



***Homer* Hot Measurement & Autotuning System**

**High-Power Microwave Impedance Analyzer and
Automatic Impedance Matching Unit**

Communication Protocol

List of Revisions

Version	Date	Remarks
36.1	05-Nov-04	Original version
37.1	20-Dec-04	Modification of Measurement Ranges command (Sec. 11.5)
37.2	07-Jul-04	Addition to typecasting (Sec. 1.2.1)
38.1	16-Jul-07	Corrected error in checksum upper summation index (Sec. 4.2.2) Update to Restart Server, Halt Server (Sec. 9.1)
42.1	25-Jun-08	Sec. 4: Adding transmission of reflected power in Homer Measurement Results. Renaming of bytes R1, SL, SH to RE, SRL, SRH and their updated meaning. Adding meaning to bit 6 of Homer Status Byte HSB (Sec. 4.3).
44.1	06-Oct-08	Homer Status Byte, bit b0 description (Sec. 4.3). SRS value different from 0, 1 (Sec. 11.1)
49.1	14-Nov-09	Values of constants given throughout in text in addition to their symbolic names. Bit 3 of motor status byte MS2 not used (Sec. 4.6). Ping message (Sec. 8.9) modified. ClrFifo (Sec. 8.7) command returns response. FetchLast command (Sec. 8.2) returns also motor positions. New commands Get Timeouts (Sec. 8.8) and Switch Waveform (Sec. 5.8). Decoding the missing reflection coefficient phase (Sec. 4.7.2).
50.1	26-Nov-09	New Switch Waveform command (Sec. 5.8)
53.1	23-Sep-10	All Stubs Home returns response (Sec. 6.2). Minor changes in terminology.
54.1	23-Sep-11	Response to Halt/Restart/Change Server added (Sec. 9.1). <i>Query Continuous Autotuning</i> functionality added (Sec. 7.3). Response to <i>Single Autotuning Step</i> modified (Sec. 7.4). References made to new HTML HomerHelp.chm. <i>Get Timeouts</i> command (Sec. 8.8) reviewed. Homer Resets added (Sec. 9)
55.1	19-Feb-12	New Autotuning Control Parameters Smooth and Delay (Sec. 7.1) MotPeriod query (Sec. 11.6)
57.1	06-Oct-16	Ping command rewritten (Sec. 8.9). Minor rewording.
57.2	13-Mar-17	Change of company official address

The revision version number corresponds to the internal firmware (Server) version number for which it starts to be applicable. For instance, the communication protocol with revision number 50.1 is applicable to Server versions 50, 51, and 52, but not 53, for which the protocol revision 53.1 starts to be applicable.

Information in this document is subject to change without notice.

© S-TEAM Lab. All rights reserved.

S-TEAM Lab, Martina Granca 3451/10, SK-84102 Bratislava, Slovak Republic

<http://www.s-team.sk>

Table of Contents

1. INTRODUCTION	7
1.1 TERMS AND ABBREVIATIONS	7
1.2 INTEGER VALUE TYPES	8
1.2.1 Typecasting	8
1.3 CONSTANTS.....	8
1.4 MISCELLANEOUS	9
2. CAN BUS SPECIFIC ISSUES	11
2.1 CAN IDENTIFIERS	11
2.2 PREPARING THE SYSTEM	11
2.3 RECEIVING MESSAGES	11
2.4 SENDING COMMANDS.....	12
2.4.1 Example: Set Autotuning ON and OFF.....	12
2.5 MULTICAN: MORE HOMERS ON CAN BUS	13
2.6 BROADCAST COMMANDS	13
3. RS232 SPECIFIC ISSUES	15
3.1 SETTING UP COM PORT	15
3.2 TRANSMITTING.....	15
3.2.1 Byte Types	15
3.2.2 Sending Command	15
3.2.3 Sending Data Byte.....	15
3.2.4 Data Objects.....	16
3.3 RECEIVING	16
3.3.1 Receiving and Decoding Bytes	16
3.3.2 Receiving Data Objects.....	16
3.4 ISSUING COMMANDS	16
3.4.1 Command Parameters.....	17
3.4.2 Waiting for Response from Homer.....	17
3.4.3 Command Execution Confirmation	17
4. MEASUREMENT RESULTS	19
4.1 CASE OF CAN BUS.....	20
4.1.1 Homer Measurement Results, Part 1	20
4.1.2 Homer Measurement Results, Part 2	20
4.1.3 Homer Measurement Results, Part 3	20
4.1.4 Motors Data	21
4.2 CASE OF RS232.....	22
4.2.1 MDO Data Bytes	22
4.2.2 Checksum Byte	23
4.3 HOMER STATUS BYTE (HST)	23
4.4 HOMER ERROR BYTE (HER)	24
4.5 MOTOR STATUS, PART 1 (MS1)	24
4.6 MOTOR STATUS, PART 2 (MS2)	24
4.7 DECODING DATA BYTES	25
4.7.1 Some Derived Quantities.....	25
4.7.2 Phase of Reflection Coefficient	26
5. HOMER ANALYZER CONTROL	27
5.1 START CONTINUOUS MEASUREMENT.....	27
5.2 STOP MEASUREMENT	29
5.3 AVERAGING	30
5.4 FREQUENCY COUNTER	31
5.5 SET SUBSTITUTE FREQUENCY	32
5.6 SET CW SAMPLING FREQUENCY	33
5.7 SET FREQUENCY TOLERANCE	34
5.8 SWITCH WAVEFORM	35
6. STEPPING MOTORS	37
6.1 MOTORS SELECTION MAP	37

6.2	MOTORS INITIALIZATION	37
6.2.1	<i>All Stubs Home</i>	38
6.2.2	<i>Selected Stub Home</i>	38
6.3	SET MOTOR POSITIONS	39
6.4	READ MOTOR POSITIONS	40
6.5	HARD STOP OF MOTORS.....	41
7.	AUTOTUNING	42
7.1	AUTOTUNING CONTROL PARAMETERS.....	42
7.1.1	<i>Server Versions Starting V55</i>	42
7.1.2	<i>Older Server Versions, Including V54</i>	43
7.2	HYSTERESIS	44
7.3	CONTINUOUS AUTOTUNING	45
7.4	SINGLE AUTOTUNING STEP	47
7.5	SWITCHING SENDING TUNE POSITIONS ON AND OFF	48
7.6	STUB SWAP ALARM PARAMETERS	49
8.	SINGLE-SHOT COMMANDS	52
8.1	MEAS COMMAND	53
8.2	FETCHLAST COMMAND.....	54
8.3	COMPSTUBS COMMAND.....	55
8.4	MEACOMP COMMAND	56
8.5	MEATUN COMMAND.....	57
8.6	MEATUNMEA COMMAND	58
8.7	CLRFIFO COMMAND.....	59
8.8	GET TIMEOUTS.....	59
8.8.1	<i>Using MeasTimeout and MotorsTimeout</i>	60
8.8.2	<i>Stopping Homer</i>	60
8.8.3	<i>Get Timeouts Command Structure</i>	60
8.9	PING MESSAGE.....	61
9.	HOMER RESET COMMANDS	63
9.1	RESTART SERVER, HALT SERVER, CHANGE SERVER.....	63
9.2	FACTORY RESET, USER-DEFINED RESET.....	65
9.3	CREATE USER-DEFINED RESET	68
10.	CW MEASUREMENT SETUP	70
10.1	MEASUREMENT TIMING	70
10.1.1	<i>Signal Measurement Period</i>	70
10.1.2	<i>Offset Measurement Period</i>	70
10.1.3	<i>Frequency Measurement Period</i>	70
10.1.4	<i>Temperature Measurement Period</i>	71
10.2	A/D CONVERTER RANGE	71
10.2.1	<i>Signal Measurement Range</i>	71
10.2.2	<i>Offset Measurement Range</i>	71
10.2.3	<i>Offset Ranges Equal to Signal Ranges</i>	71
10.3	DATA TRANSMITTING	72
10.3.1	<i>Send Event Register and Send Mask Register</i>	72
10.3.2	<i>Sending Period</i>	72
10.3.3	<i>Motors Refresh Period</i>	72
11.	MEASUREMENT SETUP COMMANDS.....	74
11.1	SET/GET RUNNING AND SENDING STATES	74
11.2	SET SIGNAL AND OFFSET MEASUREMENT PERIODS	75
11.3	SET FREQUENCY AND TEMPERATURE MEASUREMENT PERIODS	76
11.4	SET SENDING PERIOD AND SEND MASK REGISTER.....	77
11.5	MEASUREMENT RANGES	78
11.6	SET MOTORS REFRESH PERIOD	79
12.	USING ADVANCED COMMMANDS	80
12.1	NORMAL SITUATION	80
12.2	INCREASING MEASUREMENT RATE	80
12.3	STUB SWAP ALARM	80

12.4	SINGLE-SHOT COMMANDS	81
------	----------------------------	----

1. INTRODUCTION

Note: Except *Switch Waveform* command (Sec. 5.8), this document covers only CW mode of operation. Rectified and Pulsed measurement modes are not described.

1.1 Terms and Abbreviations

Homer¹ Analyzer: A system or (in case of Autotuner) its part which performs measurements and provides for communication.

Homer Mototuner: A system or (in case of Autotuner) its part which carries out tuning stub movements. Also denoted simply Tuner.

Homer Autotuner: A system combining Homer Analyzer and Mototuner to enable automatic impedance matching (autotuning). Entire Autotuner intelligence and communication capability resides within the Homer Analyzer part.

In broader sense, the term **Homer** in this document will apply to both Homer Analyzer and Homer Autotuner.

MultiCAN: A case when more Homers than one are connected to CAN bus. Details see in Help for Homer (HomerHelp.chm), topic [Advanced Topics > MultiCAN: More Homers on CAN Bus](#).

Measure-Tune-Send (MTS) sequence: A mnemonic description of type and succession of possible actions taken by Homer in one measurement cycle (measurement, autotuning, sending data). Detailed structure and order of constituent actions depend on the system state and a particular command that invokes MTS sequence.

Reference position: A motor position (tuning stub insertion depth) after executing the motor *initialization routine*. The reference position is the position corresponding to zero steps in motor movement commands (i.e. 0-mm stub insertion depth). The reference position is derived from the terminal switches.

Initialization routine: The initialization routine is a procedure which withdraws 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. The reference position is a position corresponding to zero setting in motor movement commands (i.e. zero-millimeter stub insertion depth).

When applied to all motors, the routine is also denoted **All Stubs Home**, **Reset Stubs**, **All Motors Home** or just **All Home** procedure.

Abbreviations:

AP	Actual Positions (of motors)
CAN	Controller Area Network
HER	Homer Error Byte
HMR	Homer Measurement Results
HST	Homer Status Byte
ID	CAN object identifier
LSB	Least significant bit/byte
MD	Motors Data
MDO	Measurement Data Object
TP	Tune Positions (of motors)
MS1	Motor Status, Part 1
MS2	Motor Status, Part 2
MSB	Most significant bit/byte
MSM	Motors Selection Map
MTS	Measure-Tune-Send
PC	Personal computer (or another controller)
SER	Send Event Register
SMR	Send Mask Register
SSA	Stub Swap Alarm

¹ The term HOMER comes from HOt MEasuRement system.

Notes:

- In bit representation of a byte, bit 0 is LSB (weight 1), bit 7 is MSB (weight 128).
- Bit n of a byte B will be designated $B.n$.
- If not explicitly stated otherwise, numbers without suffix, e.g. 57, will be understood as decimal.
- Numbers with suffix **h** are hexadecimal, e.g. 57 (decimal) = 39h (hexadecimal).

1.2 Integer Value Types

Integer value types used here are:

Type	Description	Min value	Max value
Byte	Unsigned 8-bit	0	255
Word	Unsigned 16-bit (2-byte)	0	65535
Integer	Signed 16-bit (2-byte)	-32768	32767
Longint	Signed 32-bit (4-byte)	-2147483648	2147483647

1.2.1 Typecasting

Programming languages offer typecasting from unsigned to signed variable types and vice versa. To indicate typecasting of a word W to integer I , we shall use Pascal-like notation, i.e. $I = \text{smallint}(W)$.

In principle, if MSB of W is zero, then $I = W$. If MSB of W is one, then

- take the remaining bits: $W_1 = W \text{ and } 32767$
- invert them: $W_2 = \text{not}(W_1) \text{ and } 32767$
- add unity to thus obtained number: $W_3 = 1 + W_2$
- add minus sign. $I = -W_3$

Mathematically

$$I = W \quad \text{if } W \text{ and } 32768 = 0$$

$$I = -\{1 + [\text{not}(W \text{ and } 32767) \text{ and } 32767]\} \quad \text{if } W \text{ and } 32768 = 1$$

Note that in certain environments, like Microsoft Excel, the above procedure can be written simply as

$$I = W \quad \text{if } W < 32768$$

$$I = W - 65536 \quad \text{if } W > 32767$$

Nevertheless, one should verify it for a range of W values. Some typecasting examples:

W	I
0	0
1	1
32766	32766
32767	32767
32768	-32768
32769	-32767
65534	-2
65535	-1

Similar considerations apply to signed vs. unsigned type of any size.

1.3 Constants

This is the list of used constants, along with brief description of their meaning.

Symbol	Dec	Hex	Description
msAtOFF	0	00	Switch continuous autotuning OFF
msAtON	1	01	Switch continuous autotuning ON
msAtSingle	2	02	Perform one autotuning step with the latest measured data
msCanSetAtPar	3	03	Only CAN: Set parameters governing autotuning behavior
msConfirm	4	04	End message of command execution confirmation
msGetAtune	5	05	Query continuous autotuning status
msSetFdelta	6	06	Set frequency tolerance
msSetFsubst	7	07	Set substitute frequency
msReset	8	08	Homer resets
LF	10	0A	Line Feed
CR	13	0C	Carriage Return
msMeasObject	16	10	End message of Measurement Data Object (MDO)
msSweepStart	17	11	Start measurement
msSweepStop	18	12	Stop measurement
msMotStop	19	13	Hard stop of motors
msPingPong	20	14	Only CAN: Password and response to it
msDataBegin	28	1C	Message indicating that the following will be data bytes
msSrvHalt	34	22	Terminate Homer measurement program (Server), start file transfer program
msFetchLast	39	27	Instruct Homer to send the latest Measurement Results
msHmWform	53	35	Switch signal waveform mode of operation (CW, Rectified, Pulsed)
msHmCounter	56	38	Set frequency counter
msHmAvrg	57	39	Set Homer averaging numbers
msGetTmouts	61	3D	Get Homer measurement timeout and motors movement timeout
msMotInit	69	45	Initialize all motors
msOneHome	70	46	Initialize one motor
msMotSet	71	47	Set motor positions
msATunCmd	72	48	Only RS232: Autotune commands
msATunPar	73	49	<ul style="list-style-type: none"> • RS232: Set or read parameters governing autotuning behavior • CAN: Read autotuning parameters
msMotRead	74	4A	Read motor positions
msSetFsmp1Cw	75	4B	Set sampling frequency
msMotRefresh	76	4C	Motors Refresh period
msSrvRst	80	50	Terminate and restart Homer measurement program (Server)
msClrFifo	84	54	Clear Homer FIFO
msMeas	85	55	Make one measurement
msCompStubs	86	56	Compute stub positions to achieve impedance match
msMeaComp	87	57	Measure and compute stub positions to achieve impedance match
msMeaTun	88	58	Measure and make one autotuning step
msMeaTunMea	89	59	Measure, make one autotuning step, measure again
msYesSendTune	90	5A	Send Tune Positions during continuous measurement
msNoSendTune	91	5B	Do not send Tune Positions during continuous measurement
msSetStbSwap	92	5C	Send Stub Swap Alarm parameters to Homer
msHmSetOther	94	5E	Send various Homer parameters (e.g. timing, ADC range)
msTuSetOther	96	60	Send various Tuner parameters (e.g. Hysteresis)
CmndLbl	128	80	Command Label; precedes command code byte

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.

1.4 Miscellaneous

If not stated otherwise, all unspecified bits and bytes in the following description are irrelevant.

System behavior at power-up is controlled by internal **Hom.cfg**, **Tun.cfg**, **Rs232.cfg** and **Can.cfg** files. For details and ways of modifying them, see Help for Homer (HomerHelp.chm), topics

Advanced Topics > Homer Analyzer Starting Parameters,

Advanced Topics > Homer Autotuner Starting Parameters, and

Homer Start > Communication Issues

By factory default, the system starts continuous measurement without autotuning and periodically sends measurement results and motor positions. You can intercept the messages and use the data they carry. In case of CAN bus, the data are sent in the form of CAN messages. In case of RS232, the data are sent in the form of byte streams called Measurement Data Objects (MDO). Example Delphi/Pascal code for receiving MDO via RS232 is attached as **HmRs232.pas** file.

To start continuous autotuning manually, set the **AutoTune** switch to ON position. To start continuous autotuning programmatically, leave the **AutoTune** switch in OFF position and send the message described in Section 7.3. To control autotuning parameters, see the commands in Section 7.1.

Please understand that the system is being continually improved, which may lead to some future changes in 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. CAN BUS SPECIFIC ISSUES

Protocol applied: CAN Specification 2.0, Part B.

2.1 CAN Identifiers

This is the list of the CAN object identifiers used, along with brief description of their purpose. In case of MultiCAN, this is the set of *Base Identifiers*, from which the set of ID for each individual Homer is derived (see Section 2.5).

Symbol	Dec	Hex	Sender	Purpose
ciBrCast9	9	09	PC	ID used for Broadcast Commands (see Section 2.6)
ciHmStop	10	0A	PC	Stop Homer internal measurement and tuning
			Homer	Response to Stop command
ciHmRes1	11	0B	Homer	Measurement results, Part 1 of 3
ciHmRes2	12	0C	Homer	Measurement results, Part 2 of 3
ciHmRes3	13	0D	Homer	New: Measurement results, Part 3 of 3
ciMotC	14	0E	PC	Motors commands
ciMotS	15	0F	Homer	Motors status (positions and errors)
ciHomC	16	10	PC	Homer commands (i.e. those concerning measurement)
ciAtunC	17	11	PC	Commands controlling the autotuning process
ciHomR	18	12	Homer	Response to Homer commands (ciHomC)
ciAtunR	19	13	Homer	Response to autotuning commands (ciAtunC)
	20	14		Reserved
	21	15		Reserved
ciMotSQ	22	16	Homer	Motors status (positions and errors); equivalent with ciMotS
ciTransfC	64	40	PC/Homer	Used in transfer of data blocks
ciTransfR	65	41	PC/Homer	Used in transfer of data blocks
ciObjTX	66	42	PC/Homer	Used in transfer of data blocks
ciObjRX	67	43	PC/Homer	Used in transfer of data blocks

Notes 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.
- Do process messages with ID = **ciMotS** and ID = **ciMotSQ** equally.

2.2 Preparing the System

Using a CAN Bus cable, connect the CAN interface of your controller (PC) with Homer.

Start Homer. Depending on the AUTORUN line in its internal Hom.cfg configuration file, Homer either remains idle, waiting for commands, or starts measurements, possibly autotuning, and sending messages into the CAN bus. An example of the AUTORUN line is shown below:

```
AUTORUN=3 ;AutoStart after power-up: Run (b0=1), Send (b1=1)
```

See Help for Homer or Homer Handbook for details about Hom.cfg file.

Notes:

- Be sure that Homer is equipped with CAN Bus option.
- You do not need microwave power to make experiments with communication.

2.3 Receiving Messages

In your PC, start a CAN Bus monitoring program. Set baudrate equal to that of Homer. These are:

- for SrvHo.exe firmware version V9 and less (before 12-Dec-02) 100 kbit/s fixed,
- for SrvHo.exe firmware version V10 and more (after 12-Dec-02) 500 kbit/s default, selectable by editing **can.cfg** (see Help for Homer).

Receive messages and observe them. The train of messages typically may look as follows (decimal):

```

11:  12   0  23 112  10 250   0 255
12: 142   5 208 253  64 103 154  14
13:  24 252 131   6   0   0   0   0
15: 224 248   0   0 184  11 119   0
11:  12   0  23 112  10 250   0 255
12: 143   5 205 253  64 103 154  14
13:  25 252 135   6   0   0   0   0
15: 224 248   0   0 184  11 119   0

```

The first number (red) is CAN identifier (ID), the rest are data bytes (B0 to B7).

- ID = 11 means that this message is Part 1 of Homer Measurement Results; it contains 8 data bytes. See Section 4.1 for how to interpret the data bytes.
- ID = 12 identifies Part 2 of Homer Measurement Results.
- ID = 13 identifies Part 3 of Homer Measurement Results.
- ID = 15 means that this is a message carrying motor positions and status.

2.4 Sending Commands

To issue a command to Homer:

- Set CAN message identifier ID.
- Set the data byte count.
- Prepare data bytes (Byte0 to Byte7). Byte0 is command code (identifies the command); the remaining bytes, if any, are command parameters.
- Send the message.
- Receive messages and wait for a response from Homer (i.e. a message on the bus with a specific identifier).
- Check Byte0 of the returned response. If the command was successful, Byte0 is equal to the command code sent. In case of command error, the returned Byte0 is the command code sent incremented by 128 (i.e. its MSB is set to 1). In addition or as alternative to that, some commands return error code in one of the response data bytes.

2.4.1 Example: Set Autotuning ON and OFF

(Be sure the **AutoTune** hardware switch on your Homer is in OFF position.)

Execution of this command is easily verifiable even without microwaves provided tuning is not suspended for low microwave powers (see Section 7.1, byte denoted “Wait when RF power is low”). It is because the stubs will move randomly when Autotuning is ON. The effect must be similar to setting the manual **AutoTune** switch to ON position.

To Set Autotuning ON:

- ID = 17
- Data bytes count = 1
- B0 = 1
- The message to be sent (decimal):

```
17:  1
```
- Send the above message.
- Receive messages and wait for a message with ID = 19. The message returns one data byte, which should be equal to 1.

```
19:  1
```

If the returned byte is 0, the command was not successful (autotuning is still OFF).

To Set Autotuning OFF, the command is the same as before except that B0 = 0 in both the command and the response. Hence

```

17:  0 (command)
19:  0 (response)

```

2.5 MultiCAN: More Homers on CAN Bus

Theoretically, up to 20 Homers can be connected at the same time on CAN bus and controlled independently; details see in Help for Homer (HomerHelp.chm), topic [Advanced Topics > MultiCAN: More Homers on CAN Bus](#). The following measures must be taken:

- All Homers must be set to the same CAN baudrate.
- Each Homer must be assigned a unique number, ranging from 1 to 20, called *CAN Address*. The assignment is made by the following statement in **Hom.cfg** configuration file (the example below assigns a Homer the address 4):

```
CAN_ADDR=4
```

Caution! Be careful to avoid a case of more Homers having *the same* CAN Address. This will lead to confusion because such Homers cannot be distinguished. Otherwise, the addresses may be arbitrary. Example: If $N=4$ Homers are on the bus, the addresses may be for instance **7, 11, 2, 4** or **1, 2, 5, 6** or **1, 2, 3, 4**, etc.

Each Homer works with its own set of CAN identifiers, which are derived from the set of *Base Identifiers*, listed in Section 2.1, as follows: If ID_0 is a Base Identifier and N is the Homer CAN Address, then the corresponding identifier ID_N is

$$ID_N = ID_0 + 100(N - 1)$$

This means that Homer #1 uses the set of Base Identifiers; for Homer #2 the IDs are shifted by 100 (i.e. within range 100-199), etc.

If a message with identifier ID has been received, the following formulas serve for identifying the sender Homer and base identifier ID_0 :

$$N = ID \text{ div } 100$$

$$ID_0 = 1 + (ID \text{ mod } 100)$$

where *div* is integer division (e.g. $216 \text{ div } 100 = 2$) and *mod* returns the remainder (e.g. $216 \text{ mod } 100 = 16$).

Example

You wish to start measurement with Homer #3, i.e. the one with CAN Address $N = 3$. The command is described in Section 5.1. The base ID for this command is $ID_0 = \text{ciHomC} = 16 = 10\text{h}$. You therefore send a message with $ID = 16 + 100 \cdot (3 - 1) = 216$. The Homer responds with a message with $ID = 218$, hence the base ID is $ID_0 = \text{ciHomR} = 18 = 12\text{h}$.

2.6 Broadcast Commands

Sometimes you may wish to make equal setting of all Homers on the CAN bus without necessity to set them individually one after another. Or you may wish to trigger an action synchronously in all Homers. Broadcast commands serve this purpose. Unlike selective commands, accepted only by addressed Homer, the broadcast commands are accepted by all Homers. They are built as follows.

Suppose a message should be broadcast with original identifier OrigID and data byte count OrigCnt.

Broadcast commands are restricted to $\text{OrigCnt} \leq 7$. Let the data bytes be OrigB0 to (maximally) OrigB6. The broadcast message is then set up as follows.

- The CAN ID of the broadcast message is always **ciBrCast9** = 9.
- Data bytes B0...B6 of the broadcast message are equal to OrigB0...OrigB6, i.e. $B_0 = \text{OrigB}_0$, $B_1 = \text{OrigB}_1$, ... $B_6 = \text{OrigB}_6$. Unused bytes may have arbitrary value.
- Last data byte (B7) of the broadcast message is the ID of the original message: $B_7 = \text{OrigID}$.
- Data byte count of the broadcast message is always 8.

Now, you can send the message and try to intercept and process responses from all the Homers trying to talk at the same time...

Since the broadcast commands are restricted to $\text{OrigCnt} \leq 7$, you cannot use them e.g. for the commands Set Motor Positions (Section 6.3) or Set Autotuning Control Parameters (Section 7.1), requiring 8 data bytes. Such commands must be sent to each of the Homers individually.

Example:

Set Autotuning ON in all Homers available on the bus. The original message has identifier OrigID=17 and data byte count OrigCnt=1, the only data byte being OrigB0=1 (cf. Example in Section 2.4.1). The original message in its selective form, issued only to Homer #1 would be

17: 1 (command)

19: 1 (response)

The same command to and response from Homer #3, i.e. the one with CAN Address $N = 3$ would involve identifiers $17+100*(N-1) = 17+100*(3-1) = 217$ (command) and $19+100*(3-1) = 219$ (response):

217: 1 (command)

219: 1 (response)

To broadcast the command, you must sent an 8-byte message with ID = **ciBrCast9** = 9 as follows:

9: 1 x x x x x x 17

where **x** may assume arbitrary value. The responses from Homers #1 and #3 would be as before:

19: 1

219: 1

3. RS232 SPECIFIC ISSUES

3.1 Setting up COM Port

Fixed COM port settings are:

- 8 data bits
- 1 stop bit
- No parity

Variable COM port settings are defined by **Rs232.cfg** file stored inside Homer. A copy of this file is in a subfolder of your S-TEAM installation folder: something like

`c:\Program Files\S-Team\MyDevices\STHT14_139\`

There you can learn BaudRate of your Homer's RS232. An alternative way is to open **Rs232.stp** file, which is automatically created in your S-TEAM installation folder: something like

`c:\Program Files\S-Team\HomSoft\Ver4040\`

after you terminate the *HomSoft* Windows visualization and control program.

3.2 Transmitting

3.2.1 Byte Types

The transmitted bytes fall in the following categories:

- **Command codes** (message codes). Command code is a byte, which identifies the action that should be executed or identifies the nature of the data that have just been sent. It may have any value from 0 to 127.
- **Data bytes**. A data byte is interpreted as a numerical value (a number or a part of a number) or as a character.
- There is one special byte called *Command Label Byte*; its value is **CmndLb1** = 128 = 80h.

3.2.2 Sending Command

To send a command, the associated command code byte (let us denote it **CommandCode**) must be preceded by Command Label Byte. Hence, two bytes are transmitted:

`CmndLb1, CommandCode`

i.e.

`128, CommandCode`

Example: To send Data Begin command (command code **msDataBegin** = 28 = 1Ch), the two bytes **CmndLb1, msDataBegin** are transmitted, i.e. `128, 28`.

3.2.3 Sending Data Byte

Two cases must be distinguished: the data byte (let us denote it **DataByte**) differing from **CmndLb1** = 128 = 80h and data byte equal to **CmndLb1**.

- If **DataByte** differs from **CmndLb1**, it is simply transmitted as a single byte.
- If **DataByte** = **CmndLb1**, it is doubled, i.e. sent as two consecutive identical bytes:

`DataByte, DataByte`

i.e.

`128, 128`

Example: To send the 5-byte data stream **1, 2, 128, 3, 4**, the following six bytes must be transmitted:

`1, 2, 128, 128, 3, 4`

3.2.4 Data Objects

Transfer of data (however they may be interpreted) between Homer and an external controller is realized by means of *Data Objects*, which are variable-length byte sequences consisting of three consecutive components:

1. **Data Begin** message, which is a two-byte sequence: **CmndLb1** = 128 = 80h and **msDataBegin** = 28 = 1Ch, i.e. **128, 28**.
2. **Data** itself (note that each byte equal to **CmndLb1** must be doubled).
3. **End Message**, which is a two-byte sequence **CmndLb1, EndMsg**, where **EndMsg** is data identifier, indicating that Data stream has ended and how it should be interpreted.

Example: To send the data stream 30, 128, 40 with end message code 99, the following data sequence must be transmitted:

128, 28, 30, 128, 128, 40, 128, 99

Data bytes are indicated by boldface; note the doubling of byte 128.

3.3 Receiving

3.3.1 Receiving and Decoding Bytes

- When a byte (B1) has been received and this byte differs from **CmndLb1** = 128 = 80h, this byte is interpreted as data byte equal to B1.
- When the received byte B1 = **CmndLb1** = 128 = 80h, another byte (B2) must be received as well.
 - If the second byte B2 differs from **CmndLb1** = 128 = 80h, this 2-byte eventuality is interpreted as the reception of a command (message) with command code equal to B2.
 - If this second byte B2 = **CmndLb1**, this 2-byte occurrence is interpreted as the reception of a single data byte equal to B1 = B2 = **CmndLb1**.

As an example implementation, see **ReceiveAndDecodeByte** procedure in HmRs232.pas file. The procedure returns:

- **b**: data byte or command code (depending on **IsMessage** flag).
- **IsMessage**: boolean flag. If TRUE, **b** is command code; if FALSE, **b** is data byte.
- (implicitly) **Err00**: communication error (zero in case of no error occurring while receiving).

3.3.2 Receiving Data Objects

To intercept a data object, you may proceed as follows:

- Clear the receive buffer.
- Receive and decode bytes repeatedly until a Data Begin message arrives (command code **msDataBegin** = 28 = 1Ch).
- Following this, receive, decode, and store all incoming data bytes until the next command comes. The command code is interpreted as the end message (**EndMsg**) associated with the just received data sequence.

See **ReceiveDataObject** function in HmRs232.pas file as an example implementation. The function input is maximum wait time **wait_ms** for arriving a Data Begin message (and, after that, for completing the data object receiving). The function result is TRUE in case of success. The function also returns:

- **Buffer**: buffer for received data bytes; it is implicitly typecast to a desired structure **RxBuff**.
- **n**: received byte count (valid data are **Buffer[0]** to **Buffer[n-1]**).
- **EndMsg**: end message code.

3.4 Issuing Commands

Issuing a command (let its code be denoted **CommandCode**) consists generally of four steps:

1. Send the **msDataBegin** = 28 = 1Ch command.

2. Send the command parameters.
3. Send the **CommandCode** command.
4. Wait for a response from Homer.

Points 1 and 2 are skipped if there are no parameters associated with the command (e.g. All Stubs Home command).

Point 4 is omitted if there is no response from Homer (or if we choose to ignore it).

3.4.1 Command Parameters

Command parameters, if present, are transmitted in the form of a text *command string* composed of three substrings:

1. **Label** substring, identifying the command type. (It is in fact superfluous since it doubles the command code; it has remained for historical reasons and is helpful when debugging programs.) The Label *must* be separated from the rest by at least one Space character (ASCII code 32) or Tab character (ASCII code 9).
2. **Parameters** substring, comprising the text representation of the parameters associated with the command. The parameters substring form depends on the particular command (see individual commands description). The particular parameters appearing in the parameters substring must be separated by at least one Space or Tab character.
3. **Terminator** substring is a two-byte sequence Carriage Return (**CR** = 13) + Line Feed (**LF** = 10); it needs not be separated from the rest.

Text representation of a number means its conversion from binary to string. As an example, the text representation of the number $-1.37 \cdot 10^{-2}$ is the string "**-1.37E-2**", or the string "**-0.0137**", etc.

Example: To switch continuous autotuning OFF and ON (command code **msATunCmd**=72=48h), the Label is "**ATC**" and the parameter to be sent is 0 and 1 for OFF and ON, respectively. The corresponding command strings (without terminator) are "**ATC 0**" and "**ATC 1**", which corresponds to byte sequences **65, 84, 67, 32, 48** and **65, 84, 67, 32, 49**, respectively. The complete commands are transmitted as the following byte sequences:

```
128, 28, 65, 84, 67, 32, 48, 13, 10, 128, 72    to switch autotuning OFF
128, 28, 65, 84, 67, 32, 49, 13, 10, 128, 72    to switch autotuning ON
```

Note the arbitrary use of more spaces (32) in the second command.

3.4.2 Waiting for Response from Homer

Generally, Homer responds to a command by

- executing the command,
- returning a data object.

Some commands (queries) need no execution; some do not return anything.

The returned data may comprise Homer parameters or settings requested by a particular command. For most of the commands, however, the returned object only confirms command execution.

3.4.3 Command Execution Confirmation

To confirm the receipt and execution of a command (suppose its code being **CommandCode**), Homer returns a two-byte data object with end message code **msConfirm** = 4.

- First data byte (B1) is the replica of the command code: B1 = **CommandCode**.
- Second data byte (B2) is the error byte (B2 = 0 if the command has been executed successfully). An exception is the response to **Ping** command where B2 is equal to *Test Byte* sent by the **Ping** command (see Section 8.9).

Example: Response to the above **ATC 0** and **ATC 1** successfully executed commands (command code **msATunCmd**=72=48h) is

```
128, 28, 72, 0, 128, 4
```

If Homer could not execute the commands successfully, the return may be for instance

```
128, 28, 72, 3, 128, 4
```

where **3** is error code.

Note: In the RS232 command examples given throughout this document, command strings are shown without the **CR+LF** terminator.

4. MEASUREMENT RESULTS

All data relating to Homer measurement (*Homer Measurement Results*, HMR) and motor positions (*Motors Data*, MD) are sent from Homer either periodically or on request.

Homer Measurement Results include

- tuner input reflection coefficient
- incident power
- frequency
- internal temperature
- load reflection coefficient, also called “deembedded” reflection coefficient
- Homer Error Byte
- reflected power: only in case of Rectified and Pulsed sampling when averaged results are transmitted (i.e. not individual samples).

Deembedded or load reflection coefficient is the reflection coefficient that would be measured if the tuning stubs were withdrawn to zero positions and the measurement plane shifted to the load plane. See Help for Homer (HomerHelp.chm) for details concerning the reference planes.

Motors Data include

- motor positions
- motors status

Motors status contains various information about current state of the motors (e.g. whether moving or in desired positions or if an error occurred).

Motor positions may be either

- Actual Positions (AP) or
- Tune Positions (TP).

Tune Positions are values computed to achieve an impedance match. Input data for the computation are Homer Measurement Results and Actual Positions. Bit 7 of Motor Status, Part 2 (Section 4.6) determines if the sent motor positions are AP or TP. Section 7.5 describes the command controlling which of the two motor position types is to be transmitted.

4.1 Case of CAN Bus

In case of CAN bus, **Homer Measurement Results** consist of three consecutive CAN messages, the first with ID = **ciHmRes1** = 11 = 0Bh, the second with ID = **ciHmRes2** = 12 = 0Ch, and the third with ID = **ciHmRes3** = 13 = 0Dh.

4.1.1 Homer Measurement Results, Part 1

ID	Sender	Byte Count
ciHmRes1 =11=0Bh	Homer	8

Byte	Name	Meaning	Note
0	HST	Homer Status Byte (Section 4.3)	
1	HER	Homer Error Byte (Section 4.4)	
2	PH	Coded incident power – MSB	
3	PL	Coded incident power – LSB	
4	PE	Coded incident power – exponent	
5	TL	Internal temperature in units of 0.1 Celsius – LSB	
6	TH	Internal temperature in units of 0.1 Celsius – MSB	
7	RE	Coded reflected power – exponent	S

4.1.2 Homer Measurement Results, Part 2

ID	Sender	Byte Count
ciHmRes2 =12=0Ch	Homer	8

Byte	Name	Meaning	Note
0	XL	Coded real part of measured reflection coefficient – LSB	
1	XH	Coded real part of measured reflection coefficient – MSB	
2	YL	Coded imaginary part measured of reflection coefficient – LSB	
3	YH	Coded imaginary part measured of reflection coefficient – MSB	
4	F0	Frequency in units of 10 Hz – Byte 0 (LSB)	F
5	F1	Frequency in units of 10 Hz – Byte 1	F
6	F2	Frequency in units of 10 Hz – Byte 2	F
7	F3	Frequency in units of 10 Hz – Byte 3 (MSB)	F

4.1.3 Homer Measurement Results, Part 3

ID	Sender	Byte Count
ciHmRes3 =13=0Dh	Homer	8

Byte	Name	Meaning	Note
0	DXL	Coded real part of load reflection coefficient – LSB	
1	DXH	Coded real part of load reflection coefficient – MSB	
2	DYL	Coded imaginary part of load reflection coefficient – LSB	
3	DYH	Coded imaginary part of load reflection coefficient – MSB	
4	SRL	Sample serial number/Coded reflected power – LSB	S
5	SRH	Sample serial number/Coded reflected power – MSB	S
6	R1	Reserved	
7	R2	Reserved	

Sample serial number (SRL, SD) is valid only if individual samples are transmitted in Rectified and Pulsed measurement modes. These modes are not covered in this document.

Notes:

F If bit HST.3 is set to 1, then frequency has been newly measured; otherwise the value from the foregoing frequency measurement has been sent.

S Meaning of bytes SRL, SRH, RE depends on the value of bits HST.6, HST.0, HST.1 as follows:

- If HST.6=0
 - If HST.0=1 **or** HST.1=1 (transfer of individual samples in Rectified and Pulsed modes of operation)
 - SRL Sample serial number – LSB
 - SRH Sample serial number – MSB
 - RE not used (value irrelevant)
- If HST.6=1
 - If HST.0=0 **and** HST.1=0 (CW mode of operation or averaged results of Rectified and Pulsed modes of operation)
 - SRL Coded reflected power – LSB
 - SRH Coded reflected power – MSB
 - RE Coded reflected power – exponent
- Other combinations of HST.6, HST.0, HST.1
 - RE, SRL, SRH not used (values irrelevant)

4.1.4 Motors Data

Motors Data occupy one CAN message with ID = **ciMotS** = 15 = 0Fh or **ciMotSQ** = 22 = 16h.

ID	Sender	Byte Count
ciMotS =15=0Fh	Homer	8
ciMotSQ =22=16h	Homer	8

Byte	Name	Meaning
0	M1L	Motor 1 position – LSB
1	M1H	Motor 1 position – MSB
2	M2L	Motor 2 position – LSB
3	M2H	Motor 2 position – MSB
4	M3L	Motor 3 position – LSB
5	M3H	Motor 3 position – MSB
6	MS1	Motor Status, Part 1 (Section 4.5)
7	MS2	Motor Status, Part 2 (Section 4.6)

The message with ID = **ciMotSQ** is sent *only* as a response to the motor query (Section 6.4) or single-shot commands (Section 8). In all other cases the message with ID = **ciMotS** is sent.

4.2 Case of RS232

In case of RS232, data are sent in the form of Data Objects with end message `msMeasObject` = 16 = 10h, called *Measurement Data Objects* (MDO). The data contents of an MDO is a sequence of bytes MDO[0]...MDO[N-1] with variable length N. The maximum byte count is $N_{max} = 31$, i.e. MDO[0] to MDO[30]. Note, however, that the count of sent bytes (hence bytes which must be received) may be greater than MDO length N (see Section 3.2.3).

One MDO may contain either Homer Measurement Results or Motors Data or both. Information which components are present in MDO and which are not is included in its first byte, MDO[0], named Homer Status Byte (HST). Homer Status Byte is *always* present in MDO; its structure is given below in Section 4.3.

4.2.1 MDO Data Bytes

Byte	Name	Meaning	Note
0	HST	Homer Status Byte (Section 4.3)	A
1	HER	Homer Error Byte (Section 4.4)	H
2	PH	Coded incident power – MSB	H
3	PL	Coded incident power – LSB	H
4	PE	Coded incident power – exponent	H
5	TL	Internal temperature in units of 0.1 Celsius – LSB	H
6	TH	Internal temperature in units of 0.1 Celsius – MSB	H
7	RE	Coded reflected power – exponent	H,S
8	XL	Coded real part of input (measured) reflection coefficient – LSB	H
9	XH	Coded real part of input (measured) reflection coefficient – MSB	H
10	YL	Coded imaginary part input (measured) of reflection coefficient – LSB	H
11	YH	Coded imaginary part input (measured) of reflection coefficient – MSB	H
12	F0	Frequency in units of 10 Hz – Byte 0 (LSB)	H,F
13	F1	Frequency in units of 10 Hz – Byte 1	H,F
14	F2	Frequency in units of 10 Hz – Byte 2	H,F
15	F3	Frequency in units of 10 Hz – Byte 3 (MSB)	H,F
16	DXL	Coded real part of load reflection coefficient – LSB	H
17	DXH	Coded real part of load reflection coefficient – MSB	H
18	DYL	Coded imaginary part of load reflection coefficient – LSB	H
19	DYH	Coded imaginary part of load reflection coefficient – MSB	H
20	SRL	Sample serial number/Coded reflected power – LSB	H,S
21	SRH	Sample serial number/Coded reflected power – MSB	H,S
22	M1L	Motor 1 position – LSB	M
23	M1H	Motor 1 position – MSB	M
24	M2L	Motor 2 position – LSB	M
25	M2H	Motor 2 position – MSB	M
26	M3L	Motor 3 position – LSB	M
27	M3H	Motor 3 position – MSB	M
28	MS1	Motor Status, Part 1 (Section 4.5)	M
29	MS2	Motor Status, Part 2 (Section 4.6)	M
30	CS	Checksum byte	A

Notes:

- A** Byte is always present.
- H** Byte is present only if Homer Measurement Results (HMR) are included (bit HST.2 set to 1).
- F** If bit HST.3 is set to 1 then frequency has been freshly measured; otherwise the value from the foregoing frequency measurement has been sent.
- M** Byte is present only if Motors Data are included (bit HST.4 set to 1).
- S** Meaning of bytes SRL, SRH, RE depends on the value of bits HST.6, HST.0, HST.1 as follows:
 - If HST.6=0

- If HST.0=1 **or** HST.1=1 (transfer of individual samples in Rectified and Pulsed modes of operation)
 - SRL Sample serial number – LSB
 - SRH Sample serial number – MSB
 - RE not used (value irrelevant) but present if HMR are included in MDO
- If HST.6=1
 - If HST.0=0 **and** HST.1=0 (CW mode of operation or averaged results of Rectified and Pulsed modes of operation)
 - SRL Coded reflected power – LSB
 - SRH Coded reflected power – MSB
 - RE Coded reflected power – exponent
- Other combinations of HST.6, HST.0, HST.1
 - RE not used (value irrelevant) but present if HMR are included in MDO
 - SRL, SRH not present in MDO

4.2.2 Checksum Byte

Checksum Byte CS is the LSB of sum of all MDO bytes (except CS itself, of course):

$$CS = 255 \text{ and } \sum_{i=0}^{N-2} MDO[i]$$

After receiving MDO, checksum byte should be computed using the above formula and compared with the received CS byte. MDO should be rejected if these two values are not equal.

4.3 Homer Status Byte (HST)

Bits	Value	Meaning
0+1	0	a) CW mode of operation, or b) MDO is <i>not</i> a sample in pulsed measurements (carries averaged results)
	1	MDO is the first sample in pulsed measurements
	2	MDO is an intermediate sample in pulsed measurements
	3	MDO is the last sample in pulsed measurements
2	0	Homer Measurement Results are not included
	1	Homer Measurement Results are included
3	0	Frequency has been measured
	1	Frequency has not been measured: previous value is used
4	0	Motors Data are not included
	1	Motors Data are included
5	0	The MDO is a periodically sent data object
	1	The MDO is a response to motor query or a single-shot command
6	0	Byte RE irrelevant, bytes SRL, SRH used as sample S/No in pulsed modes when individual samples are transferred (b0=1 or b1=1)
	1	Bytes RE, SRL, SRH used for reflected power transfer (only if b0=0 and b1=0)
7	-	Not used

Notes:

- With CAN bus, only bits 0, 1, and 3 are relevant.
- Bit HST.3 set to 1 means that the frequency has been newly measured; otherwise the value from the foregoing measurement is copied. (In SrvHo.exe server versions V35 and less, the bit had a different meaning: it indicated the less important instance of fresh *temperature* measurement.)

4.4 Homer Error Byte (HER)

Bit	Meaning
0	Pulsed mode measurement error
1	A/D converter overflow in at least one of the four detector channels
2	Internal temperature is too low – below the calibration interval
3	Internal temperature is too high – above the calibration interval
4	Low Signal: signal level is too low to assure accurate measurement (warning)
5	Substitute frequency sent (warning)
6	Measurement data are (for any reason) invalid
7	Reserved

Notes:

- An error or warning condition is indicated by the corresponding bit set to 1.
- Warning is not a fatal error: measurement accuracy may, however, be degraded.
- Substitute frequency is sent when the counter is switched OFF or when the frequency measurement was in error.

4.5 Motor Status, Part 1 (MS1)

Bit	Value	Meaning
0	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)
1	0	Motor 2 not initialized
	1	Motor 2 successfully initialized
2	0	Motor 3 not initialized
	1	Motor 3 successfully initialized
3	-	not used
4	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
5	0	Motor 2 is not in the desired position
	1	Motor 2 has reached the desired position and is not moving
6	0	Motor 3 is not in the desired position
	1	Motor 3 has reached the desired position and is not moving
7	-	not used

4.6 Motor Status, Part 2 (MS2)

Bit	Value	Meaning
0	0	Motor 1 data valid
	1	Motor 1 error
1	0	Motor 2 data valid
	1	Motor 2 error
2	0	Motor 3 data valid
	1	Motor 3 error
3	-	not used
4	0	Stub Swap Alarm not raised
	1	Stub Swap Alarm raised
5	-	not used
6	-	not used
7	0	The message contains Actual Positions (AP)
	1	The message contains Tune Positions (TP)

Bits 0 through 2 of MS2 indicate motor errors. A motor error occurs e.g.

- after its unsuccessful initialization,

- when the motor reaches a terminal switch,
- when the motors are hard-stopped (Section 6.5).

If (MS2 and 7) = 0 then all motors are without error.

Bit 4 of MS2 is **Stub Swap Alarm** (SSA) warning; the bit is set to **1** if the next autotuning step would involve sizeable switching from the stub pair 1+2 to the pair 2+3 or vice versa. SSA is evaluated anytime Tune Positions are computed (either for the purpose of actual tuning or only for Tune Position sending). See Section 7.6 for more details on SSA.

Bit 7 of MS2 represents *SendTunePos* flag; the value **1** means that the message does not carry actual positions but Tune Positions, values computed for achieving impedance match.

4.7 Decoding Data Bytes

Note: Care must be taken with expressions like $256*B$ where B is byte and the result is expected to be 2-byte variable (word). This may, depending on programming language, lead to either run-time error or incorrect result (truncated to one byte). To avoid such problems, you should check the behavior of your program or first convert B to word, and then multiply: $256*word(B)$. Similar applies to byte vs. longint and word vs. longint.

Reflection Coefficient

To obtain complex reflection coefficient $R = X + jY$, proceed as follows:

- Create 16-bit *signed integer* XS from the bytes XL, XH (values between -32768 and 32767), e.g. in Pascal notation $XS = smallint(XL + 256*XH)$; see also Section 1.2.1.
- In a similar way, create signed integer YS from the bytes YL, YH.
- Then

$$X = XS/4096.0$$

$$Y = YS/4096.0$$

Frequency

Frequency F in Hz is computed as follows:

$$F = (F0 + 256*(F1 + 256*(F2 + 256*F3)))*10$$

Incident Power

Incident power Pi in watts is computed as follows:

$$Pi = (PL + 256* PH)*10^{(PE-10)}$$

Reflected Power

If bytes RE, SRL, SRH are used for reflected power transfer (see e.g. notes to Section 4.2.1), the reflected power Pr in watts is computed as follows:

$$Pr = (SRL + 256* SRH)*10^{(RE-10)}$$

Temperature

Internal temperature T in degrees Celsius is computed as follows:

$$T = smallint (TL + 256*TH)/10.0$$

4.7.1 Some Derived Quantities

Squared magnitude of reflection coefficient (used in power computations below)

$$S = X^2 + Y^2$$

Magnitude of reflection coefficient

$$M = \sqrt{S}$$

Return Loss

$$RL = -20\log(M)$$

Voltage standing wave ratio

$$VSWR = \frac{1+M}{1-M}$$

Reflected power (applicable when bytes RE, SRL, SRH are *not* used for reflected power transfer, see e.g. notes to Section 4.2.1)

$$P_r = P_i S = P_i (X^2 + Y^2)$$

Power absorbed in load

$$P_a = P_i - P_r = P_i(1 - S) = P_i(1 - X^2 - Y^2)$$

4.7.2 Phase of Reflection Coefficient

To obtain correct phase ϕ of reflection coefficient in all four quadrants, *do not* be tempted to apply standard *arctangent* function on the ratio Y/X of imaginary (Y) and real (X) reflection coefficient parts. In programming languages the *arctangent* function is usually denoted as **arctan(x)** (e.g. Pascal) or **atan(x)** (e.g. C).

Instead, many languages support a function of two variables, denoted for instance as **arctan2(y, x)** (e.g. Pascal) or **atan2(y, x)** (e.g. C), yielding directly the phase angle in radians in proper quadrant. The usage is

$$\phi_{rad} = \arctan2(Y, X) \quad (\text{radians})$$

$$\phi_{deg} = \frac{180}{\pi} \phi_{rad} \quad (\text{degrees})$$

If the programming language supports only the standard *arctangent* function, use the following routine (written in Pascal notation):

```
if abs(X)<1E-20 {to avoid division by zero}
then begin
  if Y>0.0
  then phase:=PI/2.0
  else phase:=-PI/2.0
end
else begin
  a:=arctan(Y/X); {interim result}
  if X>0.0
  then phase:=a
  else if Y<0.0
  then phase:=a-PI
  else phase:=a+PI
end;
phase_dg:=phase*180.0/pi; {degrees}
```

If the programming language supports only the standard *arccosine* function, use the following routine:

```
r:=sqrt(X*X+Y*Y); {reflection coefficient magnitude}
if r<1E-20 then begin
  phase:=0.0 {phase undefined}
end
else begin
  if Y>=0.0
  then phase:=arccos(X/r)
  else phase:=-arccos(X/r);
end;
phase_dg:=phase*180/pi;
```

5. HOMER ANALYZER CONTROL

This section describes commands controlling the behavior of the *measurement* portion of the system (Homer Analyzer). See Section 4 for the measurement data structure.

5.1 Start Continuous Measurement

Description: Homer starts making measurements and associated actions (*Measure-Tune-Send* sequences, or *MTS sequences*) repeatedly in that it sets its *Running* and *Sending* flags to TRUE. Measurement may be asynchronous in the sense that MTS sequence follows immediately the previous one. Another option is not to start a new MTS sequence until a defined time (see Section 10.1.1) has elapsed.

Duration of an MTS sequence depends on the system settings (like averaging, frequency counting interval, whether tuning stubs are moved, etc.).

The actions executed in one MTS sequence depend on **Autotune** and **SendTunePos** flag settings (see Sections 7.3 and 7.5, respectively); they can include:

#	Action	Executed
1	Measurement	Always
2	Computing Tune Positions	if Autotune = TRUE <i>or</i> SendTunePos = TRUE
3	Moving stubs to Tune Positions	if Autotune = TRUE
4	Sending Homer Measurement Results	Always
5	Sending Actual Positions	Always
6	Sending Tune Positions	if SendTunePos = TRUE <i>and</i> Autotune = FALSE

The measurement outcome is Homer Measurement Results HMR. Input data for the computations of Tune Positions are HMR and Actual Positions of the motors AP.

After issuing the command, your controller can repeatedly receive and process CAN messages or RS232 MDOs. To stop continuous measurement, send Stop Measurement command.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSweepStart =17=11h

Homer Response:

As the first thing, a command confirmation message is returned, containing actual values of Homer's *Running* and *Sending* states (See Section 11.1), which are in this case both 1:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	3

Data bytes:

Byte	Value/Meaning
0	msSweepStart =17=11h
1	1 (Running = TRUE)
2	1 (Sending = TRUE)

Following this confirmation (you may choose to ignore it), a periodic stream of messages is sent with the following structure (see Section 4 for the meaning of data bytes contained in the individual messages):

ID	Sender	Note
ciHmRes1 =11=0Bh	Homer	
ciHmRes2 =12=0Ch	Homer	
ciHmRes3 =13=0Dh	Homer	
ciMotS =15=0Fh	Homer	AP
ciMotS =15=0Fh	Homer	TP; sent only if SendTunePos = TRUE <i>and</i> Autotune = FALSE

Notes:

- Byte count of all messages is 8.
- The fifth message with Tune Positions is sent *only* if the flags **SendTunePos** = TRUE and **Autotune** = FALSE. Whether a message contains AP or TP is distinguished by bit 7 of MS2: 0=>AP, 1=>TP (Section 4.6).
- If **Autotune** = TRUE, Homer Measurement Results correspond to stub positions before their shifting to new positions while AP sent are the new positions.
- See Section 11.1 about extended use of **msSweepStart** = 17 = 11h command (setting of *Running* and *Sending* flags).

Example:

The command:

```
16: 17
```

The response example for HST = 4, HER = 0, Pinc = 23.42 mW, Temp = 25.4 C, Measured Re = 0.05225, Im = 0.3105, F = 2454.11 MHz, Load (deembedded) Re = 0.21753, Im = -0.02905, AP = (2583, 1571, 0) steps, TP = (0, 513, 4000) steps, AP sent (e.g. **SendTunePos** = FALSE, **Autotune** = FALSE):

```
18: 17 1 1
11: 4 0 9 38 5 254 0 255
12: 214 0 248 4 184 172 160 14
13: 123 3 137 255 139 0 0 0
15: 23 10 35 6 0 0 119 0
11: 4 0 9 38 5 254 0 255
12: 214 0 248 4 184 172 160 14
13: 123 3 137 255 139 0 0 0
15: 23 10 35 6 0 0 119 0
etc...
```

The response in case when TP are sent (**SendTunePos** = TRUE, **Autotune** = FALSE):

```
18: 17 1 1
11: 4 0 9 38 5 254 0 255
12: 214 0 248 4 184 172 160 14
13: 123 3 137 255 139 0 0 0
15: 23 10 35 6 0 0 119 0
15: 0 0 1 2 160 15 119 128
11: 4 0 9 38 5 254 0 255
12: 214 0 248 4 184 172 160 14
13: 123 3 137 255 139 0 0 0
15: 23 10 35 6 0 0 119 0
15: 0 0 1 2 160 15 119 128
```

etc...

RS232

Command Code	msSweepStart =17=11h
Label	none
Parameters	none
Response	Command Execution Confirmation (Section 3.4.3) followed by MDOs repeatedly until Stop Measurement command

Example:

The command is transmitted as the byte sequence

128, 17

The response example for HST = 4, HER = 0, Pinc = 23.42 mW, Temp = 25.4 C, Measured Re = 0.05225, Im = 0.3105, F = 2454.11 MHz, Load (deembedded) Re = 0.21753, Im = -0.02905, AP = (2583, 1571, 0) steps, TP = (0, 513, 4000) steps, AP sent (e.g. **SendTunePos** = FALSE, **Autotune** = FALSE):

128, 28, 17, 0 128, 4

128, 28, 20, 0, 9, 38, 5, 254, 0, 255, 214, 0, 248, 4, 184, 172, 160, 14,
123, 3, 137, 255, 23, 10, 35, 6, 0, 0, 119, 0, 240, 128, 16,

128, 28, 20, 0, 9, 38, 5, 254, 0, 255, 214, 0, 248, 4, 184, 172, 160, 14,
123, 3, 137, 255, 23, 10, 35, 6, 0, 0, 119, 0, 240, 128, 16,

etc...

The first line is Command Execution Confirmation. MDO starts by **msDataBegin** command (128, 28), followed by Homer Status Byte HST (20) and the rest of data. MDO is terminated by **msMeasObject** = 16 = 10h command (128, 16) preceded by checksum byte CS (240).

The response in case when TP are sent (**SendTunePos** = TRUE, **Autotune** = FALSE):

128, 28, 17, 0 128, 4

128, 28, 20, 0, 9, 38, 5, 254, 0, 255, 214, 0, 248, 4, 184, 172, 160, 14,
123, 3, 137, 255, 23, 10, 35, 6, 0, 0, 119, 0, 240, 128, 16,

128, 28, 16, 0, 0, 1, 2, 160, 15, 119, 128, 128, 185, 128, 16,

128, 28, 20, 0, 9, 38, 5, 254, 0, 255, 214, 0, 248, 4, 184, 172, 160, 14,
123, 3, 137, 255, 23, 10, 35, 6, 0, 0, 119, 0, 240, 128, 16,

128, 28, 16, 0, 0, 1, 2, 160, 15, 119, 128, 128, 185, 128, 16,

etc...

Note that an additional MDO is sent containing only Motors Data. Note also the doubling of MS2 = 128 before the checksum byte CS = 185 (the sequence 128, 128, 185).

5.2 Stop Measurement

Description: Homer stops continuous measurement and associated actions, like impedance matching in that it sets its *Running* and *Sending* flags to FALSE. To restore measurement, send Start Measurement command.

Note: If Homer is busy with a time-consuming task (e.g. pulsed signal sampling or Tuner moving the stubs), the response may come later than a timeout set in your PC program. An adaptive timeout or several attempts to intercept the response may help.

CAN

Command Message:

ID	Sender	Byte Count
ciHmStop =10=0Ah	PC	1

Data bytes:

Byte	Value/Meaning
0	msSweepStop =18=12h

Note that the command has a special ID (**ciHmStop** = 10 = 0Ah rather than = 16 = 10h).

Homer Response:

Homer responds with a message identical with the command:

ID	Sender	Byte Count
ciHmStop =10=0Ah	Homer	1

Data bytes:

Byte	Value/Meaning
0	msSweepStop =18=12h

RS232

Command Code	msSweepStop =18=12h
Label	none
Parameters	none
Response	Command Execution Confirmation (Section 3.4.3)

5.3 Averaging

Description: Modifies Homer averaging numbers. Each averaging number is represented by a 2-byte integer; accepted values range from 1 to 4096. *Detector voltages measurement averaging* concerns only CW mode of sampling and is the sample count taken for one measurement point in CW mode at a given sampling frequency (Section 5.6).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	msHmAvg =57=39h
1	Detector voltages measurement averaging for CW sampling mode – LSB
2	Detector voltages measurement averaging for CW sampling mode – MSB
3	Temperature measurement averaging – LSB
4	Temperature measurement averaging – MSB

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	5

Data bytes: The same structure as above, returning actual values. If an error occurred in Homer while executing the command, the most significant bit of Byte 0 (returned command code) is set to 1.

Note: Starting with SrvHo firmware version V36, bytes 5 and 6 are no more used. In older versions, these bytes represented Temperature measurement cycle Tcycl; this obsolete quantity has been substituted by Temperature Measurement Period (Section 10.1.4).

Example:

To set Vavg = 256, Tavg = 8, the command message is

16: 57 0 1 8 0

The response is

18: 57 0 1 8 0

In case of error executing the command the response might be

18: 185 0 1 8 0 208 7

RS232

Command Code	msHmAvrg =57=39h
Label	AVR
Parameters	Detector voltages measurement averaging for CW sampling mode (Vavg) Temperature measurement averaging (Tavg)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set Vavg = 256, Tavg = 8, the command string is "**AVR 256 8**"; the complete command is transmitted as the sequence

128, 28, 65, 86, 82, 32, 50, 53, 54, 32, 56, 13, 10, 128, 57

The response may be

128, 28, 57, 0, 128, 4

The first data byte (57) is the replica of the command code **msHmAvrg** = 57 = 39h. The second data byte (0) is error code, in this case indicating no error.

5.4 Frequency Counter

Description:

- Defines count time for CW signal waveform.
- Turns Homer frequency counter ON or OFF (for any signal waveform).

Count time for CW signal waveform is expressed in microseconds as a 4-Byte integer (longint in Pascal notation). The accepted count time range is 16 microseconds to 1 second.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	6

Data bytes:

Byte	Value/Meaning
0	msHmCounter =56=38h
1	Count time for CW sampling mode – Byte 0 (LSB)
2	Count time for CW sampling mode – Byte 1
3	Count time for CW sampling mode – Byte2
4	Count time for CW sampling mode – Byte 3 (MSB)
5	0 : Counter OFF; 1 : Counter ON

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes: The same structure as above, returning actual values. If an error occurred in Homer while executing the command, the most significant bit of Byte 0 (returned command code) is set to 1.

Example:

To set TcountCW = 10 ms = 10000 μ s and switch the counter ON, the command message is

16: 56 16 39 0 0 1

The response is

18: 56 16 39 0 0 1

RS232

Command Code	msHmCounter=56=38h
Label	xxx
Parameters	Count time for CW sampling mode (TcountCW) 0 = Counter OFF, 1 = Counter ON
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set TcountCW = 10 ms = 10000 μ s and switch the counter ON, the command string is "xxx 10000 1"; the complete command is transmitted as the sequence

128, 28, 88, 88, 88, 32, 49, 48, 48, 48, 48, 32, 49, 13, 10, 128, 56

Response:

128, 28, 56, 0, 128, 4

5.5 Set Substitute Frequency

Description: Defines the frequency Homer should send in a case when:

- Frequency counter is disabled.
- Frequency measurement error occurred.
- Measured frequency is out of bounds: differs from nominal frequency more than tolerated.

CAN

The frequency to be set is expressed in units of kHz as a 4-Byte integer (longint in Pascal notation). Homer responds with a one-data-byte message, returning the command byte.

Command Message:

ID	Sender	Byte Count
ciHomC=16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	msSetFsubst=7
1	Substitute frequency in kHz – Byte 0 (LSB)
2	Substitute frequency in kHz – Byte 1
3	Substitute frequency in kHz – Byte 2
4	Substitute frequency in kHz – Byte 3 (MSB)

Homer Response:

ID	Sender	Byte Count
ciHomR=18=12h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msSetFsubst=7

Example:

To set $F_{\text{subst}} = 2450 \text{ MHz} = 2450000 \text{ kHz}$, the command message is

16: 7 80 98 37 0

The response is

18: 7

RS232

The substitute frequency to be set is expressed in units of kHz.

Command Code	<code>msSetFsubst=7</code>
Label	FRE
Parameters	Substitute frequency in kHz (F_{subst})
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set $F_{\text{subst}} = 2450 \text{ MHz} = 2450000 \text{ kHz}$, the command string is "**FRE 2450000**"; the complete command is transmitted as the sequence

128, 28, 70, 82, 69, 32, 50, 52, 53, 48, 48, 48, 48, 13, 10, 128, 7

Response:

128, 28, 7, 0, 128, 4

5.6 Set CW Sampling Frequency

Description: Defines the sampling frequency F_{saml} for A/D conversion in CW mode of sampling (10 Hz to 200 kHz).

CAN

The frequency to be set is expressed in units of Hz as a 4-Byte integer (longint in Pascal notation). Homer responds with a one-data-byte message, returning the command byte.

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	<code>msSetFsamlCw=75=4Bh</code>
1	Sampling frequency in Hz – Byte 0 (LSB)
2	Sampling frequency in Hz – Byte 1
3	Sampling frequency in Hz – Byte 2
4	Sampling frequency in Hz – Byte 3 (MSB)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	1

Data bytes:

Byte	Value/Meaning
0	<code>msSetFsamlCw=75=4Bh</code>

Example:

To set $F_{\text{saml}} = 100 \text{ kHz} = 100000 \text{ Hz}$, the command message is

16: 75 160 134 1 0

The response is

18: 75

RS232

The sampling frequency to be set is expressed in hertz.

Command Code	msSetFsmp1Cw =75=4Bh
Label	FRE
Parameters	Sampling frequency in Hz (Fsampl)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set Fsampl = 100 kHz = 100000 Hz, the command string is "**FRE 100000**"; the complete command is transmitted as the sequence

128, 28, 70, 82, 69, 32, 49, 48, 48, 48, 48, 48, 13, 10, 128, 75

Response:

128, 28, 75, 0, 128, 4

5.7 Set Frequency Tolerance

Description: Defines the maximum acceptable deviation of measured frequency from nominal frequency (if exceeded, Homer will send the substitute frequency rather than the measured frequency).

CAN

The frequency tolerance to be set is expressed in units of MHz as a 4-Byte integer (longint in Pascal notation). Homer responds with a one-data-byte message, returning the command byte.

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	msSetFdelta =6
1	Frequency tolerance in MHz – Byte 0 (LSB)
2	Frequency tolerance in MHz – Byte 1
3	Frequency tolerance in MHz – Byte 2
4	Frequency tolerance in MHz – Byte 3 (MSB)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msSetFdelta =6

Example:

To set Fdelta = 50 MHz, the command message is

16: 6 50 0 0 0

The response is

18: 6

RS232

The frequency tolerance to be set is expressed in units of MHz.

Command Code	msSetFdelta =6
Label	FRE
Parameters	Frequency tolerance in MHz (Fdelta)
Response	Command Execution Confirmation (Section 3.4.3)

Example: To set Fdelta = 50 MHz, the command string is "**FRE 50**"; the complete command is transmitted as the sequence

128, 28, 70, 82, 69, 32, 53, 48, 13, 10, 128, 6

Response:

128, 28, 6, 0, 128, 4

5.8 Switch Waveform

Description: Changes mode of sampling of the signal waveform (CW, Rectified, Pulsed).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	2

Data bytes:

Byte	Value/Meaning
0	msHmWform =53=35h
1	Desired signal waveform sampling mode (0 : CW; 1 : Rectified; 2 : Pulsed)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msHmWform =53=35h
1	Actual sampling mode (0 : CW; 1 : Rectified; 2 : Pulsed)

Example:

To set Waveform = Pulsed, the command message is

16: 53 2

The response is

18: 53 2

If Pulsed option is not available, the response would be CW mode:

18: 53 0

RS232

Command Code	msHmWform =53=35h
Label	SIG
Parameters	Desired signal waveform sampling mode (0 : CW; 1 : Rectified; 2 : Pulsed)
Response	Command Execution Confirmation (Section 3.4.3)

Example: To set Waveform = Rectified, the command string is "**SIG 1**"; the complete command is transmitted as the sequence

128, 28, 83, 73, 71, 32, 1, 13, 10, 128, 53

Response:

128, 28, 53, 0, 128, 4

6. STEPPING MOTORS

This section describes messages enabling to set the tuning stubs to defined positions (in terms of motor steps relative to the reference position) as well as read the current motor positions and their status.

Motor 1 is the one closest to the signal source (magnetron). Motor 3 is the one closest to the load. Motor 2 is between Motor 1 and Motor 3.

6.1 Motors Selection Map

Motors Selection Map (MSM) byte selects the motors a command will act upon. MSM has only three relevant bits. To select a motor, set the corresponding bit to **1**. MSM byte is used only in CAN bus commands.

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

6.2 Motors Initialization

Description: *Initialization routine* (see Section 1.1) is executed with selected motors (tuning stubs); the selected stubs are set to the reference (zero) positions. Since initialization may take relatively long time (depending on motor speed 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 their initialization (Section 6.4).

CAN

Command Message:

ID	Sender	Byte Count
ciHmStop =10=0Ah or eiMotC	PC	2

Note: Use messages with ID = **ciHmStop** = 10 = 0Ah. The case ID = **ciMotC** = 14 = 0Eh serves for backward compatibility only.

Data bytes:

Byte	Value/Meaning
0	msMotInit =69=45h
1	Motors Selection Map MSM

Note: To initialize all motors (All Stubs Home, Reset Stubs), set Byte 1 equal to **7**.

Homer Response:

There is no response to this command for firmware (Server) versions V52 and less.

For Server versions starting V53 (22-Sep-2010), the response is as follows:

ID	Sender	Byte Count
ciHmStop =10=0Ah	Homer	2

Data bytes:

Byte	Value/Meaning
0	msMotInit =69=45h (incremented by 128=80h in case of command fail)
1	Error code (0 : Success; 1 : Fail)

Example 1:

All Stubs Home (initialize all motors). This is an example with easily verifiable execution: the result must be the same as pressing the *All Stubs Home* button on your Homer device. Motors Selection Map for this case is 111 binary, i.e. 7 decadic.

Command message

10: 69 7

Response (success)

10: 69 0

In case of error executing the command the response would be (197 = 69 + 128)

10: 197 1

Example 2:

Initialize Motors 2 and 3. MSM for this case is 110 binary, i.e. 6 decadic.

Command message

10: 69 6

Response (success)

10: 69 0

RS232

In case of RS232, you can select either all motors or one motor.

6.2.1 All Stubs Home

Description: *Initialization routine* is executed with all motors.

Command Code	msMotInit =69=45h
Label	none
Parameters	none
Response	<ul style="list-style-type: none">Server V52 and less: NoneServer V53 and more: Command Execution Confirmation (Section 3.4.3)

Example:

To initialize all motors, just send the **msMotInit** command:

128, 69

The response for Server V53 and more may be

128, 28, 69, 0, 128, 4

The first data byte (68) is the replica of the command code. The second data byte (0) is error code, in this case indicating no error.

6.2.2 Selected Stub Home

Description: *Initialization routine* is executed with one selected motor.

Command Code	msOneHome =70=46h
Label	M1H
Parameters	Selected motor number (1 , 2 , or 3)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To initialize Motor 2, the command string is "**M1H 2**"; the complete command is transmitted as the sequence

128, 28, 77, 49, 72, 32, 50, 13, 10, 128, 70

Response:

128, 28, 70, 0, 128, 4

6.3 Set Motor Positions

Description: Selected motors are moved to desired positions. Each position is represented by a 2-byte integer equal to the number of motor steps from the reference position. The step size and the maximum allowable step count can be learned using your *HomSoft* Windows visualization and control program (**Motors** page of the **System Info** window).

Homer responds with Motors Data message. You may use this response or ignore it and call Read Motor Positions function afterwards.

CAN

Command Message:

ID	Sender	Byte Count
ciMotC =14=0Eh	PC	8

Data bytes:

Byte	Value/Meaning
0	msMotSet =71=47h
1	Motors Selection Map
2	Motor 1 desired position – LSB
3	Motor 1 desired position – MSB
4	Motor 2 desired position – LSB
5	Motor 2 desired position – MSB
6	Motor 3 desired position – LSB
7	Motor 3 desired position – MSB

Homer Response:

ID	Sender	Byte Count
ciMotSQ =22=16h	Homer	8

Data bytes: See Section 4.1.4.

Note: This command cannot be broadcast because the message contains more than 7 data bytes (see Section 2.6).

Example:

To set motors 1, 2, 3 to 0, 513, and 4000 steps from zero position, respectively, MSM = 7 and the command message is

14: 71 7 0 0 1 2 160 15

Response:

22: 0 0 1 2 160 15 119 0

RS232

Command Code	msMotSet =71=47h
Label	MPO
Parameters	Motor 1 position
	Motor 2 position

	Motor 3 position
Response	MDO with motors data, bits 4 and 5 of HST set to 1

After issuing the command, the controller should wait for and receive MDO which

- contains motors data (i.e. bit 4 of HST is **1**), *and*
- is response to a single-shot command (i.e. bit 5 of HST is **1**).

Details of HST and MDO structure see in Sections 4.3 and 4.2.1, respectively.

Example:

To set motors 1, 2, 3 to 0, 513, and 4000 steps from zero position, respectively, the command string is "**MPO 0 513 4000**". The complete command is transmitted as the sequence

128, 28, 77, 80, 79, 32, 48, 32, 53, 49, 51, 32, 52, 48, 48, 48, 13, 10,
128, 71

Response:

128, 28, 48, 0, 0, 1, 2, 160, 15, 119, 0, 89, 128, 16

6.4 Read Motor Positions

Description: Queries for current motor positions and status.

CAN

Command Message:

ID	Sender	Byte Count
ciMotSQ =22=16h or eiMotC	PC	1

Note: Using message with ID = **ciMotSQ** is preferable in order not to possibly overwrite a previous message with ID = **ciMotC**.

Data bytes:

Byte	Value
0	msMotRead =74=4Ah

Homer Response:

Motors Data – a message with ID = **ciMotSQ** = 22 = 16h (see Section 4.1.4)

Example:

Command message:

22: 74

Response example (motors 1, 2, 3 set to 0, 513, and 4000 steps, respectively):

22: 0 0 1 2 160 15 119 0

RS232

Homer responds by sending a Measurement Data Object (MDO) containing motors data. Bit 5 of Homer Status Byte HST = MDO[0] is set to **1** (which indicates that this particular MDO has been sent as the response to Read Motor Positions command).

Command Code	msMotRead =74=4Ah
Label	none
Parameters	none
Response	MDO with motors data, bits 4 and 5 of HST set to 1

After issuing the command, the controller should wait for and receive MDO which

- contains motors data (i.e. bit 4 of HST is **1**), *and*

- is response to motors query (i.e. bit 5 of HST is **1**).

Details of HST and MDO structure see in Sections 4.3 and 4.2.1, respectively.

Example:

Command:

128, 74

Response example (motors 1, 2, 3 set to 0, 513, and 4000 steps, respectively):

128, 28, 48, 0, 0, 1, 2, 160, 15, 119, 0, 89, 128, 16

6.5 Hard Stop of Motors

Description: All motors are immediately stopped and disconnected from power supply. Following this, motors should be initialized because the track of their current positions has been lost. There is no response from Homer to this command.

CAN

Command Message:

ID	Sender	Byte Count
ciMotC =14=0Eh	PC	1

Data bytes:

Byte	Value/Meaning
0	msMotStop =19=13h

RS232

Command Code	msMotStop =19=13h
Label	none
Parameters	none
Response	none

7. AUTOTUNING

7.1 Autotuning Control Parameters

Description: Sets parameters controlling the autotuning process. For their meaning, see e.g. Help for Homer (HomerHelp.chm), topic [Autotuner > System Info Window > Autotune](#).

Accepted range of values:

- Tolerance 0 to 1000
- Skipped measurements 0 to 255
- Target 0 to 1000
- Smoothing 1 to 255
- Delay 0 to 31 (5 bits)

Note: This command cannot be broadcast because the message contains more than 7 data bytes (see Section 2.6).

7.1.1 Server Versions Starting V55

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	8

Data bytes:

Byte	Value/Meaning
0	msCanSetAtPar =3
1	Tolerance in milliunits – LSB
2	Tolerance in milliunits – MSB
3	Skipped measurements
4	Smoothing
5	b7 – b3 Delay (0 to 31 shifted left by 3 bits)
	b2, b1 Not used (set them to 0)
	b0 Wait when RF power is low (0 = NO; 1 = YES)
6	Target in milliunits – LSB
7	Target in milliunits – MSB

Homer Response:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msCanSetAtPar =3

Example:

To set Toler = 25 mU, Skip = 10, Smooth = 8, Delay = 12, WaitRF = YES, Targ = 150 mU, the command message is:

17: 3 25 0 10 8 **97** 150 0

Byte 5 has been obtained as 97 = **01100001** binary, where the upper five bits represent Delay = 12 = 01100 and the least significant bit (1 on the left) represents WaitRF.

Response example:

19: 3

RS232

Command Code	msATunPar=73=49h
Label	ATP
Parameters	Toler: Tolerance in milliunits (0 to 1000) Skip: Skipped measurements (0 to 255) Smooth: Smoothing count (1 to 255) WaitRF: Wait when RF power is low (0 = NO; 1 = YES) Targ: Target in milliunits (0 to 1000) Delay: Tuning delay (0 to 31)
Response	Command Execution Confirmation (Section 3.4.3)

Note: For YES option in **WaitRF** setting, any normal text string starting with **y, Y, t, T, 1**, can be used, e.g. **yes, TRUE**, etc. All other cases (e.g. **NO, false, 0**, etc.) will be evaluated as NO.

Example:

To set Toler = 25 mU, Skip = 10, Smooth = 8, WaitRF = NO, Targ = 150 mU, Delay = 12, the command string may be "**ATP 25 10 8 N 150 12**"; the complete command is transmitted as the sequence

128, 28, 65, 84, 80, 32, 50, 53, 32, 49, 48, 32, 56, 32, 78, 32, 49, 53, 48,
32, 49, 50 13, 10, 128, 73

Response:

128, 28, 73, 0, 128, 4

7.1.2 Older Server Versions, Including V54

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC=17=11h	PC	8

Data bytes:

Byte	Value/Meaning
0	msCanSetAtPar=3
1	Tolerance in milliunits – LSB
2	Tolerance in milliunits – MSB
3	Skipped measurements
4	Wait until Tolerance is exceeded (0 = NO; 1 = YES) Ignored starting Server version V50 (always forced to YES)
5	Wait when RF power is low (0 = NO; 1 = YES)
6	Target in milliunits – LSB
7	Target in milliunits – MSB

Homer Response:

ID	Sender	Byte Count
ciAtunR=19=13h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msCanSetAtPar=3

Example:

To set Toler = 25 mU, Skip = 10, WaitTol = YES, WaitRF = NO, Targ = 150 mU, the command message is:

17: 3 25 0 10 1 0 150 0

Response example:

19: 3

RS232

Command Code	msATunPar =73=49h
Label	ATP
Parameters	Toler: Tolerance in milliunits
	Skip: Skipped measurements
	WaitTol: Wait until Tolerance is exceeded (0 = NO; 1 = YES) Ignored starting Server version V50 (always forced to YES)
	WaitRF: Wait when RF power is low (0 = NO; 1 = YES)
	Targ: Target in milliunits
Response	Command Execution Confirmation (Section 3.4.3)

Note: As YES in **WaitTol** and **WaitRF** setting, any normal text string starting with **y, Y, t, T, 1**, can be used, e.g. **yes, TRUE**, etc. All other cases (e.g. **NO, false, 0**, etc.) will be evaluated as NO.

Example:

To set Toler = 25 mU, Skip = 10, WaitTol = YES, WaitRF = NO, Targ = 150 mU, the command string may be "**ATP 25 10 Y N 150**"; the complete command is transmitted as the sequence

128, 28, 65, 84, 80, 32, 50, 53, 32, 49, 48, 32, 89, 32, 78, 32, 49, 53, 48,
13, 10, 128, 73

Response:

128, 28, 73, 0, 128, 4

7.2 Hysteresis

Description: Sets **Hysteresis** parameter for the autotuning in degrees. For the meaning of Hysteresis, consult Help for Homer (HomerHelp.chm), topic [Autouner > Hysteresis](#).

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	3

Data bytes:

Byte	Value/Meaning
0	msTuSetOther =96=60h
1	1 (fixed specifier)
2	Hysteresis in degrees t (0 to 255)

Homer Response:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	3

Data bytes: Same as above; actual Hysteresis value returned in Byte 2.

Example:

To set Hysteresis = 7 degrees, the command message is:

17: 96 1 7

Response:

19: 96 1 7

RS232

Command Code	msTuSetOther=96=60h
Label	TSO
Parameters	1 (fixed specifier) Hysteresis in degrees t (0 to 255)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set Hysteresis = 7 degrees, the command string is "TSO 1 7"; the complete command is transmitted as the sequence

128, 28, 84, 83, 79, 32, 49, 32, 55, 13, 10, 128, 96

Response:

128, 28, 96, 0, 128, 4

7.3 Continuous Autotuning

Description:

This command sets the **Autotune** flag to TRUE or FALSE. The **Autotune** state affects the system behavior only when *continuous* measurement is running (see Section 5.1). When **Autotune** = TRUE, an autotuning attempt is made after each measurement. Whether or not it will be actually carried out depends on the measured reflection coefficient and the conditions defined by Autotuning Control Parameters.

Starting with Server version V54 (Jan-2011), this command also enables *querying* current **Autotune** state without changing it.

Notes:

- The software setting of **Autotune** flag as described in this section can be overridden by *changing* the state of the Tuner's hardware **AutoTune** switch. Vice versa, the software command overrides the HW switch setting. In this way you can always combine setting of **Autotune** flag manually and programmatically.
- Autotuning can only take place after continuous measurement has started. The line

AUTORUN=3

in **Hom.cfg** file determines whether continuous measurement will start or not after Homer power-up. Bit 0 of the constant on the right-hand side sets the *Running* flag, bit 1 sets the *Sending* flag (see Section 11.1). For details about **Hom.cfg** file see Help for Homer (HomerHelp.chm), topic [Advanced Topics > Homer Analyzer Starting Parameters](#).

To start measurement programmatically, use **Start Measurement** command (Section 5.1).

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC=17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msAtOFF=0 To switch autotuning OFF msAtON=1 To switch autotuning ON msGetAtune=5 To query autotuning state (Server version V54 and more only)

Homer Response:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	1

Data bytes:

Byte	Value/Meaning
0	Same as in the command
1	0 if Autotune = FALSE 1 if Autotune = TRUE

Example 1:

Command message (Set **Autotune** to FALSE):

17: **0**

Response (command success):

19: **0** **0** (**Autotune** = FALSE)

Response (command fail):

19: **128** **0**

In this case, Byte 0 of the response has been incremented by 128 (see Sec. 2.4) and the value of Byte 1 is irrelevant.

Example 2:

Command message (Set **Autotune** to TRUE):

17: **1**

Response:

19: **1** **1** (**Autotune** = TRUE)

Example 3:

Command message (Query **Autotune** state – Server V54 and more only):

17: **5**

Possible responses:

19: **5** **0** (**Autotune** = FALSE)

19: **5** **1** (**Autotune** = TRUE)

19: **133** **0** (command fail, Byte 2 irrelevant)

RS232

Command Code	msATunCmd =72=48h
Label	ATC
Parameters	0 Autotuning OFF 1 Autotuning ON 2 Query Autotuning state (Server V54 and more only)
Response	Command Execution Confirmation (Section 3.4.3); in case of query modified as described below

Response to Query

In case of query (command **ATC 2**), the meaning of second data byte (B2) of the response is modified as follows:

0: **Autotune** = FALSE

1: Error executing the command

2: **Autotune** = TRUE

Example 1:

To switch continuous autotuning OFF and ON, the command string is "**ATC 0**" and "**ATC 1**", respectively. The complete commands are transmitted as the following byte sequences:

128, 28, 65, 84, 67, 32, 48, 13, 10, 128, 72 to switch autotuning OFF
 128, 28, 65, 84, 67, 32, 49, 13, 10, 128, 72 to switch autotuning ON

Response:

128, 28, 72, 0, 128, 4 (command execution OK)
 128, 28, 72, 1, 128, 4 (command execution fail)

Example 2:

(Server version V54 and more only.) To query **Autotune** state, the command string is "**ATC 2**". The complete command is transmitted as the following byte sequence:

128, 28, 65, 84, 67, 32, 48, 13, 10, 128, 72 to query **Autotune** state

Response:

128, 28, 72, 0, 128, 4 (**Autotune** = FALSE)
 128, 28, 72, 2, 128, 4 (**Autotune** = TRUE)
 128, 28, 72, 1, 128, 4 (**command execution fail!!!**)

7.4 Single Autotuning Step

Description: Triggers one autotuning step. The command does *not* initiate measurement: the autotuning uses the latest measured data available. It is therefore meaningful to use it only in continuous measurement with autotuning switched off.

Notes:

- The command is meaningful only in continuous measurement with continuous autotuning switched off.
- Since the autotuning step may take long (depending on the needed stubs travel distance and speed), you should allow ample time for the response arrival.
- To achieve good match of a grossly mismatched load, send at least three commands consecutively, each preceded by measurement. The same holds for a load that has much changed since the last autotuning step that took place. For fine-tuning, one command will usually do.
- Consult Section 8.5 on how to launch measurement + autotuning.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msAtSingle =2

Homer Response:

The response consists of two consecutive messages:

- Motors Data – a message with ID = **ciMotSQ** = 22 = 16h (see Section 4.1.4) and motors position after the tuning step
- Command confirmation message as follows:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msAtSingle =2

After issuing the command, the controller should wait for and receive a Motors Data message with CAN ID = **ciMotSQ** = 22 = 16h. (Details of Motors Data structure see in Section 4.1.4.) Following this, the controller can wait for or ignore the command confirmation message.

Alternatively, the controller can only wait for the command confirmation message. In this case, however, the current stub positions have not been updated. Following this, therefore, Read Motor Positions function should be called.

Example:

Command message:

17: 2

Response example:

22: 61 9 21 7 0 0 119 0 (Motors Data)

19: 2 (Confirmation)

RS232

Command Code	msATunCmd =72=48h
Label	ATC
Parameters	S
Response	<ul style="list-style-type: none"> MDO with motors data, bits 4 and 5 of HST set to 1; <i>followed by</i> Command Execution Confirmation (Section 3.4.3)

After issuing the command, the controller should wait for and receive MDO which

- contains motors data (i.e. bit 4 of HST is **1**), *and*
- is response to motors query (i.e. bit 5 of HST is **1**).

(Details of HST and MDO structure see in Sections 4.3 and 4.2.1, respectively.) Following this, the controller can wait for or ignore the Command Execution Confirmation message.

Alternatively, the controller can only wait for the Command Execution Confirmation message. In this case, however, the current stub positions have not been updated. Following this, therefore, Read Motor Positions function should be called.

Example:

The command string for making one autotuning step is "**ATC S**". The complete commands is transmitted as the byte sequence

128, 28, 65, 84, 67, 32, 83, 13, 10, 128, 72

Response:

128, 028, 48, 61, 9, 21, 7, 0, 0, 119, 0, 9, 128, 016 (MDO)

128, 28, 72, 0, 128, 4 (Confirmation)

7.5 Switching Sending Tune Positions ON and OFF

Description: This command sets and clears the **SendTunePos** flag (TRUE or FALSE). Setting of the flag affects the system behavior when *continuous* measurement is running. When **SendTunePos** = TRUE *and* **Autotune** = FALSE, Tune Positions are computed and sent after each measurement (details see in Section 5.1).

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msNoSendTune =91=5Bh to switch SendTunePos OFF msYesSendTune =90=5Ah to switch SendTunePos ON

Homer Response:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msNoSendTune =91=5Bh if SendTunePos = FALSE msYesSendTune =90=5Ah if SendTunePos = TRUE

Example:

Command message to switch **SendTunePos** ON:

17: 90 (to send Tune Positions)
17: 91 (not to send Tune Positions)

Response, respectively:

19: 90
19: 91

RS232

Command Code	msATunCmd =72=48h
Label	ATC
Parameters	F = SendTunePos OFF; T = SendTunePos ON
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To switch **SendTunePos** to FALSE and TRUE, the command string is "**ATC F**" and "**ATC T**", respectively. The complete commands are transmitted as the following byte sequences:

128, 28, 65, 84, 67, 32, 70, 13, 10, 128, 72 to switch **SendTunePos** OFF
128, 28, 65, 84, 67, 32, 84, 13, 10, 128, 72 to switch **SendTunePos** ON

Response:

128, 28, 72, 0, 128, 4

7.6 Stub Swap Alarm Parameters

Description: Sets parameters controlling

- generation in Homer of **Stub Swap Alarm** (SSA), and
- motors behavior when such alarm is raised.

For the background on the stubs swapping problem, consult User's Handbook or Help for Homer (HomerHelp.chm), topic [Autotuner > Hysteresis](#).

Stub Swap Alarm is evaluated anytime Tune Positions are computed (either for the purpose of actual tuning or only for Tune Position sending). When SSA is raised, Homer, *if* sending motors data, sets bit 4 of Motor Status MS2 to 1. That is how the user can learn about the alarm occurrence.

Whether motors data with alarm will be sent or not depends on Send Mask Register (Sections 10.3.1 and 11.4). To assure that alarm be sent anytime it is raised, set bit 3 of Send Mask Register to 1.

The SSA controlling parameters are MagLimit, InsLimit, Eval, and MotMove.

MagLimit

Defines the occurrence of *MagAlarm* condition (FALSE or TRUE). A *MagAlarm* condition occurs when load reflection coefficient magnitude (i.e. that “behind the tuner”) exceeds *MagLimit*. *MagAlarm* will not be raised for smaller reflection coefficients. *MagLimit* is defined in percent (0 to 100); 100% means total reflection. Note that *MagLimit* is a measure of field strength rather than power: the reflected-to-incident-power limit is equal to *MagLimit* squared.

InsLimit

Defines the occurrence of *InsAlarm* condition. An *InsAlarm* condition occurs when the maximum of the swapping stubs insertions before and after the movement exceeds *InsLimit* (see the example below). *InsAlarm* will not occur for shorter stub travels. *InsLimit* is defined as percentage of full stub travel; 100% means that at least one of the swapping stubs must go from zero to maximum or vice versa for the condition to occur.

Example:

Let the insertion in motor steps of the stubs 1 and 3 before and after swapping be

Before[1]=0, Before[3]=3000, After[1]=4000, After[3]=0.

The value for the evaluation of *InsAlarm* condition is

MaxIns = max {Before[1], Before[3], After[1], After[3]} = max {0, 3000, 4000, 0} = 4000.

Let the full stub insertion is 5000 motor steps. Then *InsAlarm* condition arises when

$\text{InsLimit} > \text{MaxIns}/5000 \times 100\% = 80\%$.

Eval

Defines how *MagAlarm* and *InsAlarm* conditions are combined to evaluate an aggregate *CombiAlarm* condition (see the following table).

Eval	Result	Note
0	CombiAlarm = FALSE	Alarm generation OFF
1	CombiAlarm = MagAlarm	Only <i>MagAlarm</i> considered
2	CombiAlarm = InsAlarm	Only <i>InsAlarm</i> considered
3	CombiAlarm = MagAlarm or InsAlarm	At least one of the conditions must arise
4	CombiAlarm = MagAlarm and InsAlarm	Both conditions must arise at the same time

Stub Swap Alarm SSA is raised when

- CombiAlarm = TRUE, **and**
- Swap Condition occurs.

Swap condition occurs when Stub 1 starts from zero to a nonzero position while Stub 3 goes from nonzero to zero, or vice versa. Mathematically, the Swap condition can be defined as follows:

Swap = ((Before[1] = 0) and (After[1] ≠ 0) and (Before[3] ≠ 0) and (After[3] = 0))
or ((Before[1] ≠ 0) and (After[1] = 0) and (Before[3] = 0) and (After[3] ≠ 0))

As already mentioned, when Stub Swap Alarm is raised, Homer sets bit 4 of Motor Status MS2 byte to 1. The external controller program can evaluate this bit when receiving Motors Data and decide what action to take.

MotMove

MotMove parameter may assume values 0 and 1. When set to 1, motors will ignore the SSA warning and always move to positions computed by the autotuning algorithm. When MotMove = 0, the motors will *not* move if SSA is raised.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC=17=11h	PC	6

Data bytes:

Byte	Value/Meaning
0	msSetStbSwap =92=5Ch
1	MagLimit (0 to 100)
2	InsLimit (0 to 100)
3	Eval (0 to 4)
4	MotMove (0 or 1)
5	(Irrelevant)

Homer Response:

ID	Sender	Byte Count
ciAtunR =19=13h	Homer	6

Data bytes: Same as above; actual values returned.

Byte	Value/Meaning
0	msSetStbSwap =92=5Ch

Example:

To set MagLimit = 60%, InsLimit = 70%, Eval = 3, MotMove = 0, the command message is:

17: 92 60 70 3 0 0

Response example:

19: 92 60 70 3 0 0

RS232

Command Code	msSetStbSwap =92=5Ch
Label	SSA
Parameters	MagLimit (0 to 100)
	InsLimit (0 to 100)
	Eval (0 to 4)
	MotMove (0 or 1)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set MagLimit = 60%, InsLimit = 70%, Eval = 3, MotMove = 0, the command string is "**SSA 60 70 3 0**"; the complete command is transmitted as the sequence

128, 28, 83, 83, 65, 32, 54, 48, 32, 55, 48, 32, 51, 32, 48, 13, 10, 128, 92

Response:

128, 28, 92, 0, 128, 4

8. SINGLE-SHOT COMMANDS

This section describes an assortment of commands which give user full control over the tuning process, including its timing. The commands trigger execution of one Measure-Tune-Send (MTS) sequence; hence the name single-shot commands. In case of MultiCAN, the single-shot commands enable synchronous triggering of more Homers on the same CAN bus.

The *Measure-Tune-Send Sequence* is a combination of one or more of the following actions (essentially the same as described in Section 5.1). Order of the individual steps may vary.

#	Action
1	Measurement
2	Sending Homer Measurement Results
3	Sending Actual Positions AP
4	Computing Tune Positions TP
5	Moving stubs to Tune Positions
6	Sending Tune Positions

Outcome of *Measurement* action is Homer Measurement Results HMR. Input data for the Tune Positions computation are Homer Measurement Results and Actual Positions of the motors.

In case of CAN bus, HMR is represented by a triplet of consecutive CAN messages. AP and TP are transmitted as a single message each.

In case of RS232, measurement results and motor positions can be combined to form one Measurement Data Object, MDO (Section 4.2). Consequently, these MDO types are possible: HMR, AP, TP, HMR+AP, HMP+TP.

The following table gives a brief overview of single-shot commands; their detailed description follows.

Command	Command Description	Response
Meas	Make one measurement and send its results.	HMR
FetchLast	Instructs Homer to send the latest measurement results. <u>Caution:</u> If Homer is in <i>Running</i> state, repeated call of this command gives different results.	HMR
CompStubs	Using the latest measurement results and Actual Positions, compute and send Tune Positions. Do not move motors. <u>Caution:</u> if between the measurement and calling this command the motors were moved to different positions, the computed values are in error.	TP
MeaComp	Make a measurement and send results including Actual Positions, then execute CompStubs procedure.	HMR, AP, TP
MeaTun	Make one measurement. If successful, compute Tune Positions and move motors to these positions. Send measurement results including Actual Positions (which are now equal to Tune Positions).	HMR, AP
MeaTunMea	Make one measurement. If successful, compute Tune Positions and move motors to these positions. Make a new measurement. Send measurement results including Actual Positions.	HMR, AP
ClrFifo	Clear Homer's buffer (FIFO) of received messages. The command assures immediate response to a subsequent command.	none

ClrFifo in fact does not fall into the class of single-shot commands but is closely associated with them. On the other hand, some commands falling into this class are described elsewhere because of the context. These include for instance Set Motor Positions (Section 6.3), Read Motor Positions (Section 6.4), and Single Autotuning Step (Section 0).

Notes:

- Correct use of single-shot commands requires that Homer be idle, i.e. in *not Running* state. To achieve it, use **Stop Measurement** command.
- Homer Measurement Results are sent always in case of commands that should respond by sending HMR, irrespective of whether the measurement could be performed or not or whether an error occurred during the measurement process. Check bit 6 of Homer Error Byte HER to see if the data are valid (the bit set to 1 means invalid measurement data).
- In case of Homer Analyzer alone (without Tuner), only operations not involving motors are performed in the course of commands execution.

8.1 Meas Command

Description: Make one measurement and send its results including motor positions. If bit 6 of Homer Error Byte HER is found to be 1, the measurement data, including the remaining HER bits, are invalid (measurement could not be performed at all or an error preventing its completion occurred during the process).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeas =85=55h

Homer Response:

Homer Measurement Results and Motors data, i.e. the following four consecutive CAN messages:

ID	Sender	Byte Count	Note
ciHmRes1 =11=0Bh	Homer	8	
ciHmRes2 =12=0Ch	Homer	8	
ciHmRes3 =13=0Dh	Homer	8	
ciMotSQ =22=16h	Homer	8	AP

Data bytes: the meaning see in Section 4.1.

Bit 7 of Motor Status 2 MS2 is set to 0 which means that the **ciMotSQ** message contains Actual Positions AP.

Example:

The command:

16: 85

The response example (HST=4, HER=0, Pinc=23.42 mW, Temp=25.4 C, Measured Re=0.05225, Im=0.3105, F=2454.11 MHz, Load (deembedded) Re=0.21753, Im=-0.02905, motors 1, 2, 3 at 0, 513, and 4000 steps from zero position, respectively):

11: 52 0 9 38 5 254 0 255
12: 214 0 248 4 184 172 160 14
13: 123 3 137 255 139 0 0 0
22: 0 0 1 2 160 15 119 0

RS232

Command Code	msMeas =85=55h
Label	none
Parameters	none
Response	MDO with Homer Measurement Results and Motors Data

Example:

The command is transmitted as the byte sequence

128, 85

The response example:

128, 28, 52, 0, 9, 38, 5, 254, 0, 255, 214, 0, 248, 4, 184, 172, 160, 14, 123, 3, 137, 255, 0, 0, 1, 2, 169, 15, 119, 0, 129, 128, 16

MDO starts by **msDataBegin** command (128, 28), followed by Homer Status Byte HST (52), Homer Error Byte (0) and the rest of data. MDO is terminated by **msMeasObject** = 16 = 10h command (128, 16) preceded by checksum byte CS (129).

8.2 FetchLast Command

Description: Instructs Homer to send the latest measurement results and motors data, without measuring. See also description of Meas command. **Caution:** If Homer is in *Running* state, repeated calls of this command may provide different results.

The command can be used to query measurement results in case when Homer is internally measuring (*Running* = TRUE) but sending no data (*Sending* = FALSE).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msFetchLast =39=27h

Homer Response:

Homer Measurement Results and Motors data, i.e. the following four consecutive CAN messages:

ID	Sender	Byte Count	Note
ciHmRes1 =11=0Bh	Homer	8	
ciHmRes2 =12=0Ch	Homer	8	
ciHmRes3 =13=0Dh	Homer	8	
ciMotSQ =22=16h	Homer	8	AP

Data bytes: the meaning see in Section 4.1.

Bit 7 of Motor Status 2 MS2 is set to 0 which means that the **ciMotSQ** message contains Actual Positions AP.

Example:

The command:

16: 39

The response example: As in Meas command.

RS232

Command Code	msFetchLast =39=27h
Label	none
Parameters	none
Response	MDO with Homer Measurement Results and Motors Data

Example:

The command is transmitted as the byte sequence

128, 39

The response example: As in Meas command.

8.3 CompStubs Command

Description: Using the latest measurement results and Actual Positions of the motors, compute and send Tune Positions. Do not move motors. **Caution:** If between the measurement and calling this command the motors were moved to different positions, the computed values are in error.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msCompStubs =86=56h

Homer Response:

ID	Sender	Byte Count	Note
ciMotSQ =22=16h	Homer	8	TP

Data bytes: the meaning see in Section 4.1. Bit 7 of Motor Status 2 MS2 set to 1 indicates that the message contains TP not the actual stub positions.

Example:

The command:

17: 86

The response example (motor 1, 2, 3 Tune Positions 2583, 1571, 0 motor steps from zero position, respectively, MS1=119, MS2=128):

22: 23 10 35 6 0 0 119 128

RS232

Command Code	msCompStubs =86=56h
Label	none
Parameters	none
Response	MDO with Tune Positions TP, without measurement results

Note: Bit 7 of Motor Status 2 MS2 set to 1 indicates that the message contains TP not the actual stub positions.

Example:

The command is transmitted as the byte sequence

128, 86

The response example (Tune Positions 2583, 1571, 0, MS1=119, MS2=128 (note the doubling of this byte), checksum CS=110:

128, 28, 48, 23, 10, 35, 6, 0, 0, 119, 128, 128, 113, 128, 16

Note that the doubled byte is added to checksum CS only as one byte (128 not 256).

8.4 MeaComp Command

Description: Make a measurement and send results including Actual Positions of the motors. Using the measurement results and Actual Positions, compute and send Tune Positions. Do not move motors.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC=17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeaComp=87=57h

Homer Response:

ID	Sender	Byte Count	Note
ciHmRes1=11=0Bh	Homer	8	
ciHmRes2=12=0Ch	Homer	8	
ciHmRes3=13=0Dh	Homer	8	
ciMotSQ=22=16h	Homer	8	AP
ciMotSQ=22=16h	Homer	8	TP

Data bytes: the meaning see in Section 4.1.

Bit 7 of Motor Status 2 MS2 set to 0 means that the message contains Actual Positions AP. The bit set to 1 indicates that the message contains Tune Positions TP.

Example:

The command:

17: 87

The response example:

11: 4 0 9 43 5 254 0 255
12: 223 0 241 4 98 216 159 14
13: 125 3 150 255 10 0 0 0
22: 172 13 132 8 0 0 119 0
22: 23 10 35 8 0 0 119 128

The first of the two ID=22 messages contains Actual Positions AP, which is indicated by Byte7=MS2=0. The second ID=22 message contains Tune Positions TP (MS2=128).

RS232

Command Code	msMeaComp=87=57h
Label	none
Parameters	none
Response	MDO with Homer Measurement Results and Actual Positions AP followed by MDO with Tune Positions TP

Note: Bit 7 of Motor Status 2 MS2 set to 1 indicates that the message contains TP not the actual stub positions.

Example:

The command is transmitted as the byte sequence

128, 87

The response example:

```
128, 28, 52, 0, 9, 43, 5, 254, 0, 255, 223, 0, 241, 4, 98, 216, 159, 14,
125, 3, 150, 255, 172, 13, 132, 8, 0, 0, 119, 0, 246, 128, 16,
128, 28, 48, 23, 10, 35, 8, 0, 0, 119, 128, 128, 115, 128, 16
```

The response consists of two MDOs, started by **msDataBegin** command (128, 28), followed by Homer Status Byte HST (52 and 48, respectively) and the rest of data. HST=52 means bits 2, 4 and 5 set to 1, indicating HMR present, Motors Data present, and motor query confirmation, respectively. HST=48 means only bits 4 (Motors Data) and 5 (query) set to 1. MDOs are terminated by **msMeasObject** = 16 = 10h command (128, 16) preceded by checksum byte CS (246 and 115, respectively). MS2 = 0 in the first MDO means AP, MS2 = 128 in the second MDO means TP. Note the doubling of MS2 = 128 in the second MDO (the bytes preceding CS). The doubled byte contributes only once to CS (128 rather than 256).

8.5 MeaTun Command

Description: Make one measurement. If successful, compute Tune Positions and move motors to these positions. Send measurement results including Actual Positions of the motors (which are now equal to Tune Positions).

Note that the motors data represent final stub settings whereas the measurement results correspond to starting stub positions.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC =17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeaTun =88=58h

Homer Response:

ID	Sender	Byte Count	Note
ciHmRes1 =11=0Bh	Homer	8	
ciHmRes2 =12=0Ch	Homer	8	
ciHmRes3 =13=0Dh	Homer	8	
ciMotSQ =22=16h	Homer	8	AP

Data bytes: the meaning see in Section 4.1.

Example:

The command:

```
17: 88
```

The response example:

```
11: 4 0 9 43 5 254 0 255
12: 223 0 241 4 98 216 159 14
13: 125 3 150 255 10 0 0 0
22: 23 10 35 8 0 0 119 0
```

RS232

Command Code	msMeaTun =88=58h
Label	none
Parameters	none
Response	MDO with Homer Measurement Results and Actual Positions AP.

Example:

The command is transmitted as the byte sequence

128, 88

The response example:

128, 28, 52, 0, 9, 43, 5, 254, 0, 255, 223, 0, 241, 4, 98, 216, 159, 14,
125, 3, 150, 255, 23, 10, 35, 8, 0, 0, 119, 0, 253, 128, 16

8.6 MeaTunMea Command

Description: Make one measurement. If successful, compute Tune Positions and move motors to these positions. Make another measurement. Send measurement results of the *second* measurement including Actual Positions of the motors (which are now equal to the Tune Positions).

Note that unlike MeaTun command both the motors data and the measurement results represent the final stub positions.

CAN

Command Message:

ID	Sender	Byte Count
ciAtunC=17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeaTunMea=89=59h

Homer Response:

ID	Sender	Byte Count	Note
ciHmRes1=11=0Bh	Homer	8	
ciHmRes2=12=0Ch	Homer	8	
ciHmRes3=13=0Dh	Homer	8	
ciMotSQ=22=16h	Homer	8	AP

Data bytes: the meaning see in Section 4.1.

Example:

The command:

17: 89

The response example:

11: 4 0 9 43 5 254 0 255
12: 5 0 51 0 98 216 159 14
13: 81 3 202 255 10 0 0 0
22: 23 10 35 8 0 0 119 0

RS232

Command Code	msMeaTunMea=89=59h
Label	none
Parameters	none
Response	MDO with Homer Measurement Results after tuning and Actual Positions AP.

Example:

The command is transmitted as the byte sequence

128, 89

The response example:

128, 28, 52, 0, 9, 43, 5, 254, 0, 255, 5, 0, 51, 0, 98, 216, 159, 14, 81, 3,
202, 255, 23, 10, 35, 8, 0, 0, 119, 0, 105, 128, 16

8.7 ClrFifo Command

Description: Clear Homer's buffer (FIFO) of received messages (CAN) or bytes (RS232). The command assures Homer's immediate response to a subsequent command.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msClrFifo =84=54h

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msClrFifo =84=54h
1	Error code (0 in case of success)

Example:

Command:

16: 84

Response:

18: 84, 0

RS232

Command Code	msClrFifo =84=54h
Label	none
Parameters	none
Response	Command Execution Confirmation (Section 3.4.3)

Example:

The command is transmitted as the byte sequence

128, 84

Response (success):

128, 28, 84, 0, 128, 4

8.8 Get Timeouts

Execution time of a command given to Homer can vary widely from tens of milliseconds to tens of seconds, depending for instance on:

- Homer model (e.g. its motors speed and maximum tuning stub extension)
- Homer settings (e.g. sampling rate and number of samples to take for one measurement point)
- Load reflection coefficient (dictating e.g. whether the stubs are to be moved or not)
- Whether Homer is in *Running* state or idle

After issuing a command, the controller (e.g. your PC) should wait for the response (set its timeout) in accordance with these circumstances. It is difficult for the controller to compute the timeout needed, and it is not wise to set it at a potential maximum because in case of communication problems this could lead to intolerable or annoying delays.

To help set appropriate timeouts, the **Get Timeouts** function has been implemented. The command instructs Homer to compute for the current settings and report back a suggested *measurement* timeout and *motors movement* timeout. The command returns two numbers:

- **MeasTimeout** (minimum time in milliseconds to wait for completing Homer measurement), and
- **MotorsTimeout** (minimum time in milliseconds to wait for completing full stub travel).

8.8.1 Using MeasTimeout and MotorsTimeout

Actual timeout (*Tout*) the controller should employ with a given command depends on a particular situation: whether Homer continuous measurement is running (*Running* flag), whether autotuning is activated (*Autotune* flag), whether the command is a *motors command*, i.e. such that can result in tuning stubs movement. Motors commands include Stepping Motors commands and single-shot MeaTun and MeaTunMea commands.

Some common situations and appropriate timeouts are summarized in the table below, where T_{MEA} and T_{MOT} stand for *MeasTimeout* and *MotorsTimeout*, respectively.

Running	Autotune	Motors Command	Tmout (ms)	Note
NO	-	NO	1000	Small value, independent of T_{MEA} and T_{MOT}
NO	-	YES	$1000 + T_{MOT}$	
YES	NO	NO	T_{MEA}	
YES	NO	YES	$T_{MEA} + T_{MOT}$	
YES	YES	NO	$T_{MEA} + T_{MOT}$	
YES	YES	YES	$T_{MEA} + 2 \times T_{MOT}$	

In the last case for instance, since a measurement may have started just before the command was issued, we must first wait until the measurement finishes (T_{MEA}). Then, because autotuning is ON, we should wait for completing potential motors movement (T_{MOT}). Only then Homer checks for the incoming command and processes it. Since this is a motors command, we must wait another T_{MOT} to complete the stubs movement.

8.8.2 Stopping Homer

It is good practice to stop Homer before issuing commands, in particular if the sampling mode is Rectified or Pulsed and a series of commands is to be issued. With Homer stopped, shorter timeouts can be employed in the subsequent commands.

To stop Homer at all circumstances, the timeout should be

$$Tout = MeasTimeout + MotorsTimeout$$

or repeated attempts should be made with proportionally lower *Tout* value.

8.8.3 Get Timeouts Command Structure

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msGetTmouts =61=3Dh

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	5

Data bytes:

Byte	Value/Meaning
0	msGetTmouts =61=3Dh
1	Homer Timeout in milliseconds – LSB
2	Homer Timeout in milliseconds – MSB
3	Motors Timeout in milliseconds – LSB
4	Motors Timeout in milliseconds – MSB

Example:

Command:

16: 61

Response example (Homer Timeout = 1000 ms, Motors Timeout = 3700 ms):

18: 61 232 3 116 14

RS232

Command Code	msGetTmouts =61=3Dh
Label	none
Parameters	none
Response	4-byte Data Object (Section 3.3.2)

Data bytes:

Byte	Value/Meaning
0	Homer Timeout in milliseconds – LSB
1	Homer Timeout in milliseconds – MSB
2	Motors Timeout in milliseconds – LSB
3	Motors Timeout in milliseconds – MSB

Example:

The complete command is transmitted as the sequence

128, 61

Response example (Homer Timeout = 1000 ms, Motors Timeout = 3700 ms):

128, 28, 232, 3, 116, 14, 128, 61

MDO starts by **msDataBegin** command (128, 28), followed by the four data bytes. MDO is terminated by **msGetTmouts**=61=3Dh command (128, 61).

8.9 Ping Message

Ping command requests Homer to respond by sending a **Pong** message. **Ping** message includes a sender-defined *Ping Byte B* which **Pong** message returns. In this way synchronization between commands and responses can be tested.

Tip: When sending repeatedly, you may wish to increment the Ping Byte before each command in the succession.

CAN

Using **Ping/Pong** messages, PC establishes CAN bus communication with Homer.

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	2

Data bytes:

Byte	Value/Meaning
0	msPingPong =20=14h
1	Ping Byte <i>B</i> (arbitrarily defined by sender)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msPingPong =20=14h
1	Ping Byte <i>B</i> (received Ping Byte value return)

Example:

Ping message with *B* = 235

16: 20 235

The response (**Pong** message) is

18: 20 235

RS232

Ping message includes a sender-defined *Ping Byte B* in form of ASCII string *S*. If *S* does not represent a byte, **Pong** returns 255.

Command Code	msPingPong =20=14h
Label	PNG
Parameters	ASCII string <i>S</i> representing Ping Byte <i>B</i>
Response	1-byte Data Object (Section 3.3.2) with End Message = msPingPong

Data byte:

Byte	Value/Meaning
0	Returned Ping Byte <i>B</i>

Example: For **Ping** message with *B* = 210, the command string is "**PNG 210**"; the complete command is transmitted as the sequence (CC stands for Command Code)

128, 28, 80, 78, 71, 32, 50, 49, 48, 13, 10, 128, 20
P N G 2 1 0 CC

The response contains the data byte *B* and CC as End Message:

128, 28, 210, 128, 20

9. HOMER RESET COMMANDS

Homer reset commands bring Homer to a predefined power-up state.

9.1 Restart Server, Halt Server, Change Server

Description:

Restart Server command terminates the execution of Homer internal control program (SrvHo.exe), operating in either RS232 or CAN bus or DeviceNet communication mode, and starts it again *in the same* communication mode. This type of reset closely emulates cycling Homer DC power off and on.

Halt Server command terminates the execution of SrvHo.exe and starts RS232 file transfer program (SrvDld.exe).

Change Server command terminates the execution of SrvHo.exe and, depending on the command bytes of this command:

- Starts it again with the desired communication mode and protocol (RS232, CAN Bus, DeviceNet)
- Restarts itself with *the same* communication mode and protocol (same effect as **Restart Server**)
- Starts the RS232 file transfer program SrvDld.exe (same effect as **Halt Server**)

As seen, **Change Server** can be used also in place of the **Restart Server** and **Halt Server** commands.

The response depends on the Server version: see below. To handle all possible Server versions, you may choose not to wait for the response after issuing a command.

CAN

Restart Server

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSrvRst =80=50h

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msSrvRst =80=50h
1	99=63h

Note:

For older Server versions (up to and including V53 of 22-Sep-2010), only the first of the two bytes is transmitted.

Example:

Command message:

16: 80

Response:

18: 80 99 (Server V54 and more)

18: 80 (Server V53 and less)

Halt Server

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSrvHalt =34=22h

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msSrvHalt =34=22h

Example:

Command message:

16: 34

Response:

18: 34

Change Server

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	2

Data bytes:

Byte	Value/Meaning
0	msSrvRst =80=50h
1	94 : Start DeviceNet server 95 : Exit to operating system 96 : Start RS232 file transfer program SrvDld.exe 97 : Start server in RS232 mode 98 : Start server in CAN bus mode 99 : Restart currently running server

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msSrvRst =80=50h
1	Copy of Command Byte 1

Note:

For older Server versions (up to and including V53 of 22-Sep-2010), only the first of the two bytes is transmitted.

Example:

Command message:

16: 80 97 (Restart Server in RS232 mode)

Response:

18: 80 97 (Server V54 and more)

18: 80 (Server V53 and less)

RS232

Restart Server, Halt Server

Command Code	msSrvRst =80=50h (Restart Server) msSrvHalt =34=22h (Halt Server)
Label	none
Parameters	none
Response	<ul style="list-style-type: none">• Server V53 and less: None• Server V54 and more: Command Execution Confirmation (Section 3.4.3)

Example: To restart or halt server program, the byte sequences to be sent are

128, 80 (Restart Server)

128, 34 (Halt Server)

Change Server

Command Code	msSrvRst =80=50h
Label	RSS
Parameters	94 : Start DeviceNet server 95 : Exit to operating system 96 : Start RS232 file transfer program SrvDld.exe 97 : Restart server in RS232 mode 98 : Restart server in CAN bus mode 99 : Restart currently running server
Response	<ul style="list-style-type: none">• Server V53 and less: None• Server V54 and more: Command Execution Confirmation (Section 3.4.3)

Example: To restart server in RS232 mode, the command string is "**RSS 97**"; the complete command is transmitted as the sequence

128, 28, 82, 83, 83, 32, 57, 55, 13, 10, 128, 80

9.2 Factory Reset, User-Defined Reset

Description:

Factory Reset: The command returns Homer completely to the state as it had left factory (out-of-box default) by way of copying firmware files to Homer internal working directory from a specific *Factory Default* location. This helps resolve unwanted confusing situations, e.g. after accidental uploading of improper firmware files. The factory default (contents of the *Factory Default* location) can be changed only by the manufacturer.

User-Defined Reset: Similar to the Factory Reset, this reset type returns Homer to a state previously defined by user by means of **Make User-Defined Reset** command, described in the next section. The corresponding firmware files are stored in a specific *User Default* location.

After performing this type of resets, Homer DC power ought to be cycled.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	7

Data bytes:

Byte	Value/Meaning
0	msReset =8=08h
1	1 (Factory Reset) 100 (User-Defined Reset)
2	1 (irrelevant)
3	255
4	255
5	255
6	127

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes:

Byte	Value/Meaning
0	msReset =8=08h
1	Failed File List – Byte 0 (LSB)
2	Failed File List – Byte 1
3	Failed File List – Byte 2
4	Failed File List – Byte 3 (MSB)
5	Reset Result

Failed File List identifies bitwise the files that could not be transferred (depending on the Homer options, not all possible files need to be present in *Factory Default* or *User Default* locations). A bit set to 1 corresponds to a file that was not available as follows:

Bit	File
0	HOM.CFG
1	TUN.CFG
2	START.CFG
3	SRVHO.CFG
4	RS232.CFG
5	CAN.CFG
6	BBOX.CFG
7	DAQ.CFG
8	SRVHO.EXE
9	SRVDNET.EXE
10	LOADER.EXE
11	SRVDLD.EXE
12	START.EXE
13	HOM.MEM
14	TUN.MEM
15	CIP.MEM
16	AUTOEXEC.BAT
17	CONFIG.SYS

For instance, if SRVDNET.EXE and CIP.MEM files are missing (not needed without DeviceNet option), the *Failed File List* = 33280 = 001000001000000000 binary.

Reset Result has the following meaning (should be zero in case of Factory reset):

Value	Meaning
0	Reset OK; Server has been restarted after the file transfers
1	Reset OK; Server has <i>not</i> been restarted after the file transfers
2	Reset fail

Example:

Command message:

16: 8 1 1 255 255 255 127 (in case of Factory Reset)

16: 8 100 1 255 255 255 127 (in case of User-Defined Reset)

Response example for *Failed File List* = 33280:

18: 8 0 130 0 0 0

RS232

Command Code	msReset =8=08h
Label	RST
Parameters	1 (Factory Reset)
	100 (User-Defined Reset)
	1 (irrelevant)
	2147483647
Response	Array of 5 bytes B[n] , n = 0...4

Meaning of bytes **B[n]**:

n	Meaning
0	Failed File List – Byte 0 (LSB)
1	Failed File List – Byte 1
2	Failed File List – Byte 2
3	Failed File List – Byte 3 (MSB)
4	Reset Result

Meaning of *Failed File List* and *Reset Result*: see CAN Bus above.

Example: The command string is

RST 1 1 2147483647 (in case of Factory Reset)
RST 100 1 2147483647 (in case of User-Defined Reset)

In the latter case, the complete command is transmitted as the sequence

128, 28, 82, 83, 84, 32, 49, 48, 48, 32, 49, 32, 50, 49, 52, 55, 52, 56, 51,
54, 52, 55, 128, 8

Response example for *Failed File List* = 33280 (only bytes **B[n]** and *Reset Result*):

0, 130, 0, 0, 0

9.3 Create User-Defined Reset

Description:

The command stores all relevant firmware files from Homer internal working directory (defining current power-up behavior) to a specific *User Default* location, making it source for the **User-Defined Reset**. The reset is useful e.g. after performing firmware upgrades, buying new options, or adapting Homer behavior (editing Hom.cfg, Tun.cfg) to your application. Factory Reset would eradicate all such changes.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	7

Data bytes:

Byte	Value/Meaning
0	msReset =8=08h
1	101 (Make User-Defined Reset)
2	1 (irrelevant)
3	255
4	255
5	255
6	127

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes:

Byte	Value/Meaning
0	msReset =8=08h
1	Failed File List – Byte 0 (LSB)
2	Failed File List – Byte 1
3	Failed File List – Byte 2
4	Failed File List – Byte 3 (MSB)
5	Reset Result

See the previous Section 9.2 for the meaning of *Failed File List* and *Reset Result*.

Example:

Command message:

16: 8 101 1 255 255 255 127

Response example for *Failed File List* = 33280:

18: 8 0 130 0 0 0

RS232

Command Code	msReset=8=08h
Label	RST
Parameters	101 (Make User-Defined Reset)
	1 (irrelevant)
	2147483647
Response	Array of 5 bytes B[n], n = 0...4

Meaning of bytes B[n]:

n	Meaning
0	Failed File List – Byte 0 (LSB)
1	Failed File List – Byte 1
2	Failed File List – Byte 2
3	Failed File List – Byte 3 (MSB)
4	Reset Result

See the previous Section 9.2 for the meaning of *Failed File List* and *Reset Result*.

Example: The command string is “RST 101 1 2147483647”

The complete command is transmitted as the sequence

128, 28, 82, 83, 84, 32, 49, 48, 49, 32, 49, 32, 50, 49, 52, 55, 52, 56, 51,
54, 52, 55, 128, 8

Response example for *Failed File List* = 33280 (only bytes B[n] and *Reset Result*):

0, 130, 0, 0, 0

10. CW MEASUREMENT SETUP

Measurement results, like reflection coefficient and incident power, are obtained from the measurement of the following quantities:

- Measurement of the four six-port reflectometer (SPR) detector voltages when microwave signal is applied to them (referred to as Signal Measurement).
- Measurement of the detector voltages when microwave signal is not applied to them (referred to as Offset Measurement). Offsets are mainly due to post-detection DC amplification circuitry. They are slowly varying quantities, dependent primarily on temperature.
- Measurement of signal frequency.
- Measurement of internal temperature. This again is a slowly varying quantity.

Not all of these quantities (specifically offsets and temperature) need to be measured with the same rate as signal. This may increase the overall continuous measurement throughput. Further methods to increase speed include:

- Lowering of voltage measurement averaging (number of samples taken to obtain one voltage reading).
- Increasing of sampling rate (Section 5.6).
- Decreasing of frequency counting time.
- (less effective) Using fixed A/D converter ranges rather than autoranging.

This section describes commands giving the user more control over details of CW measurement process, like timing of particular measurements or setting of A/D converter ranges. Section 11 gives details of command messages.

10.1 Measurement Timing

This section describes timing of particular operations in one Measurement action. The settings do not apply to single-shot commands (Section 8); these are executed anytime a single-shot command is issued. Timing of data transmitting is also covered here.

10.1.1 Signal Measurement Period

Description: The command governs continuous measurement rate in terms of minimal interval (**OnPeriod**) in milliseconds between two consecutive signal measurements. If OnPeriod is nonzero, the program loop checks whether at least OnPeriod time has elapsed since the last measurement, and only in such case a new measurement is triggered. If OnPeriod is set to zero, no such check is performed and measurement runs at maximum possible rate. The power-up value is governed by the following line in Hom.cfg file:

```
OnPeriod_ms=0
```

If this line is missing or incorrect, the default is also zero.

10.1.2 Offset Measurement Period

Description: The command sets **OfsPeriod**, which is a minimum interval in seconds between two consecutive DC offsets measurements. If OfsPeriod is nonzero, the program loop checks whether at least OfsPeriod time has elapsed since the last offset measurement, and only in such case a new offset measurement is launched. If OfsPeriod is set to zero, offset measurement takes place anytime signal is being measured.

Since DC offsets are slow time-varying functions, they need not be measured too often. The power-up value is governed by the following line in Hom.cfg file:

```
OfsPeriod_s=30
```

If the line is missing or incorrect, the default is also 30 seconds.

10.1.3 Frequency Measurement Period

Description: The command sets **FPeriod**, which is a minimum interval in milliseconds between two consecutive frequency measurements. If FPeriod is nonzero, the program loop checks whether at least

FPeriod time has elapsed since the last frequency measurement, and only in such case a new frequency measurement is performed. If FPeriod is set to zero, frequency measurement takes place anytime signal is being measured.

The power-up value is governed by the following line in Hom.cfg file:

```
FPeriod_ms=0
```

If this line is missing or incorrect, the default is also zero. It is sensible to let FPeriod_ms=0. A better way of increasing measurement speed is to decrease count time down to as low as 32 μ s (Section 5.4) and/or signal measurement averaging down to as low as 1 (Section 5.3).

10.1.4 Temperature Measurement Period

Description: The command sets **TPeriod**, which is a minimum interval in seconds between two consecutive internal temperature measurements. If TPeriod is nonzero, the program loop checks whether at least TPeriod time has elapsed since the last temperature measurement, and only in such case a new temperature measurement is performed. If TPeriod is set to zero, temperature measurement takes place anytime signal is being measured.

Since temperature is a slow time-varying function, it needs not be measured too often. The power-up value is governed by the following line in Hom.cfg file:

```
TPeriod_s=30
```

If this line is missing or incorrect, the default is also 30 seconds.

10.2 A/D Converter Range

10.2.1 Signal Measurement Range

Description: The command sets **OnRnge**, which is the A/D converter range (in fact programmable-gain preamplifier setting) used for signal measurement. The possible values and corresponding gains are given in the table below:

Range Code	PGA Gain	ADC Range	Note
-1	Automatic	Automatic	Recommended; default
0	1000	5 mV	Do not use
1	100	50 mV	Do not use
2	10	500 mV	
3	1	5 V	

Automatic setting finds the appropriate *individual* range for each of the four measurement channels while fixed ranges are common for all channels. The power-up value is governed by the following line in Hom.cfg file:

```
OnRnge=-1
```

If the line is missing or incorrect, the default is also -1.

10.2.2 Offset Measurement Range

Description: The command sets **OfsRnge**, which is the A/D converter range (in fact programmable-gain preamplifier setting) used for offset measurement.

Important note: The OfsRnge setting is only applicable when offset ranges are not forced to be equal to signal ranges (see next).

The possible values and corresponding gains are given in the table above. The power-up value is governed by the following line in Hom.cfg file:

```
OfsRnge=2
```

If the line is missing or incorrect, the default is also 2.

10.2.3 Offset Ranges Equal to Signal Ranges

Description: The command sets **OfsEqualOn** boolean flag, which, if TRUE, forces the offset ranges to be the same as those used in the signal measurement. (The settings may be different in different channels if Autoranging was employed in signal measurement.) In this case OfsRnge setting is ignored. If OfsEqualOn = FALSE then OfsRnge setting is used.

The power-up value is governed by the following line in Hom.cfg file:

```
OfsEqualOn=TRUE
```

If the line is missing or incorrect, the default is also TRUE.

10.3 Data Transmitting

This section describes timing and conditions upon which Homer Measurement Results (HMR) and Motors Data (MD) are sent.

Note: The data can only be sent if Homer is both in *Running* and *Sending* state, which case is implicitly assumed in this section. See Section 11.1 on how to set Homer's *Running* and *Sending* states.

10.3.1 Send Event Register and Send Mask Register

During the course of one Measure-Tune-Send (MTS) sequence, various events occur, affecting whether data transmission will take place or not. The particular events control bits in *Send Event Register* (SER). Before each intended transmission, SER is compared against *Send Mask Register* (SMR). SMR enables only selected events, the rest are ignored. If the result of (SER **and** SMR) bitwise operation is nonzero, the data transmission will take place.

The individual bits of SER and SMR have the following meaning:

Bit	Meaning
0	Homer Measurement Results available
1	HMR sending period expired
2	Motors Refresh period expired
3	Stub Swap Alarm (SSA) raised
4	(not used)
5	(not used)
6	(not used)
7	(not used)

Bit 0 of SER is set to 1 after successful measurement. If bit 0 of SMR mask is set to 1 as well, HMR are sent anytime new HMR are available.

Bit 1 of SER is set to 1 when at least **TxPeriod** time (see below) has elapsed since the last transmission **and** valid HMR are available.

Bit 2 of SER is set to 1 when at least **MotRefresh** time (see below) has elapsed since the last transmission. If this event occurred and is enabled by SMR, only Motors Data will be sent.

Bit 3 of SER is set to 1 when Stub Swap Alarm is raised. Enabling only this bit in SMR means that nothing but alarms will be sent. See Section 7.6 about generating SSA.

10.3.2 Sending Period

Description: The command sets **TxPeriod**, which is a minimum interval in milliseconds between two consecutive transmissions of HMR (provided other conditions for the transmission are satisfied). If TxPeriod is nonzero, the program loop checks whether at least TxPeriod time has elapsed since the last HMR transmission, in which case bit 1 of SER is set to 1 otherwise to 0. If TxPeriod is zero, the bit is always set to 1. The power-up value is governed by the following line in Hom.cfg file:

```
TxPeriod_ms=0
```

If the line is missing or incorrect, the default is also zero.

10.3.3 Motors Refresh Period

Description: The command sets **MotPeriod**, which is a minimum interval in milliseconds between two consecutive transmissions of Motors Data. If MotPeriod is nonzero, the program loop checks whether at least MotPeriod time has elapsed since the last MD or HMR transmission, in which case bit 2 of SER is set to 1 otherwise to 0. If MotPeriod is set to zero, the bit is always set to 1. It is, however, unwise to set MotPeriod to zero since the communication bus will be flooded with motors data. The power-up value is governed by the following line in Hom.cfg file:

```
MOT_REFRESH=100
```


If the line is missing or incorrect, the default is also 100 ms.

11. MEASUREMENT SETUP COMMANDS

11.1 Set/Get Running and Sending States

Description: Sets or queries *Running* and *Sending* Homer states (TRUE or FALSE).

When *Running* = TRUE, Homer performs measurement and possibly autotuning continuously or at a rate defined by parameters described later in this section. When *Running* = FALSE, Homer is idle and only waits for and executes commands.

Sending state is only relevant when *Running* = TRUE. When *Sending* = FALSE, Homer sends neither measurement results nor motors data (although it may make measurement and autotuning). When *Sending* = TRUE, Homer Measurement Results and Motors Data are sent based on the states of the Send Event and Send Mask registers.

Power-up values are governed by the following line in Hom.cfg file:

```
AUTORUN=3
```

Bit 0 of the number on the right-hand side sets *Running*, bit 1 sets *Sending*. If the line is missing or incorrect, the default is also 3 (i.e. both flags TRUE).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	3

Data bytes:

Byte	Value/Meaning
0	msSweepStart =17=11h
1	Running (0 = FALSE; 1 = TRUE; 2 = Query – do not change)
2	Sending (0 = FALSE; 1 = TRUE; 2 = Query – do not change)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	3

Data bytes: Same as above; actual values returned.

Example 1:

Set *Running* = TRUE, *Sending* = FALSE. The command message is:

```
16: 17 1 0
```

Response:

```
18: 17 1 0
```

Example 2:

Set *Running* = FALSE, do not change *Sending*. The command message is:

```
16: 17 0 2
```

Response:

```
18: 17 0 1
```

Byte 3 (boldface) returned the actual *Sending* state (TRUE).

Example 3:

To query *Running* and, *Sending* states, the command message is:

```
16: 17 2 2
```

Response example:

18: 17 1 1

(both flags found to be TRUE).

RS232

Command Code	msSweepStart=17=11h
Label	SRS
Parameters	Running (0 = FALSE; 1 = TRUE; 2 = Query – do not change)
	Sending (0 = FALSE; 1 = TRUE; 2 = Query – do not change)
Response	Query (SRS 2 2): 2-byte Data Object (Section 3.3.2) with actual values
	Otherwise: Command Execution Confirmation (Section 3.4.3)

Example 1:

To set *Running* = TRUE, *Sending* = FALSE, the command string is "SRS 1 0"; the complete command is transmitted as the sequence

128, 28, 83, 82, 83, 32, 49, 32, 48, 13, 10, 128, 17

Response:

128, 28, 17, 0, 128, 4

Example 2:

To set *Sending* = FALSE without changing *Running*, the command string is e.g. "SRS 2 0"; the complete command is transmitted as the sequence

128, 28, 83, 82, 83, 32, 50, 32, 48, 13, 10, 128, 17

Response:

128, 28, 17, 0, 128, 4

Example 3 - Query:

To query *Running* and, *Sending* states, the command string is e.g. "SRS 2 2"; the complete command is transmitted as the sequence

128, 28, 83, 82, 83, 32, 50, 32, 50, 13, 10, 128, 17

Response example:

128, 28, 1, 1, 128, 17

(both flags found to be TRUE).

11.2 Set Signal and Offset Measurement Periods

Description: Sets **OnPeriod** and **OfsPeriod** for signal and offset measurement, respectively.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC=16=10h	PC	6

Data bytes:

Byte	Value/Meaning
0	msHmSetOther=94=5Eh
1	0 (fixed specifier)
2	OnPeriod in milliseconds – LSB
3	OnPeriod in milliseconds – MSB
4	OfsPeriod in seconds – LSB
5	OfsPeriod in seconds – MSB

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes: Same as above; actual values returned.

Example:

To set OnPeriod = 500 ms and OfsPeriod = 60 s, the command message is:

16: 94 0 244 1 60 0

Response:

18: 94 0 244 1 60 0

RS232

Command Code	msHmSetOther =94=5Eh
Label	HSO
Parameters	0 (fixed specifier)
	OnPeriod in milliseconds
	OfsPeriod in seconds
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set OnPeriod = 500 ms and OfsPeriod = 60 s, the command string is "**HSO 0 500 60**"; the complete command is transmitted as the sequence

128, 28, 72, 83, 79, 32, 48, 32, 53, 48, 48, 32, 54, 48, 13, 10, 128, 94

Response:

128, 28, 94, 0, 128, 4

11.3 Set Frequency and Temperature Measurement Periods

Description: Sets **FPeriod** and **TPeriod** for frequency and temperature measurement, respectively.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	6

Data bytes:

Byte	Value/Meaning
0	msHmSetOther =94=5Eh
1	1 (fixed specifier)
2	FPeriod in milliseconds – LSB
3	FPeriod in milliseconds – MSB
4	TPeriod in seconds – LSB
5	TPeriod in seconds – MSB

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes: Same as above; actual values returned.

Example:

To set **FPeriod** = 500 ms and **TPeriod** = 60 s, the command message is:

16: 94 1 244 1 60 0

Response:

18: 94 1 244 1 60 0

RS232

Command Code	msHmSetOther=94=5Eh
Label	HSO
Parameters	1 (fixed specifier)
	FPeriod in milliseconds
	TPeriod in seconds
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set FPeriod = 500 ms and TPeriod = 60 s, the command string is "HSO 1 500 60"; the complete command is transmitted as the sequence

128, 28, 72, 83, 79, 32, 49, 32, 53, 48, 48, 32, 54, 48, 13, 10, 128, 94

Response:

128, 28, 94, 0, 128, 4

11.4 Set Sending Period and Send Mask Register

Description: Sets TxPeriod and Send Mask Register SMR

CAN

Command Message:

ID	Sender	Byte Count
ciHomC=16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	msHmSetOther=94=5Eh
1	2 (fixed specifier)
2	TxPeriod in milliseconds – LSB
3	TxPeriod in milliseconds – MSB
4	Send Mask Register

Homer Response:

ID	Sender	Byte Count
ciHomR=18=12h	Homer	5

Data bytes: Same as above; actual values returned.

Example:

To set TxPeriod = 500 ms and SMR = 6, the command message is:

16: 94 2 244 1 6

Response:

18: 94 2 244 1 6

RS232

Command Code	msHmSetOther=94=5Eh
Label	HSO
Parameters	2 (fixed specifier)

	TxPeriod in milliseconds
	Send Mask Register
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set TxPeriod = 500 ms and SMR = 6, the command string is "**HSO 2 500 6**"; the complete command is transmitted as the sequence

128, 28, 72, 83, 79, 32, 50, 32, 53, 48, 48, 32, 54, 13, 10, 128, 94

Response:

128, 28, 94, 0, 128, 4

11.5 Measurement Ranges

Description: Sets signal measurement range (**OnRnge**), offset measurement range (**OfsRnge**), and **OfsEqualOn** flag.

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	6

Data bytes:

Byte	Value/Meaning
0	msHmSetOther =94=5Eh
1	3 (fixed specifier)
2	OnRnge (see Section 10.2.1)
3	OfsRnge (see Section 10.2.2)
4	OfsEqualOn (0 = FALSE; 1 = TRUE)
5	2 (voltage measurement type, see Note below)

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	6

Data bytes: Same as above; actual values returned.

Note: Byte 5 is used for servicing purposes; it controls what type of voltages are measured and used (0 = only offsets, 1 = only signals, 2 = signals minus offsets). The byte is compulsory. Set byte 5 always to 2.

Example:

To set OnRnge = -1 (Auto), OfsRnge = 2 (500 mV) and OfsEqualOn = TRUE (hence OfsRnge has no relevance), the command message is:

16: 94 3 255 2 1 2

Response:

18: 94 3 255 2 1 2

Note that the byte representation of -1 is 255 (see Section 1.2.1).

RS232

Command Code	msHmSetOther =94=5Eh
Label	HSO
Parameters	3 (fixed specifier)
	OnRnge (see Section 10.2.1)
	OfsRnge (see Section 10.2.2)

	OfsEqualOn (F = FALSE; T = TRUE)
	2 (voltage measurement type; optional, see Notes below)
Response	Command Execution Confirmation (Section 3.4.3)

Notes:

1. As TRUE in OfsEqualOn setting, any normal text string starting with **y, Y, t, T, 1**, can be used, e.g. **yes, TRUE**, etc. As NO, strings starting with **n, N, f, F, 0**, can be used, e.g. **NO, false**, etc. All other cases will not affect the value of OfsEqualOn.
2. The last parameter (voltage measurement type) is optional and is used for servicing purposes. It controls what type of voltages are measured and used (0 = only offsets, 1 = only signals, 2 = signals minus offsets). Either omit this parameter or set it to 2.

Example:

To set OnRnge = -1 (Auto), OfsRnge = 2 (500 mV) and OfsEqualOn = TRUE (hence OfsRnge has no relevance) the command string is "**HSO 3 -1 2 T**"; the complete command is transmitted as the sequence

128, 28, 72, 83, 79, 32, 51, 32, 45, 49, 32, 50, 32, 84, 13, 10, 128, 94

Response:

128, 28, 94, 0, 128, 4

11.6 Set Motors Refresh Period

Description: Sets **MotPeriod** in the range from 0 to 32767 ms or queries.

To set: The argument (MotPeriod) must be in the interval $0 \leq \text{MotPeriod} \leq 32767 = 7FFFh$.

To query: The argument must be outside of the above interval (e.g. MotPeriod = 32768 = 8000h).

CAN

Command Message:

ID	Sender	Byte Count
ciHomC =16=10h	PC	3

Data bytes:

Byte	Value/Meaning
0	msMotRefresh =76=4Ch
1	MotPeriod in milliseconds – LSB
2	MotPeriod in milliseconds – MSB

Homer Response:

ID	Sender	Byte Count
ciHomR =18=12h	Homer	3

Data bytes: Same as above; actual value returned.

Example: (suppose the actual MotPeriod = 5000 ms)

To query MotPeriod, the command message may be:

16: 76 0 128

Response:

18: 76 136 19

To set now MotPeriod = 500 ms, the command message is:

16: 76 244 1

Response:

18: 76 244 1

The same message would now be received as a response to a new query.

RS232

Command Code	<code>msMotRefresh=76=4Ch</code>
Label	<code>xxx</code>
Parameters	MotPeriod in milliseconds – LSB
Response	2-byte Data Object (Section 3.3.2) with actual value; LSB first

Example: (suppose the actual MotPeriod = 5000 ms)

To query MotPeriod, the command string may be "`xxx 32768`"; the complete command is transmitted as the sequence

`128, 28, 88, 88, 88, 32, 51, 50, 55, 54, 56, 13, 10, 128, 76`

To set MotPeriod = 500 ms, the command string is "`xxx 500`"; the complete command is transmitted as the sequence

`128, 28, 88, 88, 88, 32, 53, 48, 48, 13, 10, 128, 76`

Response:

`128, 28, 244, 1, 128, 76`

The same message would now be received as a response to a new query.

12. USING ADVANCED COMMANDS

This section contains a few practical comments and examples how to use the great variety of available commands and options in tasks concerning measurement in CW mode.

12.1 Normal Situation

For normal continuous measurement, leave all to defaults: modify neither the supplied Hom.cfg nor Tun.cfg files. We start with this situation, where signal and frequency measurements are performed in each cycle of the main program loop, data are sent after each measurement, and temperature and offsets are measured each 30 seconds.

12.2 Increasing Measurement Rate

To increase measurement rate, use these steps (with decreasing preference):

- Decrease frequency-counting time (value as low as 100 μ s is still better than switching the counter off).
- Decrease signal measurement averaging while leaving sampling frequency at its maximum.
- Use fixed signal measurement A/D converter range rather than autoranging (brings little merit).

If you want the internal measurement to be running fast but you need data less frequently, you have two choices:

1. Switch *Sending* state to FALSE and query for HMR and MD whenever you wish.
2. Leave *Sending* = TRUE, and
 - Set Sending Period to a nonzero value.
 - Set Motors Refresh Period to a nonzero value.
 - Modify Send Mask Register SMR: Clear bit 0 and set bits 1 and 2, so that SMR = 110 bin = 6.

12.3 Stub Swap Alarm

If you want Stub Swap Alarm (SSA) generated, set up conditions for its occurrence properly whereby **Eval** must differ from zero (Section 7.6).

To detect SSA occurrence, test bit 4 of Motor Status Part 2 (MS2) of the incoming Motors Data.

If you want the stubs to move in spite of the alarm raised, set **MotMove** of the alarm setup to 1. In this case you will only be notified about the alarm occurrence but cannot do much about it.

If you want that the stubs stay motionless when the alarm is raised, set **MotMove** of the alarm setup to 0. In this case you can take an appropriate action on alarm notification.

To be sure that motors data containing alarm are always sent, switch *Sending* state to TRUE and set bit 3 of SMR mask to 1.

12.4 Single-Shot Commands

If you wish to have full command over measurement/tuning process, switch *Running* state to FALSE and use single-shot commands. A possible succession of actions is as follows:

- **ClrFifo** command.
- **Meas** command.
- Verify validity of received results (check bit 6 of Homer Error Byte HER, which must be zero for valid results).
- **Read Motor Positions** command.
- **CompStubs** command.

Based on the difference between Actual Positions (AP) and Tune Positions (TP), you can decide whether stub swapping is imminent or a stub travel is excessively large, etc., and take an appropriate action. **The decision is yours because single-shot commands involving motor movements ignore Stub Swap Alarm.**

You can for instance:

- Set Motor Positions directly to the computed TP (with one command).
- Bring motors to TP with smaller steps by more Set Motor Positions commands.
- Set Motor Positions to other positions than TP.
- Do not move motors at all.