Intelligent Tuner STIT

Communication Protocol

Information in this document is subject to change without notice.

https://s-team.sk

List of Revisions

The STIT version to which a given document pertains is coded as

HW=*hh* SW=*ss* TS=*tt*

where *hh* represents the STIT HW version, *ss* represents the STIT SW version, and *tt* represents the touchscreen HW/SW version. The STIT version can be read from **More** screen at the LCD touch panel display by touching the **More** button; or as the response to **IDN** command sent via serial communication port (see Section 3.1).

Revision	**Date**	**STIT Version**	**Note**
	03-Aug-15	HW=1.1 SW=1.0 TS=1.1	Original version
A3	12-Aug-22	HW=1.1 SW=1.0 TS=1.1	Formal modifications

Table of Contents

1. INTRODUCTION

This document describes the communication protocol for STIT tuner when controlled via RS232 or RS422 serial communication interface.

STIT[1] **Tuner** is the three-stub motorized and computerized (intelligent) impedance matching system. The tuner enables communication with an external controller and executes tuning stub movements based either on commands received from the serial communication interface or from its own LCD touch panel display.

LCD Touch Panel Display is an add-on module for manual setting of stubs positions and displaying STIT parameters.

1.1 Used Value Types

The used value types are summarized in Tab. 1

Notes:

- In bit representation of a byte, bit 0 is the least significant bit (LSB, weight 1), bit 7 is the most significant bit (MSB, 128).
- Bit *n* of a byte B will be designated B.*n*.
- If not explicitly stated otherwise, numbers without suffix, e.g., `57` or `#57`, will be understood as decimal. The prefix `#` is used mostly in the cases in which the number is the ASCII code of a character.
- Numbers with suffix **h** are hexadecimal, e.g., `57` (decimal) = `39h` (hexadecimal).
- Numbers with suffix **b** are binary, e.g., `57` (decimal) = `0111001b` (binary).

Tab. 1. Used value types.

Symbol	Type	Description	Min value	Max value
`B`	Byte	Unsigned 8-bit	`0`	`255`
`BA8`	8-bit	8-bit array	`00000000b`	`11111111b`
`BA16`	16-bit	16-bit array	`0000000000000000b`	`1111111111111111b`
`I`	Integer	Signed 16-bit (2-byte)	`-32768`	`32767`
`W`	Word	Unsigned 16-bit (2-byte)	`0`	`65535`
`L`	Longint	Signed 32-bit (4-byte)	`-2147483648`	`2147483647`
`S[n]`	String	String containing max. *n* chars	*n* = `1`	*n* = `64`

1.1.1 A Note on String Type

String type for the purpose of this document is a representation of a sequence (array) of 1 to 64 bytes where each byte represents an ASCII character.

Strings will be printed `red` in this document.

Example

The sequence of bytes

```
73, 78, 65, 76, 76
```

is represented by the string

```
INALL
```

[1] The acronym STIT stands for S-Team Intelligent Tuner.

1.2 Command Labels and Codes

Tab. 2 lists the command labels (strings) used in the messages sent from a PC[2] to a STIT and the associated numerical command codes which are returned as part of the response from STIT.

Tab. 2. Command symbols, command labels (sent by PC) and command codes (returned by STIT).

Symbol	Label	Code	Description
CMD_NOCMD	NOCMD	0	Empty command
CMD_INTR	INTR	1	Interrupt and cancel all pending commands
CMD_KM_INIT_ALL	INALL	2	Initialize all motors
CMD_KM_INIT	INIC	3	Initialize selected motors
CMD_KM_GOPOS	GO	4	Set motor/motors positions
CMD_KM_MOTOR1	M1	5	Set motor 1 position
CMD_KM_MOTOR2	M2	6	Set motor 2 position
CMD_KM_MOTOR3	M3	7	Set motor 3 position
CMD_KM_RD_PAR	*PAR?	14	Read all parameters of motors
CMD_GET_IDN	*IDN?	16	Read identification parameters of STIT
CMD_GET_STB	*STB?	18	Read motors status and control register
CMD_RD_TEMP	TEMP?	19	Make one measurement of the internal temperature
CMD_RD_TEMPAVER	TEMP	20	Make and average *N* measurements of the internal temperature
CMD_E_CMD		255	Unknown command code

1.3 Error Codes

Tab. 3 lists error codes returned by STIT as part of its response messages.

Tab. 3. Error codes returned by STIT.

Symbol	Code	Description
CS_OK	0	No error
CS_BUSY	1	STIT is busy (a command is currently being executed)
CS_NOCMD	4	Empty command has been sent to STIT
CS_E_CMD	200	Unknown command
CS_E_CMDPAR	201	Incorrect command parameter
CS_E_INTR	202	Execution of a command has been interrupted
CS_E_MOT_INIT	203	Motors initialization error
CS_E_MOT_ESP	204	Setting motor position error
CS_E_WRITE_PAR	206	Error writing to internal F-RAM[3]

Note for the programmer: The values are unlikely to be changed in future, yet you are advised to use symbols in your code and assign the symbols values in one place of a program.

[2] PC stands for personal computer or any other controller communicating with STIT.

[3] F-RAM stands for Ferroelectric Random Access Memory.

1.4 Miscellaneous

1.4.1 Communication Ports

The STIT tuner is equipped with two serial communication ports: the *main COM port* and the *display COM port*.

- **The main COM port** is represented by the standard DB-9 connector situated at the top of the device. The main port supports RS232 or RS422 communication (optionally CAN Bus).
- **The display COM port** is designed specifically for the detachable LCD touch panel display unit. It employs RS422 serial communication between the STIT main unit and the display unit.

The STIT main unit listens to and accepts commands from *both ports*. After a command execution, STIT always transmits its response to both ports (i.e., also to the port that had not sent the command). This is important for instance for the LCD touch panel module to continuously refresh the display when moving the stubs is commanded from the main port.

1.4.2 STIT Power-Up

The system behavior at power-up depends on whether the LCD touch panel module is used or not.

- The STIT without the touch panel is ready after power-up for receiving commands. The user must have knowledge of certain basic STIT parameters.
- The STIT *with* the touch panel responds to several commands issued from the touch panel module during the power-up procedure. The touch panel module queries the main STIT unit about its identity, its parameters and actual contents of its status registers. A detailed description of these commands can be found later in this document.

1.4.3 STIT Upgrades

Please be aware that the STIT system undergoes continual improvement, which may lead to some future changes in the communication protocol. You are advised to build your program such that modifications can be easily accommodated (e.g., modularize your communications routines, use symbols rather than numbers for constants, define all constants in one program module, etc.).

2. RS232 SPECIFIC ISSUES

2.1 Setting up COM Port

Fixed COM port settings are:

- 8 data bits
- 1 stop bit
- No parity
- Baud Rate 115000 bits/s

2.2 Transmitting

Transmitting means sending messages (sequences of bytes) from PC to STIT. The messages are in principle single or multiple commands requesting STIT to perform certain action and return a response message for each of the commands contained in the message.

2.2.1 Structure of Transmitted Messages

The transmitted messages contain one or more of the following:

- Command Label
- Command Parameters
- Data Separator
- Command Separator
- Message Terminator

Important note: The length of a complete transmitted message string must not exceed 64 bytes.

Command Label

Command Label is a string which identifies the action that should be executed or the nature of the accompanying data (command parameters). Command Label string is *case-insensitive*; it can consist of uppercase letters, lowercase letters, or a combination of uppercase and lowercase letters (uppercase letters will be used in this document). The available command labels are listed in the *Label* column of Tab. 2.

Command Parameters

Command Parameters are strings that can be interpreted as decimal numbers or text. Some commands do not take parameters.

Data Separator

Data Separator is a character that separates the parameters from the command label as well as individual parameters. A *single space character* (#32) must be used as Data Separator, i.e., neither two or more spaces, nor other whitespace characters, such as Tab (#9).

Command Separator

Command Separator serves for separating individual commands in a multi-command message. A *single semicolon character* ; (#59) must be used as Command Separator. It must not be combined, e.g., with spaces, tabs or any other characters.

Message Terminator

Each message must be terminated by the Message Terminator, which is either a single Carriage Return byte <CR> (#13) or a single Line Feed byte <LF> (#10). Their combination <CR><LF> or combination with any other characters is illegal and will cause STIT to return an error message.

2.2.2 Single Commands without Parameters

A command that does not take parameters consists of a Command Label string and the Message Terminator.

Example

To send an empty command, the label NOCMD followed by terminator <CR> or <LF> must be transmitted, e.g.,

```
NOCMD<CR>
```

The five bytes to be transmitted are

```
78, 79, 67, 77, 68, 13
```

2.2.3 Single Commands with Parameters

A command that takes parameters consists of the Command Label string, Data Separator (space #32), command parameters (individual items separated by Data Separator), and a Message Terminator.

Example

To send the command to initialize motors 1 and 3, the label INIC must be transmitted followed by the separator (#32), parameter 5 expressed as ASCII string 5, and the terminator <CR> or <LF>:

```
INIC 5<CR>
```

Hence, the following seven bytes must be transmitted:

```
73, 78, 73, 67, 32, 53, 13
```

2.2.4 Multiple Commands

It is sometimes useful to combine two or more commands as a single message ended by <CR> or <LF>, for instance as an initialization string after the STIT power-up. The string can consist of two or more commands separated by Command Separator ; (semicolon #59).

Example

The multiple-command STIT initialization message may have the form

```
*IDN?;PAR?;*STB?;INALL<CR>
```

Hence, the following byte sequence must be transmitted:

```
42, 73, 68, 78, 63, 59, 80, 65, 82, 63, 59, 42, 83, 84, 66, 63, 59, 73, 78,
65, 76, 76, 13
```

2.3 Receiving

2.3.1 STIT Response to Commands

Generally, STIT responds to a command by:

- Executing the command
- Returning the response message

It is the responsibility of the PC to intercept the returned messages.

Some commands (queries) do not need an execution, they only return data.

2.3.2 Response Message Structure

The messages transmitted by STIT and received by PC are strings, generally structured as follows:

- Command receipt confirmation, which consists of
  - Message Begin Label (Cmd:)
  - Command Code
- Data (may not be present)
- Error Label (Err:)

- Error Code
- Message Terminator <LF> (#10)

Message Begin Label

Message Begin Label is the following fixed string (including the letter case):

Cmd:

Its byte representation is

67, 109, 100, 58

The label is used by PC to detect the beginning of a message.

Command Code

Command Code is a decimal number transmitted as ASCII string in the range 0 to 255 that identifies the particular PC command to which this STIT response belongs. Command Codes are associated with Command Labels according to Tab. 2.

Data

Data returned by STIT can be a string or more substrings separated by Data Separator (space #32). The substrings are interpreted either as decimal numbers or text.

Error Label

Error Label is the following fixed string (including the letter case):

Err:

Its byte representation is

69, 114, 114, 58

The string is used by PC to detect the start of the error code.

Error Code

Error Code is a decimal number (transmitted as ASCII string) that identifies the error while executing the command. Zero value means success (no error). The possible error codes are listed in Tab. 3.

Message Terminator

Each message from STIT is terminated by the Line Feed terminator <LF> (#10).

Example

The response message received by PC as the answer to the command

*IDN?<CR>

may take the form

Cmd:16 S-TEAM STIT S/N=001 HW=11 02-JUL-2013 SW=10 13-SEP-2013 Err:0<LF>

As can be seen from Tab. 2, the Command Code 16 (CMD_GET_IDN) following the Begin Label Cmd: is associated with the Command Label *IDN?.

The data in this case are represented by the text

S-TEAM STIT S/N=001 HW=11 02-JUL-2013 SW=10 13-SEP-2013

The Error Code string 0 following the Error Label Err: implies a successful operation (Error Code = 0 = CS_OK).

3. QUERY COMMANDS

This section describes commands querying identity, parameters, and status of the STIT system.

3.1 Device Identification

Description

This command queries the STIT system for its identity. The system returns a string composed of several items that uniquely identify the STIT. The items are separated by Data Separator (space #32).

Command

Command Label	*IDN?
Parameters	none

Response

Command Code	CMD_GET_IDN = 16	
Returned Data	**Type**	**Description**
Manufacturer	S[8]	Device manufacturer
Model	S[8]	Device model
Serial number	W	Device serial number
Hardware revision	W	Number of hardware revision
Date of hardware revision	S[12]	Date of hardware revision
Software revision	W	Number of software revision
Date of software revision	S[12]	Date of software revision

Example

The command is transmitted as

*IDN?<CR>

The response example:

Cmd:16 S-TEAM STIT S/N=001 HW=11 02-JUL-2013 SW=10 13-SEP-2013 Err:0<LF>

The data string

S-TEAM STIT S/N=001 HW=11 02-JUL-2013 SW=10 13-SEP-2013

can be parsed as follows (after removing local labels S/N=, HW=, and SW=):

Manufacturer	S-TEAM
Model	STIT
Serial number	1
Hardware revision	1.1
Date of hardware revision	02-JUL-2013
Software revision	1.0
Date of software revision	13-SEP-2013

The command was executed successfully (Error Code = 0 = CS_OK).

3.2 Motor Parameters Query

Description

To control the stepper motor movements correctly, the user must have a knowledge of certain of their parameters. This command requests the STIT to return the motor parameters.

Command

Command Label	*PAR?
Parameters	none

Response

Command Code	CMD_KM_RD_PAR = 14		
Returned Data	**Type**	**Range**	**Description**
MotManuf	S[8]		stepper motor Manufacturer
MotType	S[8]		Stepper motor Model
MaxSteps	W		Maximum number of steps
MicroStep	W	1, 2, 4, 8	Microstepping 1, 1/2, 1/4, 1/8, respectively
DistPerStep	W		Distance per step in units of 10 nm (10^{-8} m)
MaxRstSteps	W		Maximum number of steps for stubs reset
InRate	W		Stepper motor pull-in Frequency rate (Hz)
OutRate	W		Stepper motor pull-out Frequency rate (Hz)
StartStop	W		Number of steps for Start/Stop ramp
MinRate	W		Minimum stepper motor frequency rate (Hz)
ZeroSteps1	W		Steps to reach reference position for motor 1
ZeroSteps2	W		Steps to reach reference position for motor 2
ZeroSteps3	W		Steps to reach reference position for motor 3
AdInRstSteps	W		Additional Steps after activating terminal switch
AdOutRstSteps	W		Additional Steps after releasing terminal switch
RstRate	W		Stubs reset stepper motor Frequency rate (Hz)

Out of the complete set of the returned parameters, only those printed bold (*MaxSteps*, *DistPerStep*, *InRate*, *OutRate*, *MaxRstSteps*, *RstRate*) are necessary for the user, enabling to compute the maximum tuning stubs insertion into the waveguide, the number *N* of steps corresponding to a desired stub insertion *h*, and response waiting times (see Derived Quantities below).

Example

The command is transmitted as

```
*Par?<CR>
```

The response example:

```
Cmd:14 NANOTEC L3518 5000 2 500 6010 2400 2400 1 2400 100 90 140 50 50 1200
Err:0<LF>
```

The data string

```
NANOTEC L3518 5000 2 500 6010 2400 2400 1 2400 100 90 140 50 50 1200
```

can be parsed as follows:

MotManuf	NANOTEC
MotType	L3518
MaxSteps	5000 steps
MicroStep	2
DistPerStep	500 × 10-8 m = 5 × 10-6 m = 5 µm
MaxRstSteps	6010 steps
InRate	2400 Hz
OutRate	2400 Hz
StartStop	1 step
MinRate	2400 Hz
ZeroSteps1	100 steps
ZeroSteps2	90 steps

ZeroSteps3	140 steps
AdInRstSteps	50 steps
AdOutRstSteps	50 steps
RstRate	1200 Hz

The command was executed successfully (Error Code = 0 = `CS_OK`).

Derived Quantities

Using the example above, some important derived quantities can be computed.

The maximal stubs insertion depth is

h_{max} = MaxSteps × DistPerStep = 5000 × 5 × 10^{-6} m = 25 × 10^{-3} m = 25 mm.

Let the desired stub extension be h = 10 mm. The number of corresponding motor steps[4] is

$N = h$/DistPerStep = (10 × 10^{-3} m)/(5 × 10^{-6} m) = 2000.

The full stub extension travel time when pulling-in is

t_{max} = MaxSteps/InRate = 5000/2400 = 2.1 s.

The maximal duration of the motors initialization routine with fully extended stubs is approximately

t_{rst} = MaxRstSteps/RstRate = 6010/1200 = 5 s.

[4] This is the value used in the motor movement commands (irrespective of the current stub positions).

3.3 Status Registers Query

Description

Status registers contain various information about a current state of the STIT system and the motors (e.g., whether they are moving or settled in a desired positions, or if an error occurred). This command queries the STIT system for the status registers values.

Command

Command Label	*STB?
Parameters	none

Response

Command Code	CMD_GET_STB = 18	
Returned Data	**Type**	**Description**
CtrlBits	BA16	Reserved for control bits array
Temp	I	Internal temperature (°C)
MotStat	BA16	Motors Status bit array, details see Tab. 4
Mot1ReqPos	I	Motor 1 requested position in steps
Mot2ReqPos	I	Motor 2 requested position in steps
Mot3ReqPos	I	Motor 3 requested position in steps
Mot1ActPos	I	Motor 1 actual position in steps
Mot2ActPos	I	Motor 2 actual position in steps
Mot3ActPos	I	Motor 3 actual position in steps

Tab. 4. Motor status *MotStat* bits. Bit 0 is the least significant bit (LSB).

Bit	**Value**	**Meaning**
0	0	Motor 1 is not in the desired position (e.g., moving or in error)
	1	Motor 1 has reached the desired position and is not moving
1	0	Motor 2 is not in the desired position
	1	Motor 2 has reached the desired position and is not moving
2	0	Motor 3 is not in the desired position
	1	Motor 3 has reached the desired position and is not moving
3	-	(Not used)
4	0	Motor 1 not initialized (e.g., reached terminal switch or hard-stopped)
	1	Motor 1 has been successfully initialized (the bit remains 1 until error)
5	0	Motor 2 not initialized
	1	Motor 2 successfully initialized
6	0	Motor 3 not initialized
	1	Motor 3 successfully initialized
7	-	(Not used)
8	0	Motor 1 no error
	1	Motor 1 error
9	0	Motor 2 no error
	1	Motor 2 error
10	0	Motor 3 no error
	1	Motor 3 error
11-15	-	(Not used)

Bits 8 through 10 indicate motor errors. A motor error occurs when:

- Motors initialization (All Home) procedure was not successful.
- A motor reaches the terminal switch.
- Motors are hard-stopped (e.g., by switching off the DC power supply while motors are moving).

Example

The command is transmitted as

```
*STB?<CR>
```

The response example:

```
Cmd:18 16384 35 119 100 200 300 100 200 300 Err:0<LF>
```

The data string

```
16384 35 119 100 200 300 100 200 300
```

can be parsed as follows:

CtrlBits	=	16384	=	0100 0000 0000 0000b
Temp	=	35 °C		
MotStat	=	119	=	0000 0000 0111 0111b
Mot1ReqPos	=	100 steps		
Mot2ReqPos	=	200 steps		
Mot3ReqPos	=	300 steps		
Mot1ActPos	=	100 steps		
Mot2ActPos	=	200 steps		
Mot3ActPos	=	300 steps		

The command was executed successfully (Error Code = 0 = CS_OK).

The three lowest bits of MotStat are 1, and hence all three motors have reached the desired positions and are not moving. Bits 4, 5, and 6 are also 1, and hence the last **All Home** procedure was successful. Bits 8, 9, and 10 are 0, and hence there has not been a motor error.

Note on Using STB Command

The STIT stores the latest position of stubs into an internal F-RAM after each successfully executed motor move command. If the controller wants to know the current stub positions, for example after STIT power-up, the registers Mot1ActPos, Mot2ActPos, Mot3ActPos must be read by means of the STB command.

Since the responses to commands are sent to *both* COM ports, the STB command issued by PC will also refresh the LCD touch panel display if the panel is connected (as a mere indicator). To ensure correct touch panel update, the STB command should be issued by PC after each motors movement command (see Section 4.3 – Set Motor Positions).

4. STEPPER MOTORS

This section describes commands enabling the user to:

- Set the tuning stubs to desired extensions.
- Initialize motors, i.e.,
  - establish the *reference positions* of the tuning stubs, defining their zero extension into the waveguide, and
  - set the tuning stubs to these reference positions.

Reading actual stub positions and motors status is described in Section 3.3 – Status Registers Query.

4.1 Motors Selection Byte

Motors Selection Byte (MotSelect) selects the motors upon which a motor command will act. The byte has only three relevant bits (see Tab. 5). To select a motor, set the corresponding bit to 1. Some commands do not use MotSelect.

Tab. 5. Relevant bits of Motors Selection Byte *MotSelect*.

Bit	Meaning
0	1 = Select Motor 1
1	1 = Select Motor 2
2	1 = Select Motor 3

For example, to select Motors 1 and 3, set MotSelect to 00000101b = 5.

4.2 Motors Initialization

4.2.1 Initialization Routine and Reference Position

Motors Initialization Routine[5] is an internal procedure which retracts each tuning stub (or each selected tuning stub) until its top terminal switch is activated. Then, in a certain manner assuring high repeatability, the motor is stepped to a position which becomes the *reference position* for that stub.

Reference Position of a motor (tuning stub) is therefore a position corresponding to zero-steps setting in motor movement commands, i.e., zero-millimeter stub insertion depth.

The motors initialization routine resolves all problems arising from motor steps lost in the course of previous operation, i.e., the discrepancy between supposed and actual stub extensions. Such discrepancy may occur, e.g., when DC power to STIT has been switched off while the motors were moving.

[5] When applied to all motors, the routine is also denoted **All Stubs Home** or **All Home** procedure.

4.2.2 All Stubs Home

Description

Motors Initialization Routine is executed with all motors.

Command

Command Label	INALL
Parameters	none

Response

Command Code	CMD_KM_INIT_ALL = 2	
Returned Data	**Type**	**Description**
MotStat	BA16	Motors Status bit array, see Tab. 4

Notes

1. Since the initialization may take relatively long time (depending on motor speeds and the distances the stubs have to travel) you should allow ample time for the response arrival. You may also wish to read motor positions after the initialization.
2. If motor movements take longer than 200 ms then STIT automatically transmits status messages each 200 ms (see Example 2 below). The messages are identical with responses to STB command. You should ignore these messages since their content is meaningless during the initialization[6].

Example 1

This is the case when the initialization takes less than 200 ms (i.e., the stubs are nearly retracted). The command is transmitted as

```
INALL<CR>
```

The response example:

```
Cmd:2 119 Err:0<LF>
```

The command was executed successfully (Error Code = 0 = CS_OK), and hence the returned MotStat value is a valid number. The motors were initialized without error, because MotStat = 119 = 0000 0000 0111 0111b.

Example 2

This is the case when the initialization takes more than 200 ms (e.g., the stubs are much extended). The command is again transmitted as

```
INALL<CR>
```

The responses example:

```
Cmd:18 0 40 0 -1010 -6010 -6010 5000 0 0 Err:1<LF>
Cmd:18 0 40 0 -1010 -6010 -6010 5000 0 0 Err:1<LF>
Cmd:18 0 40 0 -1010 -6010 -6010 5000 0 0 Err:1<LF>
Cmd:18 0 40 0 -1010 -6010 -6010 5000 0 0 Err:1<LF>
Cmd:2 119 Err:0<LF>
```

You should ignore all Cmd:18 messages and wait only for the final Cmd:2 message. For the final message, see comments to Example 1.

[6] The messages are useful when setting stub positions after a successful initialization.

4.2.3 Selected Stubs Home

Description

Motors Initialization Routine is executed with the motors selected by MotSelect Byte.

Command

Command Label	INIC
Parameters	MotSelect as a decimal value

Response

Command Code	CMD_KM_INIT = 3	
Returned Data	**Type**	**Description**
MotStat	BA16	Motors Status bit array, see Tab. 4

Notes

1. The Command Label is indeed INIC (ending with C not T).
2. Since the initialization may take relatively long time (depending on motor speeds and the distances the stubs have to travel) you should allow ample time for the response arrival. You may also wish to read motor positions after the initialization.
3. If motor movements take longer than 200 ms then STIT automatically transmits status messages each 200 ms (see Example 2 below). The messages are identical with responses to STB command. You should ignore these messages since their content is meaningless during the initialization[7].

Example 1

This is the case when the initialization takes less than 200 ms (e.g., the stubs are retracted). Let the initialization routine be performed with Motors 1 and 3. Consequently, MotSelect = 0000 0101b = 5. The command is transmitted as

```
INIC 5<CR>
```

The response example:

```
Cmd:3 87 Err:0<LF>
```

The command was executed successfully (Error Code = 0 = CS_OK), and hence the returned value of MotStat = 87 = 0000 0000 0101 0111b is a valid number. Motors 1 and 3 have been successfully initialized because bits MotStat.4 =1 and MotStat.6 = 1 (see Tab. 4). Bit MotStat.5 = 0 means that Motor 2 has not been initialized since the last powering up of the STIT[8].

Example 2

This is the case when the initialization takes more than 200 ms (e.g., the stubs are much extended). Let the initialization routine be performed with Motors 1 and 3. Consequently, MotSelect = 0000 0101b = 5. The command is transmitted as

```
INIC 5<CR>
```

The responses example:

```
Cmd:18 0 40 2 -4010 0 -6010 2000 0 0 Err:1<LF>
Cmd:18 0 40 2 -4010 0 -6010 2000 0 0 Err:1<LF>
Cmd:18 0 40 2 -4010 0 -6010 2000 0 0 Err:1<LF>
Cmd:3 87 Err:0<LF>
```

You should ignore all Cmd:18 messages and wait only for the final Cmd:3 message. For the final message, see the comments to Example 1.

[7] The messages are useful when setting stub positions after a successful initialization.

[8] This may not be a problem although it is recommended to always initialize all motors after a STIT power-up.

4.3 Set Motor Positions

Description

Selected motors M_i (i = 1, 2, 3) are moved to desired positions h_i. Each position is represented by a 2-byte integer (`I`) equal to the number N_i of motor steps from the reference position. This step count is related with the stub extension into waveguide h_i by

$N_i = h_i$/DistPerStep

The motor step size DistPerStep and other important motor parameters, such as the maximum allowable step count MaxSteps, can be obtained by Motor Parameters Query (PAR) command. The PAR command description includes a few useful example computations.

There are two options when setting motor positions:

- Setting all or selected motors by one single command (Command Label `GO`)
- Setting each motor individually (Command Labels `M1`, `M2`, `M3`)

4.3.1 Set Positions of Selected Motors

Command

Command Label	`GO`	
Parameters	**Type**	**Description**
MotSelect	`B`	Motors Selection Byte
N1	`I`	Number of steps for Motor 1
N2	`I`	Number of steps for Motor 2
N3	`I`	Number of steps for Motor 3

Response

Command Code	`CMD_KM_GOPOS = 4`	
Returned Data	**Type**	**Description**
MotStat	`BA16`	Motors Status bit array, see Tab. 4

Notes

1. Since the motor movements may take relatively long time (depending on motor speeds and the distances the stubs have to travel) you should allow enough time for the response arrival.
2. You are strongly recommended to read motor positions via STB command after each motors move command. This ensures updating of the LCD touch panel display if such is connected.
3. If motor movements take longer than 200 ms then STIT transmits status messages automatically each 200 ms (see Example 2 below). The messages are identical with responses to STB command. You may wish to intercept these messages to refresh your stub extension displays on the run.

Example 1

This is the case when the stubs motion takes less than 200 ms (the new and old stub positions do not differ much). Let us set Motor 1 to position 1500 steps, Motor 2 to 2000 steps, and keep Motor 3 position unchanged. Consequently, MotSelect = `0000 0011b` = `3`. The command is transmitted as

```
GO 3 1500 2000 0<CR>
```

(the last parameter `0` is irrelevant). The response example:

```
Cmd:4 119 Err:0<LF>
```

The command was executed successfully (Error Code = `0` = `CS_OK`), and hence the returned MotStat value is a valid number. The value MotStat = `119` = `0000 0000 0111 0111b` means that the stubs have been successfully set to the desired positions.

Example 2

This is the case when the stubs motion takes more than 200 ms. Let us set Motor 1 to position 1500 steps, Motor 2 to 3000 steps, and keep Motor 3 position unchanged. Consequently, MotSelect = 0000 0011b = 3. After this command execution, let us read the final stub positions using the STB command (always recommended). The two-command message is transmitted as

```
GO 3 1500 3000 0;*STB?<CR>
```

(the last parameter 0 is irrelevant). Supposing that all stubs were initially in zero-mm extensions, the responses may be as follows:

```
Cmd:18 0 33 116 1500 3000 0 169 169 0 Err:1<LF>
Cmd:18 0 33 116 1500 3000 0 908 908 0 Err:1<LF>
Cmd:18 0 33 117 1500 3000 0 1500 1647 0 Err:1<LF>
Cmd:18 0 33 117 1500 3000 0 1500 2386 0 Err:1<LF>
Cmd:4 119 Err:0<LF>
Cmd:18 0 33 119 1500 3000 0 1500 3000 0 Err:0<LF>
```

The first four lines (Cmd:18) are automatic messages sent by STIT each 200 ms. The Error Code = 1 = CS_BUSY means that a command (in this case GO) is being executed, and is not yet completed. The MotStat = 116 = 0000 0000 0111 0100b in the first two lines means that Motors 1 and 2 are moving (MotStat.0 = MotStat.1 = 0) while Motor 3 is in the desired position (MotStat.2 = 1). The Motor 1 and Motor 2 positions are gradually increasing from zero (169, 908 steps).

In lines 3 and 4 Motor 1 has reached the final position (1500 steps), therefore MotStat has changed to 117 = 0000 0000 0111 0101b.

Line 5 (Cmd:4) is the final response to GO command, indicating that the motor movements have been completed successfully.

The last line is the response to *STB? command, showing the final actual motors positions (1500, 3000, 0).

4.3.2 Set Position of Single Motor

Command

Command Label	M1 for Motor 1 M2 for Motor 2 M3 for Motor 3	
Parameters	**Type**	**Description**
N	I	Number of steps for the given motor

Response

Command Code	CMD_KM_MOTOR1 = 5 for Motor 1 CMD_KM_MOTOR2 = 6 for Motor 2 CMD_KM_MOTOR1 = 5 for Motor 3	
Returned Data	**Type**	**Description**
MotStat	BA16	Motors Status bit array, see Tab. 4

Example 1

This is the case when the stub motion takes less than 200 ms. To set Motor 2 to position 1500 steps, the command is transmitted as

```
M2 1500<CR>
```

The response example:

```
Cmd:6 119 Err:0<LF>
```

The command was executed successfully (Error Code = 0 = CS_OK), hence the returned MotStat value is a valid number. The value MotStat = 119 = 0000 0000 0111 0111b means the stub has been successfully set to the desired position.

Example 2

This is the case when the stub motion takes more than 200 ms. Let us set Motor 2 to position 1500 steps, and then read the final stub positions using the STB command. The two-command message is transmitted as

```
M2 1500;*STB?<CR>
```

Supposing that all stubs were initially in zero-mm extensions, the responses may be as follows:

```
Cmd:18 0 117 87 0 1500 0 0 106 0 Err:1<LF>
Cmd:18 0 117 87 0 1500 0 0 843 0 Err:1<LF>
Cmd:6 119 Err:0<LF>
Cmd:18 0 35 119 0 1500 0 0 1500 0 Err:0<LF>
```

The first two lines (`Cmd:18`) are automatic messages sent by STIT each 200 ms. The Error Code = `1` = `CS_BUSY` means that a command (in this case `M2`) is being executed, and is not yet completed. The MotStat = `117` = `0000 0000 0111 0101b` means that Motor 2 is moving (MotStat.1 = `0`) while Motors 1 and 3 are in the desired positions (MotStat.0 = MotStat.2 = `1`). The Motor 2 position gradually increases from zero (`106`, `843` steps).

Line 3 (`Cmd:6`) is the final response to `M2` command, indicating that the motor movement has been completed successfully.

The last line is the response to `*STB?` command, showing the final actual motors positions (`0`, `1500`, `0`).

5. MISCELLANEOUS

5.1 Empty Command

Description

This command serves for the communication test purposes. It requests the STIT to return an empty message with Error Code = CS_NOCMD = 4.

Command

Command Label	NOCMD
Parameters	none

Response

Command Code	CMD_NOCMD = 0	
Returned Data	Type	Description
none		

Returned Error Code = CS_NOCMD = 4.

Example

The command is transmitted as

NOCMD<CR>

The response is

Cmd:0 Err:4<LF>

Arrival of this message means that communication is established between a PC and a STIT.

5.2 Single Measurement of Internal Temperature

Description

This command requests STIT to measure its internal temperature. After receiving the command, STIT executes one measurement of temperature using a sensor situated on the system printed circuit board. STIT returns this temperature as decimal value in degrees of Celsius. The measurement takes about 250 ms.

Command

Command Label	TEMP?
Parameters	none

Response

Command Code	CMD_RD_TEMP = 19	
Returned Data	Type	Description
Internal Temperature	I	Temperature of STIT internal printed circuit board (°C)

Example

The command is transmitted as

TEMP?<CR>

The response example:

Cmd:19 39 Err:0<LF>

The internal temperature is 39 °C. The command was executed successfully (Error Code = 0 = CS_OK).

5.3 Averaged Measurement of Internal Temperature

Description

This command requests STIT to make a specified count of temperature measurements, then compute and return the mean value. The measurement count (Averaging Number) can be specified in the range 1 – 10.

Command

Command Label	TEMP
Parameters	Averaging Number (1 – 10)

Response

Command Code	CMD_RD_TEMPAVER = 20	
Returned Data	**Type**	**Description**
Averaged Temperature	I	Temperature of STIT internal printed circuit board (°C)

Example

The command to return the average of 5 temperature measurements is

TEMP 5<CR>

The response example:

Cmd:20 28 Err:0<LF>

The averaged internal temperature is 28 °C. The command was executed successfully (Error Code = 0 = CS_OK).

5.4 Interrupt Command

Note: This command has not yet been implemented.

Description

This command,

- if possible, interrupts the execution of the currently running command, and
- cancels the execution of all pending commands waiting in STIT command buffer (e.g., in case of a multiple command sent by PC).

STIT returns Error Code = CS_E_INTR = 202.

Command

Command Label	INTR
Parameters	none

Response

Command Code	CMD_ INTR = 1	
Returned Data	**Type**	**Description**
none		

Returned Error Code = CS_E_INTR = 202.

Example

The command is transmitted as

INTR<CR>

The response is

Cmd:1 Err:202<LF>

5.5 Error Message

STIT sends an Error Message after receiving an unrecognizable command.

Error Message Structure

Command Code	CMD_E_CMD = 255
Data	none
Error Code	CS_E_CMD = 200

Example

Let the All Stubs Home command be transmitted with an extra Line Feed <LF> character:

```
INALL<CR><LF>
```

The response may be:

```
Cmd:2 119 Err:0<LF>
Cmd:255 119 Err:200<LF>
```

The first message is the response to command INALL<CR>. Then STIT finds a "message" with no bytes, terminated by <LF>. It therefore returns the second message, which is also an Error Message.