

Communications with the SQM-160

There are three possibilities for communicating with the SQM-160. The first is a DOS-based terminal program. The next is a 32 bit Windows DLL. The third is a Windows program that can collect data from the SQM-160 and save it in Excel spreadsheet format.

SQM-TERM.EXE

This simple DOS-based terminal program allows you to send a command and receive a response via the computer's RS-232 port. The computer's COM1 port baud rate must be set to match the SQM-160 before the program is run. Source code (SQM-TERM.C) is provided, so it can be customized easily. The command and response strings are detailed in the attached Communications Protocol.

SIGMACOM.DLL

This 32 bit DLL should be placed in your Windows System directory. No Windows registration is needed. The first call should be to InitCom to establish the comm port and baud rate. Functions and their calling convention are listed in the attached SIGMACOM.DLL Functions document.

MONCOMM.EXE

This program uses SIGMACOM.DLL to communicate with the SQM-160. It must be installed in Windows using the Setup MonComm installation program. It allows you to set film parameters and names, download them to the SQM-160, and collect data from the instrument. The data can be graphed, and also saved in a spreadsheet format. This program is written in Visual Basic, contact Sigma Instruments if you need the source code for this program.

SQM-160 Communications Protocol

The SQM-160 communicates with a host computer via an ASCII based protocol. The instrument defaults to 19200 baud, 8 data bits, and no parity. The baud rate can be changed in the System Menu of the SQM-160, but is always 8 data bits with no parity.

The basic protocol is:

<sync character> <length character> <1 to n data characters> <CRC1><CRC2>

Once a valid command has been transmitted to the SQM-160, a response is returned. The structure of the packet is identical in both the command and response. In the response, the first character is a Response Status. These are summarized in the following table.

Response Letter	Meaning
A	Command understood, normal response
B	Command understood, but instrument reset
C	Invalid command
D	Problem with data in command
E	Instrument in wrong mode for this command

The sync character is an exclamation point '!'. Anytime this character is received, the communications for that packet is reset. Following the sync character is the length character. This is the number of characters in the packet starting with the length and counting the 2 CRC characters. This character has a decimal 34 added to it so there cannot accidentally be a sync character embedded in the packet. The two character CRC is computed using the following algorithm:

1. The CRC is initialized to 3FFF hex.
2. Each character in the message is examined, bit by bit, and added to the CRC in the following manner:
 - a) The character is exclusive or'd with the CRC.
 - b) The CRC is shifted right one bit position.
 - c) If the character's least significant bit is a 0 then the CRC is exclusive or'd with 2001 hex.
 - d) Steps b and c are repeated for each of the 8 bits in the character.

The CRC contains 14 significant bits. This is split into two characters of 7 bits each, and then a decimal 34 is added to offset the character outside the range of the Sync Character. See the code example in the SQM-TERM.C file for an example of managing the CRC.

Command: @

Parameters: None

Description: Returns the model number and software version number.

Example: @ AMON Ver 2.01

Command: A

Parameters: [1..9], Values | ?

Description: Film parameters. The parameters available for change or inspection are

Label, Density, Tooling, Z-Factor, Final Thickness, Thickness Setpoint, Time Setpoint, Sensor Average

The parameters are sent/retrieved in that order. The label is a maximum of 8 characters, and is terminated by a space character. If you want to send a space embedded in a Label, use an underscore character '_'. Each parameter is separated by a space.

Each film's parameters are accessed by using the film's number directly after the Command. The parameters are edited by adding a value after the command film number.

The parameters are inspected by issuing a command, film number, then a question mark.

Example: A4LENS_1 6.23 125 1.05 1.525 0.450 30 1
A4? ALENS 1 6.23 125 1.05 1.525 0.450 30 1

Command: B

Description: System 1 parameters. The parameters available for change or inspection are Time Base, Simulation Mode, Frequency Mode, Rate Resolution, Rate Filter, Crystal Tooling and the parameters are sent/retrieved in that order.

Example: B? A0.25 0 0 0 8 100 100 100 100 100 100

Command: C

Description: System 2 parameters. The parameters available for change or inspection are Minimum Frequency, Maximum Frequency, Minimum Rate, Maximum Rate, Minimum Thickness, Maximum Thickness, Etch Mode and the parameters are sent/retrieved in that order.

Example: C? 5.000 6.000 0.000 100.00 0.000 1.000 0

Command: D

Parameters: 1 to 9

Description: Sets the active film.

Example: D1 Set the active film to Film #1

Command: J

Parameters: None.

Description: Read the number of channels installed. The number of channels will be either an ASCII two or six.

Example: J A2 The unit has two channels available.

Command: L

Parameters: [1..6]

Description: Read the current Rate for a channel.

Example: L1 A9.32 Channel one's rate is 9.32 Angstroms/S

Command: M

Parameters: None.

Description: Read the current Average Rate.

Example: M A10.42 Average Rate is 10.42 Angstroms/S

Command: N

Parameters: [1..6]

Description: Read the curent thickness for a channel.

Example: N4 A1.187 Channel four's Thickness is 1.187 Kilo Angstroms.

Command: O

Parameters: None.

Description: Read the current Average Thickness

Example: O A2.376 The current Average Thickness is 2.376 kilo Angstroms.

Command: P

Parameters: [1..6]

Description: Read the current Frequency for a channel.

Example: P2 A5701563.2 Channel two's current Frequency 5701563.2Hz

Command: R

Parameters: [1..6]

Description: Read the Crystal Life for a channel.

Example: R3 A57.82 Channel three's remaining life is 57.82%.

Command: S

Parameters: None.

Description: Zero Average Thickness and Rate.

Example: S A

Command: T

Parameters: None.

Description: Zero Time

Example: T A Zeroes time display on unit.

Command: U

Parameters: 0,1, or ?

Description: Toggles shutter open/closed or reads shutter state.

Example: U1 A Shutter is opened
U? A1 Shutter Status is open
U0 A Shutter is closed.

Command: Y

Parameters: None.

Description: Read the Power-Up Reset flag. The Power-Up Reset flag is set during boot-up of the unit and stays set until read through the RS-232 interface. After the flag is read, it is reset and will not be set again until the unit is power cycled.

Example: Y A1 Power-Up Reset flag is set.
Y A0 Power-Up Reset flag is reset.

Command: Z

Parameters: None.

Description: Set all Film and System parameters to defaults.
Note that this command can take over 1 second to complete

Example: Z A All Film and System parameters are set to defaults.

SIGMACOM.DLL Function Descriptions

This dll acts as an interpreter between an application and the SQM160. The dll transforms function calls to specific command sequences that the unit understands.

Transfer of data to the unit, in general, requires two function calls. The first function call is to transfer the data to the unit. The data to be sent is usually contained in the function's parameter(s). The second function call is to *ChkCommDone*. This function call ensures that the data was sent properly to the unit.

Data retrieval requires three function calls. The first function call is used to tell the unit what data is being requested. The second function call is to *ChkCommDone*. This function call is used to determine when all of the data has been transferred from the unit to the dll or if an error occurred in the communications. The third function call is used to retrieve the data from the dll.

InitComm

Parameters: 16 Bit Integer, 32 Bit Integer

Return : 16 Bit Integer.

InitComm is used to initialize the dll com port. The function's first parameter is the com port number to initialize (1 - 99 are valid). The second parameter is the baud rate for the port. The function returns zero if initialization was successful or a bit flag to indicate the failure of the initialization :

- bit 0 : Communications Port handle is invalid.
- bit 1 : Communications Port Set parameters invalid (Baud Rate)
- bit 2 : Communications Port Set timeouts invalid.
- bit 3 : Communications Port Set mask invalid.
- bit 4 : Communications Port Error – Already exists.
- bit 5 : Communications Port Set Read Thread fail.
- bit 6 : Communications Port Set Read Thread priority fail.

Example:

```
RetVal =InitComm(1,19200)  initialize Com1 to 19200 baud
if (RetVal != 0)          if port did not initialize correctly
    CloseComm()           close the port
```

ClearComm

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

ClearComm is used to clear the communications buffers in the dll.

Example: RetVal =ClearComm() Clear the comm buffers in the dll

CloseComm

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

CloseComm is used to close the currently opened communications port. *CloseComm* should always be used before attempting to open another port or before exiting the dll's calling application. The dll can have only one port open at a time.

Example:

ReturnVal =CloseComm() Close the currently open comm port

ChkCommDone

Parameters: None.

Return : 16 Bit Integer.

ChkCommDone is used to check the status of a single communications iteration. The function returns one of five different types of values:

-1: communications not complete

Positive integer : communications complete, value is byte count of returned message.

-99 : communications complete, but return message incomplete due to timeout with unit.

-98 : communications complete, but return message not valid due to a CRC error.

-97 : communications complete, but message not understood by unit.

Example:

ReturnVal =ChkCommDone() check communications status

SendGetVers

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetVers is used to retrieve the software version of the unit from the unit. This function must precede the use of the *GetVers* function

GetVers

Parameters: Pointer to Null-Terminated string.

Return : 16 Bit Integer, always returns a 1.

GetVers is used to retrieve the software version of the unit from the dll. This function must be preceded by the *SendGetVers*. The Null-terminated string is used to return the version from the dll.

Example:

```
ReturnVal = SendGetVers()           tell unit to transfer version to dll
do while(ChkCommDone == -1)        wait for comm to finish
ReturnVal = GetVers(&VersionString[0]) VersionString contains version
                                     info
```

Set160Film

Parameters: Pointer to a Film Structure.

Return : 16 Bit Integer, always returns a 1.

SetFilm is used to set a Film's parameters in the unit. All of the parameters are passed to the function through the Film Structure.

Example:

```
ReturnVal = SetFilm(&FilmStruct)    set film parameters to FilmStruct
                                     values
do while(ChkCommDone == -1)        wait for comm to finish
```

SendGetFilm

Parameters: 16 Bit Integer.

Return : 16 Bit Integer, always returns a 1.

SendGetFilm is used to get a Film's parameters from the unit. The Film's number (1 - 9) is passed to the function. This function must precede the use of *GetFilm*.

Get160Film

Parameters: Pointer to a Film Structure.

Return : 16 Bit Integer, always returns a 1.

GetFilm is used to retrieve a Film's parameters, the film requested by *SendGetFilm*, from the dll. The parameters are passed through the Film Structure.

Example:

```
ReturnVal = SendGetFilm(FilmNum)    tell unit to transfer Film #
                                     FilmNum to dll
do while(ChkCommDone == -1)        wait for comm to finish
ReturnVal = GetFilm(&FilmStruct)    FilmStruct contains Film info
```

SetSys1

Parameters: Pointer to a System1 Structure.

Return : 16 Bit Integer, always returns a 1.

SetSys1 is used to set the System1 Parameters. The parameters are passed to the function through the System1 Structure.

Example:

```
ReturnVal = SetSys1(&Sys1Struct)    set System1 parameters to  
                                   Sys1Struct values  
do while(ChkCommDone == -1)       wait for comm to finish
```

SendGetSys1

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetSys1 is used to get the System1 Parameters from the unit. This function must precede the use of the *GetSys1* function.

GetSys1

Parameters: Pointer to a System1 Structure.

Return : 16 Bit Integer, always returns a 1.

GetSys1 is used to retrieve the System1 Parameters from the dll. The parameters are passed through the System1 Structure.

Example:

```
ReturnVal = SendGetSys1()          tell unit to transfer System1 parameters  
do while(ChkCommDone == -1)      wait for comm sequence to finish  
ReturnVal = GetSys1(&Sys1Struct)  Sys1Struct contains System1 info
```

SetSys2

Parameters: Pointer to a System2 Structure.

Return : 16 Bit Integer, always returns a 1.

SetSys2 is used to set the System2 Parameters. The parameters are passed to the function through the System2 Structure.

Example:

```
ReturnVal = SetSys2(&Sys2Struct)  set System2 parameters to  
                                   Sys2Struct values  
do while(ChkCommDone == -1)       wait for comm to finish
```

SendGetSys2

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetSys2 is used to get the System2 Parameters from the unit. This function must precede the use of the *GetSys2* function.

GetSys2

Parameters: Pointer to a System2 Structure.

Return : 16 Bit Integer, always returns a 1.

GetSys2 is used to retrieve the System2 Parameters from the dll. The parameters are passed through the System2 Structure.

Example:

```
ReturnVal = SendGetSys2()    tell unit to transfer System2 parameters
do while(ChkCommDone == -1) wait for comm sequence to finish
ReturnVal = GetSys2(&Sys2Struct)    Sys2Struct contains System2 info
```

SendGetNumCh

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetNumCh is used to get the number of channels installed from the unit. This function must precede the use of the *GetNumCh* function.

GetNumCh

Parameters: None.

Return : 16 Bit Integer, Number of channels installed.

GetNumCh is used to retrieve the number of channels installed from the dll. The number of channels is returned by the function.

Example:

```
ReturnVal = SendGetNumCh()    tell unit to transfer Number of channels
do while(ChkCommDone == -1) wait for comm sequence to finish
ReturnVal = GetNumCh()        ReturnVal contains Number of channels
```

ZeroStartTime

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

ZeroStartTime is used to zero the beginning time before acquiring data with *GetAllData*.

SendGetAllData

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetAllData is used to get the data from the unit. This function must precede the use of the *GetAllData* function.

GetAllData

Parameters: Pointer to an AllData Structure.

Return : 16 Bit Integer, always returns a 1.

GetAllData is used to retrieve the data from the dll. The parameters are passed through the AllData Structure. If the TimeStamp parameter of the AllData structure returned is equal to -1 then the unit does not have new data available.

Example:

```
ReturnVal = ZeroStartTime()           zero the run time
do
    ReturnVal = SendGetAllData()       tell unit to transfer AllData
    do while(ChkCommDone == -1)       wait for comm to finish
        ReturnVal = GetAllData(&AllDataStruct) AllDataStruct contains run
                                        info
        if (AllData.TimeStamp != -1) then if new data available
            ProcessData()              then graph or save data
    while(Running)
```

SendCrystalLife

Parameters: 16 Bit Integer

Return : 16 Bit Integer, always returns a 1.

SendCrystalLife is used to get the crystal life for a channel from the unit. The parameter is the channel number to retrieve. This function must precede the use of the *CrystalLife* function.

CrystalLife

Parameters: None.

Return : Double.

CrystalLife is used to retrieve the Crystal life remaining for the channel requested by *SendCrystalLife* from the dll. The Crystal life is returned by the function.

Example:

```
ReturnVal = SendCrystalLife(XtalNum)  tell unit to transfer Life for
                                        XtalNum
do while(ChkCommDone == -1)          wait for comm to finish
ReturnVal = CrystalLife()             ReturnVal contains Life for XtalNum
```

ZeroReadings

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

ZeroReadings is used to command the unit to zero the rate and thickness values for all channels and the average rate and thickness.

Example:

```
ReturnVal =ZeroReadings()           tell unit to reset Rate and  
Thickness                             Thickness  
do while(ChkCommDone == -1) wait for comm sequence to finish
```

ZeroTime

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

ZeroTime is used to command the unit to zero the system time.

Example:

```
ReturnVal =ZeroTime()               tell unit to reset time display  
do while(ChkCommDone == -1)        wait for comm to finish
```

ShutterState

Parameters: 16 Bit Integer

Return : 16 Bit Integer, always returns a 1.

ShutterState is used to command the unit to set the shutter open or closed.

Example:

```
ReturnVal =Shutter(0)               tell unit to toggle open the shutter  
do while(ChkCommDone == -1)        wait for comm sequence to finish
```

SendGetShutter

Parameters: None.

Return : 16 Bit Integer.

SendGetShutter is used to retrieve the condition of the shutter, open or closed, from the unit.

GetShutter

Parameters: None.

Return : 16 Bit Integer, Shutter value (0 = Closed, 1 = Open).

GetShutter is used to retrieve the value of the shutter from the dll. The returned value from the function is the value of the shutter :

Example:

```
ReturnVal = SendGetShutter()    tell unit to transfer Shutter value
do while(ChkCommDone == -1)    wait for comm sequence to finish
ReturnVal = GetShutter()        ReturnVal contains Shutter value
```

SendGetReset

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

SendGetReset is used to get the value of the power up reset flag from the unit.

GetReset

Parameters: None.

Return : 16 Bit Integer, Flag value (0 = flag not set, 1 = flag set).

GetReset is used to get the value of the power up reset flag from the dll. The value of the flag is the return value of the function :

Example:

```
ReturnVal = SendCrystalLife(XtalNum)  tell unit to transfer Life left for
                                       XtalNum
do while(ChkCommDone == -1)          wait for comm to finish
ReturnVal = CrystalLife()              ReturnVal contains Life left for
                                       XtalNum
```

LoadDefaults

Parameters: None.

Return : 16 Bit Integer, always returns a 1.

LoadDefaults is used to cause the unit to load the default values into every film and system parameter.

Example:

```
ReturnVal =LoadDefaults()            tell unit to load default values
do while(ChkCommDone == -1)          wait for comm sequence to finish
```

Data Structures:

The size of each data type in the structures is :

double : 8 bytes, the LSB is thrown out before transmission to the unit.

int : 2 bytes.

char : 1 byte.

Film Data

double	Density	film density
double	Tooling	film tooling
double	ZFactor	film zfactor
double	FinThk	film End Thickness
double	ThkSet	film Thickness Setpoint
double	TimeSet	film Time Setpoint
double	SnsAvg	Sensors to average
char	Name[8]	film Name
int	FilmNum	film Number

System1 Data

double	TimeBase	
double	SimMode	simulation mode (1 = on, 0 = off)
double	FreqDisp	frequency display (1 = on, 0 = off)
double	RateRes	rate resolution (1 = hi, 0 = low)
double	RateFilt	rate filter depth (1 - 20)
double	XTool[6]	six individual crystal tooling

System2 Data

double	FMin	minimum frequency
double	FMax	maximum frequency
double	RMin	minimum rate
double	RMax	maximum rate
double	TMin	minimum thickness
double	TMax	maximum thickness
double	EtchMode	Etch mode on/off

AllData

double	TimeStamp	time relative to start time data was acquired
double	AvgRate	average rate
double	AvgThick	average thickness
double	ChRate[6]	up to six individual channels of rate
double	ChThick[6]	up to six individual channels of thickness
double	ChFreq[6]	up to six individual channels of frequency