller board. Please take the time to familiarize yourself with the material and features of the SC5 solenoid controller board package before proceeding to the next section.

2.1) Packing list

When you purchase the SC5 solenoid controller board, you should receive the following items:

- a) The SC5 solenoid controller board
- b) Five 2-position, 3.81mm Phoenix terminal blocks (one for each solenoid)
- c) One 2-position, 7.62mm Phoenix terminal block (for power)
- d) One 7-position, 3.81mm Phoenix terminal block (for TTL inputs)
- e) CD-ROM (with solenoid control library and sample application)
- f) RS-232 serial cable

2.2) Features

The SC5 solenoid controller board controls five independent solenoids. By utilizing the SC5 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 SC5 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:

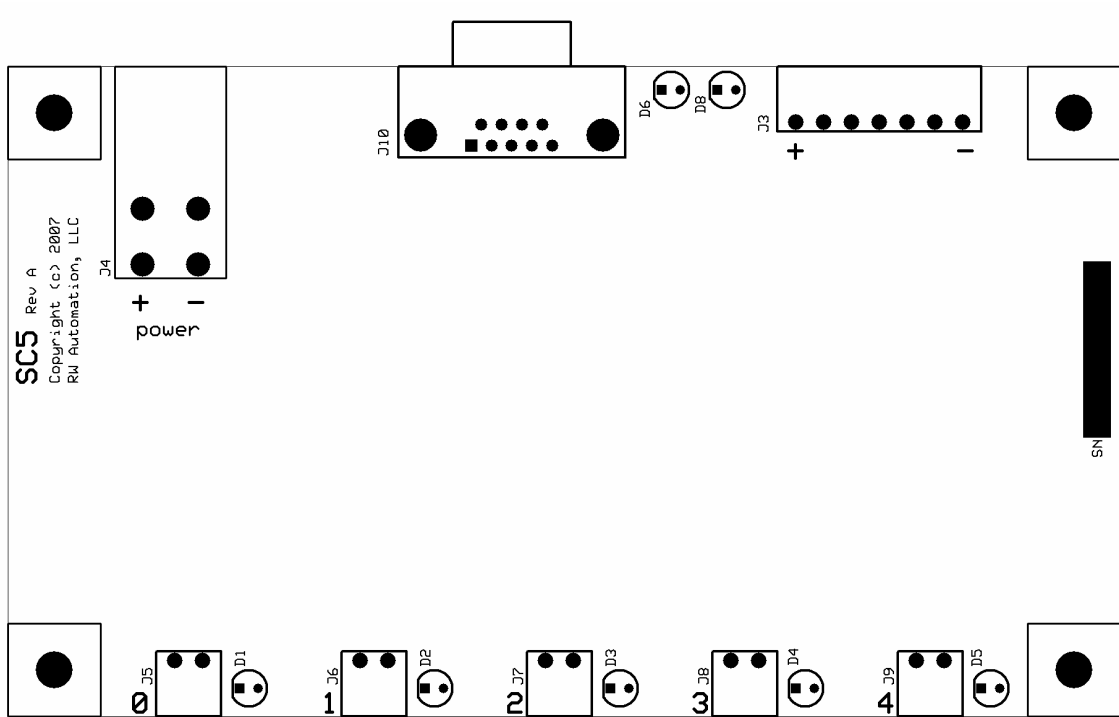
- a) RS-232 port to connect to a host computer. The baud rate is fixed at 19200.
- b) Concise ASCII command set.
- c) Allows both serial and TTL control of solenoids.
- d) User programmable PWM frequency (23.4KHz or 7.8KHz).
- e) User programmable peak PWM levels (independent for each solenoid).
- f) 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 7-24VDC.
- k) Detachable terminal block connectors for quick connect/disconnect.
- l) Power, communication, 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 is included at no additional charge.

3) Configuration

This section describes the steps needed to set up the SC5 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

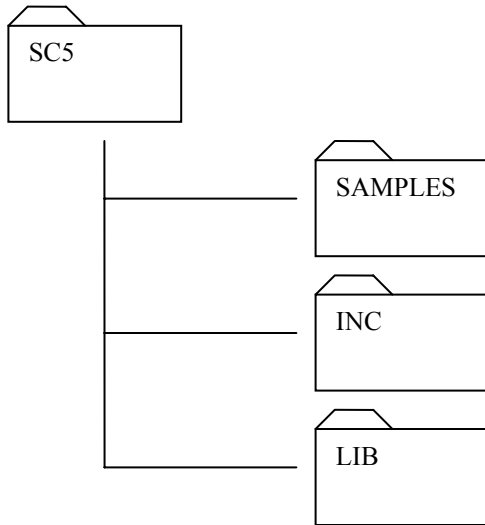
The connectors and LEDs shown in figure 1 are listed in the following table:

Component	Description
J3	TTL input connector
J4	7 to 24 VDC connector
J5	Solenoid 0 connector
J6	Solenoid 1 connector
J7	Solenoid 2 connector
J8	Solenoid 3 connector
J9	Solenoid 4 connector
J10	RS-232 serial port connector
D1	Solenoid 0 status LED
D2	Solenoid 1 status LED
D3	Solenoid 2 status LED
D4	Solenoid 3 status LED
D5	Solenoid 4 status LED
D6	Communication status LED
D8	Power LED

3.1) Installing the software

The SC5 software runs under Microsoft Windows 98/2000/XP operating systems. It is distributed as a single self-extracting setup executable file. Simply run the setup file and follow the instructions. The SC5 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.

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

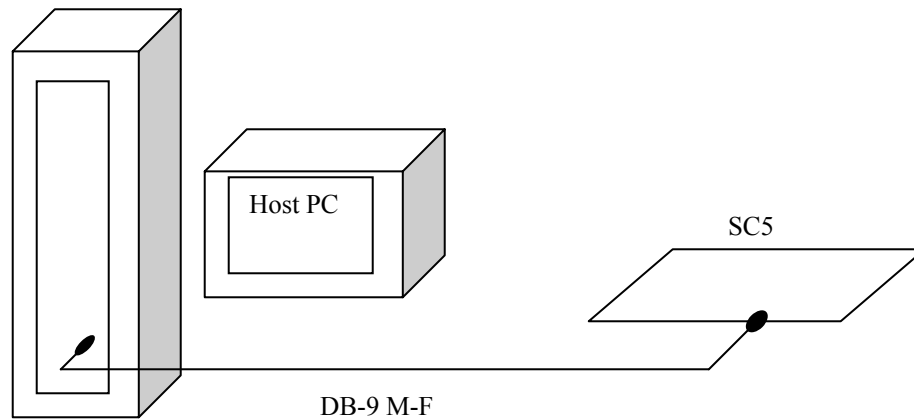
Figure 2 below illustrates an example of how to connect the SC5 solenoid controller board (the “slave”) to a host PC (the “master”). A 9-pin female-male cable wired straight through is appropriate for most recently manufactured PCs, however, older PCs and other types of computers may have different connectors (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 SC5 solenoid controller board.

Below is the pinout of the RS-232 connector of the SC5 solenoid controller board used to connect the SC5 solenoid to a host PC. Note that flow control lines are not supported by the SC5 solenoid controller board.

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 SC5 connection




3.4) Connecting power

The SC5 solenoid controller board requires a single voltage source in the range 7 to 24 VDC. In addition to supplying power to the solenoids, an on-board switching regulator is used to supply the +3.3 VDC needed for the microprocessor. This +3.3VDC is also available to the user in limited quantities via the TTL connector.


Power connector pinout	
Pin	Function
1	Ground
2	+7 to +24 VDC

If the five solenoids are to be run simultaneously at the maximum current (4 Amperes each), then the supply must be able to support a minimum of 21 Amperes of continuous current.

Figure 3 below shows the 2-pin power connector illustrating how to connect power to the SC5 solenoid controller board.

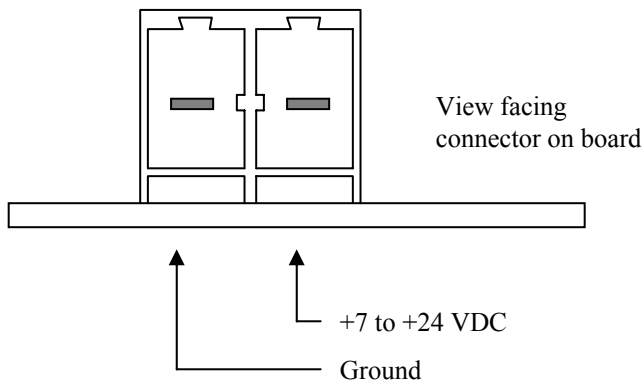


WARNING: Please note carefully the required polarity of the power supply connector. Applying power of the incorrect polarity can damage the SC5 solenoid controller board and will void the warranty.



WARNING: Applying voltages outside of the required 7-24 VDC range can damage the SC5 solenoid controller board and will void the warranty.

Figure 3 – Power connector

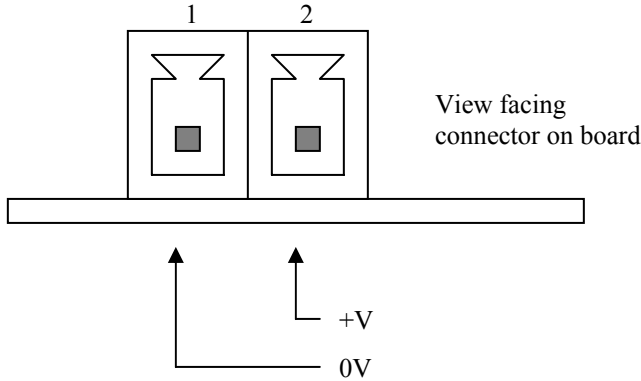


3.5) Connecting solenoids

Each solenoid is assigned a logical number from 0 to 4 as shown in figure 1. All five solenoid channels are wired identically. Figure 4 illustrates the solenoid connector and the pinout is shown in the table below.

Pin	Function
1	0V
2	+V

Figure 4 – Solenoid connector



WARNING: Shorting the outputs of the solenoid connector or otherwise driving excessive current can damage the SC5 solenoid controller board and will void the warranty.

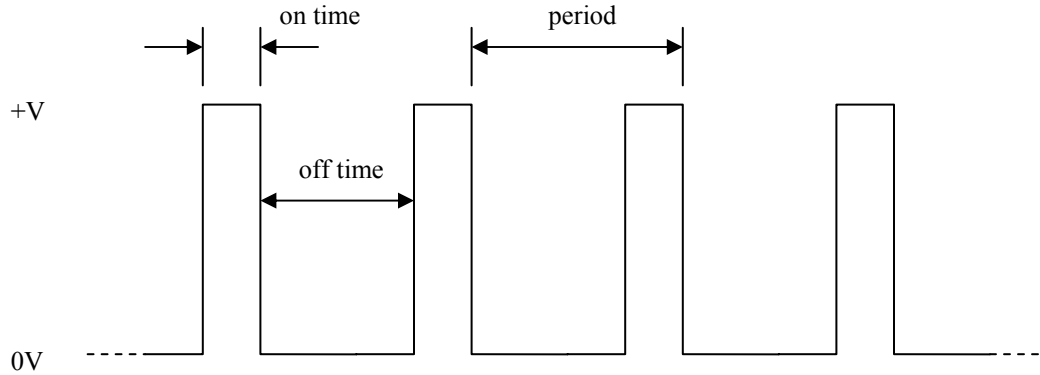


WARNING: The heatsinks on the SC5 solenoid controller board can get very hot. Avoid touching the heatsinks. It is recommended that forced air be directed at the heatsinks in order to increase their ability to dissipate heat.

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 5.

Figure 5 – 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 that average current is maintained at or below 4A per solenoid.

3.6) Connecting TTL control lines

The SC5 solenoid controller board has a 5-input TTL connector. The pinout for the connector is shown in below.

Limit switch connector pinout

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

The TTL inputs use 3.3V gates (5V tolerant). Do not exceed the 0-5VDC input range of the TTL inputs. The TTL inputs are pulled high on the SC5 board.

Each TTL input is mapped (logically) to the corresponding solenoid on the SC5 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 level 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 level 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 this connector for users that need to power circuits used with the TTL interface. **Note:** no more than 100 milliamperes should be drawn from the +3.3 VDC pin of this connector.

3.7) Status LEDs

The SC5 solenoid controller board has seven status LEDs (refer to figure 1).

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

The communications LED will initially be off when the SC5 solenoid controller board is powered up. As soon as communication with the board has been established, the LED will turn on. The LED will flash when the SC5 boards is actively receiving characters via the RS-232 interface.

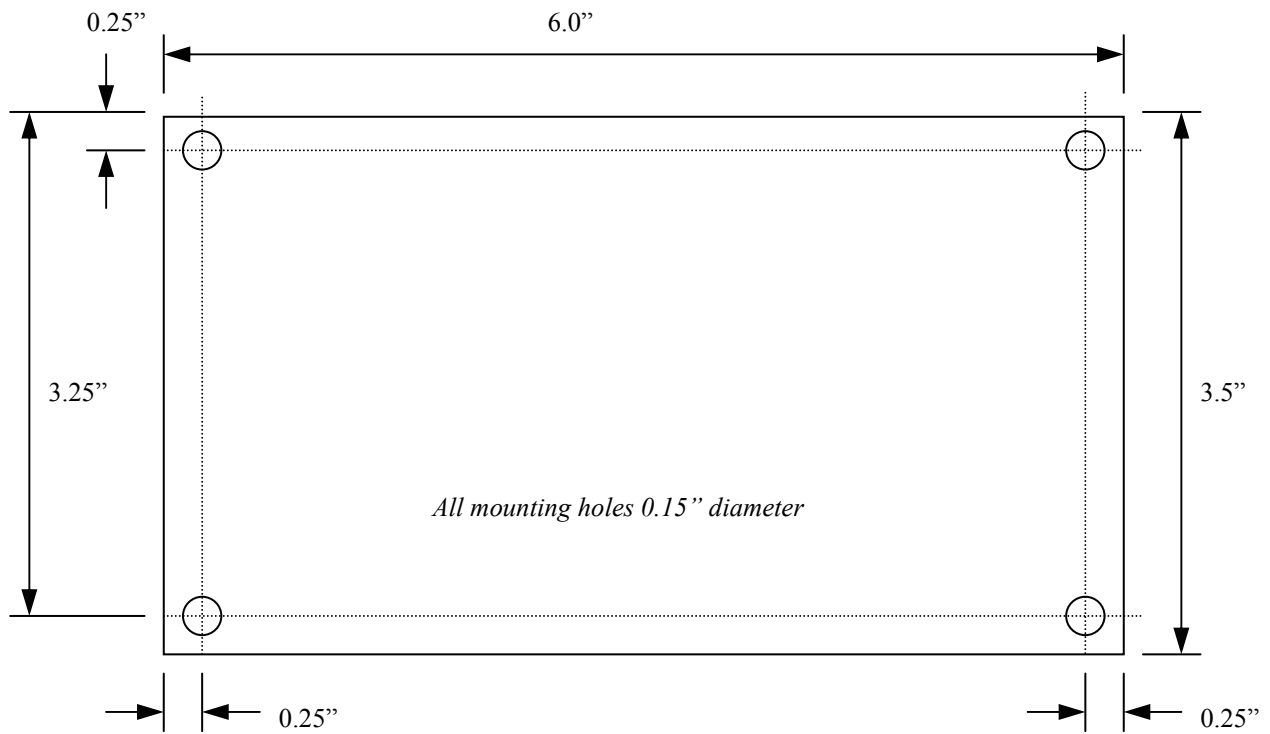
The solenoid status LEDs will be on whenever there is a current being applied to the corresponding solenoid. The solenoid state LEDs are off when no current is being applied to the corresponding solenoid.

3.8) Mechanical specifications

The SC5 solenoid controller board is 6.0" x 3.5" (152.4mm x 88.9mm). The height of the heatsinks is 1.5" (38.1 mm).

Mounting hole locations are shown in figure 6 below.

Figure 6 – 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.

These functions provide an application programming interface (API) for the solenoid controller without the need to learn the SC5 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 Visual C++ 6.0.

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 SC5 solenoid controller board fails. This will normally only occur if there is a problem with the connection between the host computer and the SC5 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) 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 SC5 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 have no effect other than to simply return a “no error” status. If the user wants to specify a different com port at runtime, then the `scl_deinitialize()` function must be called prior to calling the `scl_initialize()` function again with the new com port parameter.

4.3) 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 SC5 solenoid controller board will be re-initialized and the library will return to its uninitialized.

4.4) Set solenoid state

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

This API function sets the state for the specified solenoid (in the range 0 to 4 for the SC5 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 3.5) is reset (even if the solenoid was already on).

4.5) Set all solenoid states

```
int scl_set_all_solenoid_states(unsigned short int state0,  
                                unsigned short int state1,  
                                unsigned short int state2,  
                                unsigned short int state3,  
                                unsigned short int state4)
```

This API function sets the states for all five 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 3.5) is reset (even if the solenoid was already on).

4.6) Set solenoid parameters

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

This API function sets the peak-and-hold parameters for the specified solenoid (in the range 0 to 4 for the SC5 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.

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.

4.7) Set TTL input parameters

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

This API function sets the TTL control values for the specified solenoid (in the range 0 to 4 for the SC5 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 0 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 been held steady for the 3 times it is sampled).

4.8) Set PWM mode

```
int scl_set_pwm_mode(unsigned short int pwm_mode)
```

This API function sets PWM mode (frequency) for all solenoids. 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
```

Thus the `pwm_mode` parameter must be in the range 0 to 1. Any change to the PWM mode is applied immediately regardless of whether any solenoids are on or not.

4.9) Get solenoid states

```
int scl_get_solenoid_states(unsigned short int *state0,  
                           unsigned short int *state1,  
                           unsigned short int *state2,  
                           unsigned short int *state3,  
                           unsigned short int *state4)
```

This API function returns the states of each of the five solenoids in the corresponding short 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.

4.10) Get solenoid parameters

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

This API function gets the solenoid parameters for the specified solenoid (in the range 0 to 4 for the SC5 solenoid controller board).

The parameters are returned in the corresponding short 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 4.6)

4.11) Get TTL input parameters

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

This API function gets the TTL input parameters for the specified solenoid (in the range 0 to 4 for the SC5 solenoid controller board).

The parameters are returned in the corresponding short 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 4.7)

4.12) Get TTL input states

```
int scl_get_ttl_input_states(unsigned short int *ttl0,  
                             unsigned short int *ttl1,  
                             unsigned short int *ttl2,  
                             unsigned short int *ttl3,  
                             unsigned short int *ttl4)
```

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

The states are returned in the corresponding short 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 (5V) level.

4.13) Get PWM mode

```
int scl_get_pwm_mode(unsigned short int *pwm_mode)
```

This API function gets the PWM mode.

The PWM mode is returned in the corresponding short 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 4.8)

4.14) Save parameters

```
int scl_save_parameters(void)
```

This API function will cause the SC5 solenoid controller board to save the currently loaded solenoid parameters (see section 4.6), TTL input parameters (see section 4.7) and PWM mode (see section 4.8) to be stored in FLASH memory on SC5 solenoid controller board. These parameters then become the power-up default parameters for the SC5 solenoid controller board.

This feature is useful for those wishing to use the SC5 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 SC5 solenoid controller board has a limit of 20,000 erase-write cycles. Excessive calls to the `scl_save_parameters()` API function can wear out the FLASH memory and render it inoperative.

4.15) 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	23.4KHz



WARNING: The FLASH memory on the microcontroller used on the SC5 solenoid controller board has a limit of 20,000 erase-write cycles. Excessive calls to the `scl_restore_default_parameters()` API function can wear out the FLASH memory and render it inoperative.

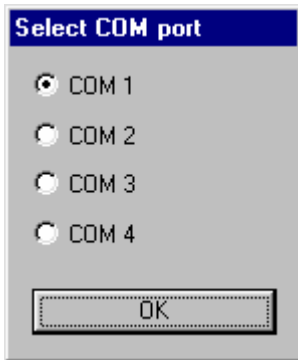
5) Sample Application

A sample application is provided that runs under Microsoft Windows 98/NT/2000/XP. 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. The source code is provided as an example of how the solenoid control library may be used.

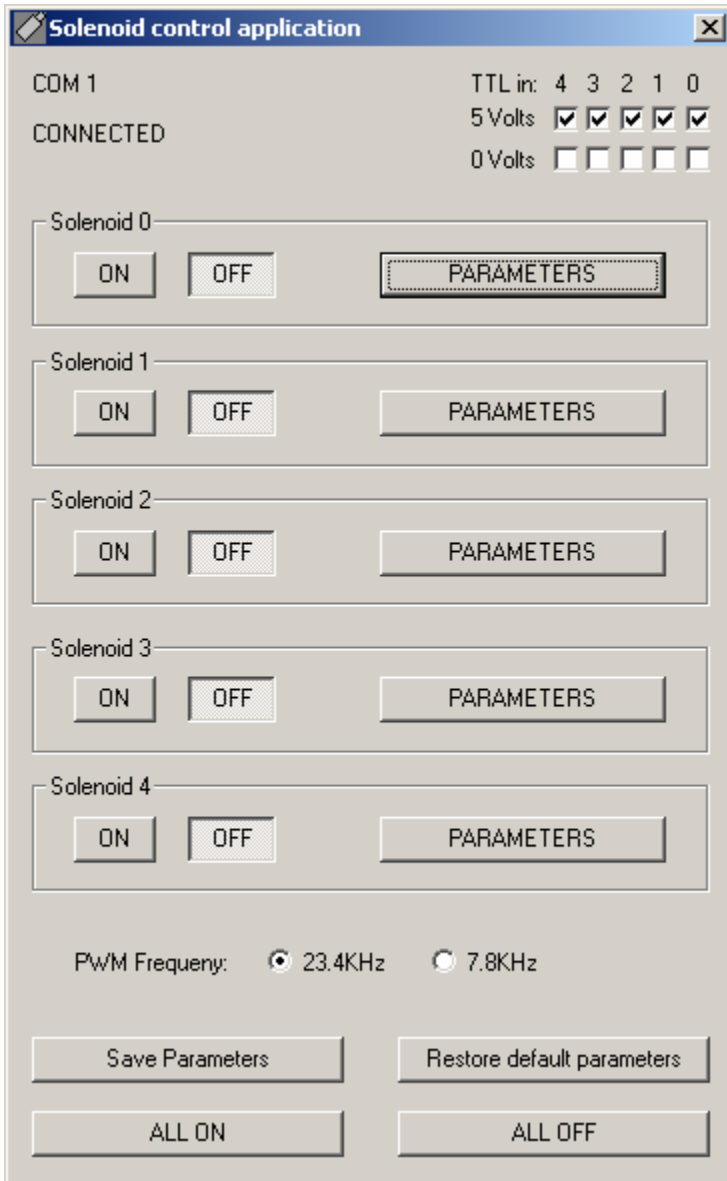
5.1) Communications port selection dialog

When the application is first started the user is prompted to select a communications port using the following dialog. The user must select the communication port corresponding to the one the SC5 solenoid controller board is connected to, and then press “OK”.



5.2) Control dialog

After the communications port is selected by the user, the application will search for and initialize the SC5 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, and the instantaneous state of the TTL inputs (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, but 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.

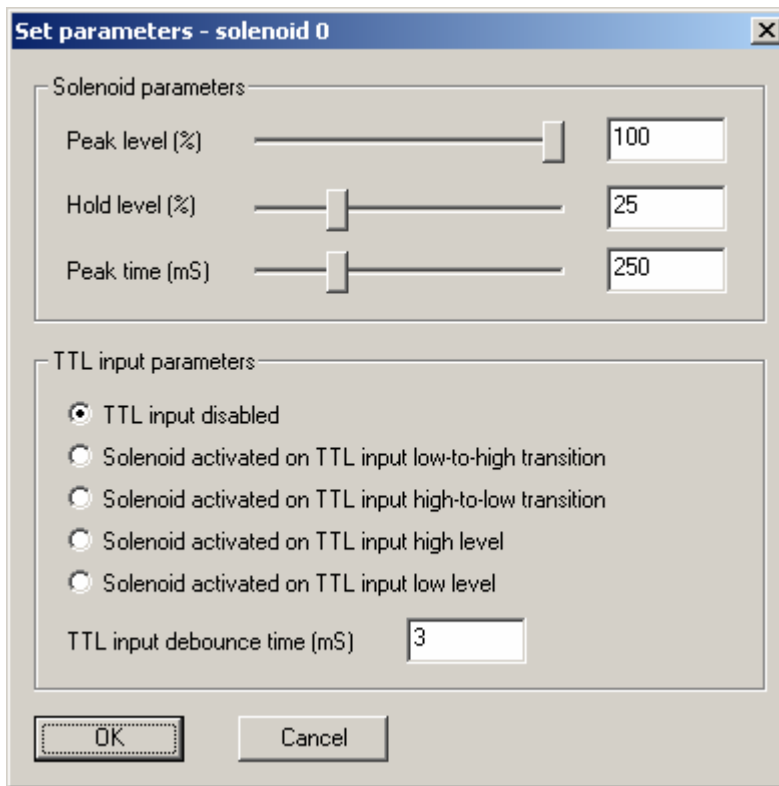
A radio control allows the user to change the PWM mode.

The “save parameters” button allows the user to store the solenoid parameters and PWM mode in FLASH memory on the SC5 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 five 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.

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 command 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 SC5 directly from their own system without the use of the solenoid control library, the serial command set supported by the SC5 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 SC5.

Below is a screen shot of the Hyperterminal properties page. Shown are the settings required to allow communications between the host PC and the SC5 solenoid controller.

The communications protocol is a simple command/response protocol. The host system sends a command to the SC5 and will then receive a response from the SC5. The SC5 will not initiate communications with the host PC.

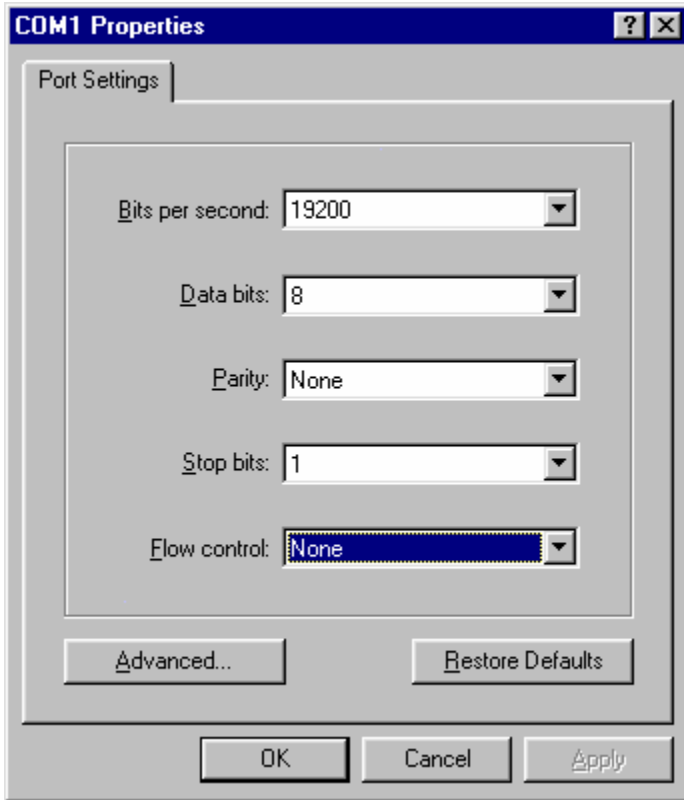
The command syntax is of the form:

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

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

All SC5 controller response strings are formatted in the same fashion as the command strings.

Sample Hyperterminal properties page



6.1) Initialize

I

Initializes the controller with default parameters loaded from FLASH memory (see section 4.15). 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 4). 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 3>,<state 4>

Set the states of all 5 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 4). 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 five TTL inputs (0 through 4) 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 +3.3 to +5V. 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 0 to 255. The controller responds to this command with T.

6.6) Set PWM mode

F<PWM mode>

Set the PWM mode (frequency) according to the following enumeration:

- 0 PWM frequency set to 23.4 KHz
- 1 PWM frequency set to 7.8 KHz

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 4). 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 4). 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 mode

H

Gets the PWM frequency mode. The controller responds with:

H<PWM frequency mode>

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 3>,<TTL 4>

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 3>,<state 4>

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

6.12) Get firmware version

V

Gets the firmware version from the SC5 controller. The controller responds with:

V<version>

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

6.13) 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 SC5 solenoid controller board has a limit of 20,000 erase-write cycles. Excessive use of the W command can wear out the FLASH memory and render it inoperative.

6.14) Restore default parameters

D

Sets the current parameters (for solenoids and TTL inputs) 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 SC5 solenoid controller board has a limit of 20,000 erase-write cycles. Excessive use of the D command can wear out the FLASH memory and render it inoperative.

6.15) Errors

Errors in sending serial commands to the SC5 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.