HOMER Hot Measurement and 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
· · ·		The dated revisions between 36.1 and 59.1 are not reported anymore.
59.1	14-Dec-22	– Command “Motors Initialization” (Sec. 6.2) for case of CAN Bus modified. Old version still applicable. – Command “Selected Stub Home” removed. – Description of bytes SRL, SRH, RE (Secs. 4.1.3, 4.2.1) reformulated. – Added Sec. 4.2.3 (MDO in case of CW sampling). – New command “Read Maximal Motors Step Count and Step Size” (Sec. 6.3). – Modification of Autotune setting and querying commands (Sec. 7.3). – Stub Swap Alarm (SSA) functionality removed.
59.2	02-Nov-23	– Improvement of Sec. 4.3. – Removed commands and constants related to sending Tune Positions, i.e., commands *Switching Sending Tune Positions ON and OFF*, *CompStubs*, and *MeaComp*.
59.3	20-Aug-24	– Removed error in Example in Sec. Switch Waveform, part RS232
60.1	06-Jul-26	– Added Sec. 12.1 (Format of RS232 Results). Added Changing Server to EtherNet/IP (Sec. 9.1). Modified Sec. 7.1.

The revision version number corresponds to the Homer firmware (Server) version number for which the protocol is best applicable. For instance, the communication protocol with revision number 59.3 is best applicable to Server version 59.

Information in this document is subject to change without notice.
https://s-team.sk

Table of Contents

1. INTRODUCTION 5

1.1 TERMS AND ABBREVIATIONS 5
1.2 INTEGER VALUE TYPES 6
1.2.1 Typecasting 6
1.3 USED CONSTANTS 7
1.4 MISCELLANEOUS 8

2. CAN SPECIFIC ISSUES 9

2.1 CAN IDENTIFIERS 9
2.2 PREPARING THE SYSTEM 9
2.3 RECEIVING MESSAGES 9
2.4 SENDING COMMANDS 10
2.4.1 Example: Set Autotuning ON and OFF 10
2.5 MULTICAN: MORE HOMERS ON CAN BUS 11
2.6 BROADCAST COMMANDS 11

3. RS232 SPECIFIC ISSUES 13

3.1 SETTING UP COM PORT 13
3.2 TRANSMITTING 13
3.2.1 Byte Types 13
3.2.2 Sending Commands 13
3.2.3 Sending Data Bytes 13
3.2.4 Data Objects 13
3.3 RECEIVING 14
3.3.1 Receiving and Decoding Bytes 14
3.3.2 Receiving Data Objects 14
3.4 ISSUING COMMANDS 14
3.4.1 Command Parameters 15
3.4.2 Waiting for Response from Homer 15
3.4.3 Command Execution Confirmation 15

4. MEASUREMENT RESULTS 17

4.1 CASE OF CAN BUS 18
4.1.1 Homer Measurement Results, Part 1 18
4.1.2 Homer Measurement Results, Part 2 18
4.1.3 Homer Measurement Results, Part 3 18
4.1.4 Motors Data 19
4.2 CASE OF RS232 20
4.2.1 MDO Data Bytes 20
4.2.2 Checksum Byte 21
4.2.3 Case of CW Sampling Mode 22
4.3 HOMER STATUS BYTE (HST) 23
4.4 HOMER ERROR BYTE (HER) 23
4.5 MOTOR STATUS, PART 1 (MS1) 24
4.6 MOTOR STATUS, PART 2 (MS2) 24
4.7 DECODING DATA BYTES 24
4.7.1 Some Derived Quantities 25
4.7.2 Phase of Reflection Coefficient 25

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. STEPPER MOTORS 37

6.1 MOTORS SELECTION MAP 37
6.2 MOTORS INITIALIZATION 37
6.3 READ MAXIMAL MOTORS STEP COUNT AND STEP SIZE 39
6.4 SET MOTOR POSITIONS 40
6.5 READ MOTOR POSITIONS 41
6.6 HARD STOP OF MOTORS 42

7. AUTOTUNING 43

7.1 AUTOTUNING CONTROL PARAMETERS 43
7.1.1 Server Versions Starting V60 43
7.1.2 Server Versions V55 to V59 44
7.2 HYSTERESIS 45
7.3 CONTINUOUS AUTOTUNING 46
7.4 SINGLE AUTOTUNING STEP 49

8. SINGLE-SHOT COMMANDS 51

8.1 MEAS COMMAND 52
8.2 FETCHLAST COMMAND 53
8.3 MEATUN COMMAND 54
8.4 MEATUNMEA COMMAND 55
8.5 CLRFIFO COMMAND 56
8.6 GET TIMEOUTS 56
8.6.1 Using MeasTimeout and MotorsTimeout 58
8.6.2 Stopping Homer 58
8.7 PING MESSAGE 59

9. HOMER RESET COMMANDS 61

9.1 RESTART SERVER, HALT SERVER, CHANGE SERVER 61
9.2 FACTORY RESET, USER-DEFINED RESET 64
9.3 CREATE USER-DEFINED RESET 66

10. MEASUREMENT SETUP PARAMETERS 68

10.1 MEASUREMENT TIMING 68
10.1.1 Signal Measurement Rate 68
10.1.2 Offset Measurement Rate 68
10.1.3 Frequency Measurement Period 68
10.1.4 Temperature Measurement Period 69
10.2 A/D CONVERTER RANGE 69
10.2.1 Signal Measurement Range 69
10.2.2 Offset Measurement Range 69
10.2.3 Offset Ranges Equal to Signal Ranges 69
10.3 DATA TRANSMITTING 70
10.3.1 Send Event Register and Send Mask Register 70
10.3.2 Sending Period 70
10.3.3 Motors Refresh Period 70
10.3.4 Measurement Rate vs. Results Transmitting Cadence 71

11. MEASUREMENT SETUP COMMANDS 71

11.1 SET/GET RUNNING AND SENDING STATES 71
11.2 SET SIGNAL AND OFFSET MEASUREMENT PERIODS 72
11.3 SET FREQUENCY AND TEMPERATURE MEASUREMENT PERIODS 73
11.4 SET SENDING PERIOD AND SEND MASK REGISTER 74
11.5 MEASUREMENT RANGES 75
11.6 SET MOTORS REFRESH PERIOD 76

12. MISCELLANEOUS COMMANDS 78

12.1 SET/QUERY FORMAT OF RS232 RESULTS 78

1. INTRODUCTION

This handbook provides guidance for communication with Homer devices via RS232 or CAN (Controller Area Network) serial interfaces.

1.1 Terms and Abbreviations

Homer[1] Analyzer: A system (or part of a system, in the case of an Autotuner) that performs measurements and provides for communication.

Homer Mototuner: A system (or part of a system, in the case of an Autotuner) that executes tuning stub movements. Also denoted simply Tuner.

Homer Autotuner: A system combining a Homer Analyzer with a Mototuner to perform automatic impedance matching (autotuning). The entire Autotuner intelligence and communication capability resides within the Homer Analyzer part.

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

MultiCAN: A scenario in which multiple Homers are connected to the CAN bus. For details, refer to Homer Help or Homer Handbook, topic *Advanced Topics > MultiCAN or More Homers on CAN Bus*.

Measure-Tune-Send Sequence (MTSS): A mnemonic description of the type and succession of possible actions taken by Homer in one measurement cycle (measurement, autotuning, sending data). The specific structure and order of constituent actions depends on the system state and the particular command that invokes an MTSS.

Reference position: The motor position or tuning stub insertion depth after executing the motor *initialization routine*. The reference position is a position corresponding to zero setting (zero steps) in motor movement commands (i.e., zero-millimeter nominal stub insertion depth).

Initialization routine: A procedure that withdraws each of the tuning stubs until its top terminal switch is activated. To ensure high repeatability, the motor is then stepped to a position which is then designated the reference position for that stub. The routine is alternatively denoted as **All Stubs Home**, **Reset Stubs**, **All Motors Home** or simply **All Home** procedure.

Abbreviations:

AP	Actual Positions (of motors, i.e., of tuning stubs)
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, i.e., of tuning stubs)[2]
MS1	Motor Status, Part 1
MS2	Motor Status, Part 2
MSB	Most significant bit/byte
MSM	Motors Selection Map
MTSS	Measure-Tune-Send Sequence
PC	Personal computer (or another controller)
SER	Send Event Register
SMR	Send Mask Register

[1] The designation Homer is derived from **Ho**t **Me**asu**r**ement system.

[2] Tune Positions are values computed to achieve an impedance match. The input data for the computation are Homer Measurement Results and Actual Positions.

Notes:

- In the bit representation of a byte, bit 0 is LSB (weight 1), bit 7 is MSB (weight 128).
- Bit *n* of a byte B is designated B.*n*.
- If not explicitly stated otherwise, numbers without suffix, e.g., 57, are assumed to be decimal.
- Numbers with prefix **0x** are hexadecimal, e.g., 57 (decimal) = 0x39 (hexadecimal).

1.2 Integer Value Types

Integer value types used here are:

Type	Other Name(s)	Description	# of Bytes	Min value	Max value
UINT8	Byte	Unsigned 8-bit number	1	0	255
UINT16	Word	Unsigned 16-bit number	2	0	65535
INT16	Integer	Signed 16-bit number	2	-32768	32767
INT32	Longint	Signed 32-bit number	4	-2147483648	2147483647

1.2.1 Typecasting

Programming languages offer typecasting from unsigned to signed variable types and vice versa. To indicate typecasting of a UINT16 W to an INT16 I, we shall use the notation I = int16(W).
In principle, if MSB of W is zero, then $I = W$. If MSB of W is one, then

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

Mathematically

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

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

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

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

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

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 types of any size.

1.3 Used Constants

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 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	0D	Carriage Return
msQuery	14	0E	General query via ASCII strings
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
msRsltForm	35	23	Set/Query Format of RS232 Results
msFetchLast	39	27	Instruct Homer to send latest Measurement Results
msHmWform	53	35	Switch signal waveform, i.e., mode of sampling (CW, Rectified, Pulsed)
msHmPcor	55	37	Set/Query power measurement calibration factor
msHmCounter	56	38	Set frequency counter
msHmAvrg	57	39	Set Homer averaging numbers
msGetTmouts	61	3D	Get Homer measurement timeout and motors movement timeout
msMotMaxSteps AndStepSize	62	3E	Get maximal motors step count and step size
msMotInit	69	45	Initialize all motors
~~msOneHome~~	70	46	Initialize one motor (deprecated)
msMotSet	71	47	Set motor positions
msATunCmd	72	48	Only RS232: Autotune commands
msATunPar	73	49	• RS232: Set or read parameters governing the autotuning behavior • CAN: Read autotuning parameters
msMotRead	74	4A	Read motor positions
msSetFsmplCw	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
msMeaTun	88	58	Measure and make one autotuning step
msMeaTunMea	89	59	Measure, make one autotuning step, measure again
msHmSetOther	94	5E	Set various Homer parameters (e.g., timing, ADC range)
msTuSetOther	96	60	Set various Tuner parameters (e.g., Hysteresis)
CmndLbl	128	80	Command Label; precedes command code byte

A note for the programmer: These values are *unlikely* to be changed in future. Despite this, we advise you to use symbols in your code and assign the symbols constant values in a single place of your program.

1.4 Miscellaneous

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

The Homer unit behavior at its power-up is controlled by the unit's internal **Hom.cfg**, **Tun.cfg, Rs232.cfg** and **Can.cfg** files. For details and methods of modifying them, refer to Homer Help or Homer Handbook, topics:

Advanced Topics > Homer Analyzer Starting Parameters

Advanced Topics > Homer Autotuner Starting Parameters

Homer Start > Configuring Communication

As a factory default, after power-up the system starts continuous measurement without autotuning and without sending measurement results. If you command Homer to send messages, you can intercept them and use the data they transmit. In the case of CAN, the data are sent in the form of CAN messages. In the case of RS232, the data are sent in the form of byte streams called Measurement Data Objects (MDO). An example Delphi/Pascal code for receiving MDOs via RS232 is attached as **HmRs232.pas** file.

To start continuous autotuning manually, set the **Autotune** switch to ON position (if it is ON after power-up, switch it OFF, then ON again). To start continuous autotuning programmatically, send the message described in Section 7.3. To control autotuning parameters, see the commands in Section 7.1.

Please be aware that the system is being continually upgraded, which may lead to some future changes in this 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 SPECIFIC ISSUES

The protocol applied: CAN Specification 2.0, Part B, with 11-bit CAN identifiers. Because the highest 7 bits cannot all be recessed (= 1), this allows a maximal ID value of 2031 (= 0x07EF = 111 11101111 (binary)).

2.1 CAN Identifiers

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

Symbol	Dec	Hex	Sender	Purpose
ci09_BrCast	9	09	PC	ID used for Broadcast Commands (see Section 2.6)
ci0A_HmStop	10	0A	PC	Stop Homer internal measurement and tuning
			Homer	Response to Stop command
ci0B_HmRes1	11	0B	Homer	Measurement results, Part 1 of 3
ci0C_HmRes2	12	0C	Homer	Measurement results, Part 2 of 3
ci0D_HmRes3	13	0D	Homer	Measurement results, Part 3 of 3
ci0E_MotCmd	14	0E	PC	Motor commands
ci0F_MotStat	15	0F	Homer	Motors status (positions and errors)
ci10_HomCmd	16	10	PC	Homer commands (i.e., those regarding measurement)
ci11_AtCmd	17	11	PC	Commands controlling the autotuning process
ci12_HomResp	18	12	Homer	Response to Homer commands (ci10_HomCmd)
ci13_AtResp	19	13	Homer	Response to autotuning commands (ci11_AtCmd)
	20	14		Reserved
	21	15		Reserved
ci16_MotQry	22	16	Homer	Motors status (positions and errors); equivalent to ci0F_MotStat
ci40_TransfC	64	40	PC/Homer	Used in transfer of data blocks
ci41_TransfR	65	41	PC/Homer	Used in transfer of data blocks
ci42_ObjTX	66	42	PC/Homer	Used in transfer of data blocks
ci43_ObjRX	67	43	PC/Homer	Used in transfer of data blocks

Notes for the programmer:

- These values are *unlikely* to be changed in future. Despite this, we advise you to use symbols in your code and assign the symbols constant values in a single place in your program.
- Process messages with ID = `ci0F_MotStat` and ID = `ci16_MotQry` in the same way.

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 (only waiting for commands), or starts measurements, possibly autotuning, and sending messages into the CAN bus network. An example of the AUTORUN line is shown below:

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

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

Notes:

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

2.3 Receiving Messages

In your PC, start a CAN Bus monitoring program. Set its baudrate equal to that of Homer. The factory default Homer baudrate is 500 kbits/s. The Homer baudrate can be modified by editing the Homer internal **Can.cfg** file (see Homer Help or Homer Handbook).

Receive CAN messages (frames) and observe them. The train of messages may typically 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). Their number can be anywhere between 0 and 7.

- 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 the message contains motor positions and status.

2.4 Sending Commands

To transmit a command to Homer:

- Set CAN message identifier ID.
- Set the data byte count.
- Prepare data bytes (B0 to B7). B0 is the 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 B0 of the returned response. If the command was successful, B0 is equal to the command code you transmitted. In case of command error, the returned B0 is the command code incremented by 128 (i.e., its MSB is set to 1). In addition, or as alternative to this, some commands return error code in one of the response data bytes.

2.4.1 Example: Set Autotuning ON and OFF

(Please ensure that the **Autotune** hardware switch on your Homer is in OFF position.)

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

To Set Autotuning ON:

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

```
17:    1
```

- Send the above message.
- Receive messages and wait for a message with ID = 19. The message returns two data bytes:

```
19:    1  1
```

  Byte B0 should be the same as the one sent (B0 = 1).
  - If this is the case, then byte B1 returns the actual autotuning state. In our example, B1 = 1. If B1 = 0, the autotuning still remained OFF.

– If the first returned byte is B0 = 1 + 128 = 129, the command has not been not executed successfully. Byte B1 is then meaningless.

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  0    (response in case of success)
```

2.5 MultiCAN: More Homers on CAN Bus

Theoretically, up to 20 Homers can be connected at the same time to a CAN network and controlled independently. For details, refer to Homer Help or Homer Handbook, topic *Advanced Topics > MultiCAN or More Homers on CAN Bus*.The following precautions 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 assigning multiple Homers ***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:

$N = ID$ div 100

$ID_0 = 1 + (ID \bmod 100)$

where *div* is integer division (e.g., 216 div 100 = 2) and *mod* returns the remainder (e.g., 216 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 = `ci10_HomCmd` = 16 = 10h. You therefore send a message with $ID = 16+100*(3-1) = 216$. The Homer responds with a message with $ID = 218$, hence the base ID is ID_0 = `ci12_HomResp` = 18 = 12h.

2.6 Broadcast Commands

Sometimes, you may wish to send a command to all Homers on the CAN bus instead of sending the command individually to each Homer 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 an 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 OrigCnt ≤ 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 `ci09_BrCast` = 9.
- Data bytes B0...B6 of the broadcast message are equal to OrigB0...OrigB6, i.e., B0=OrigB0, B1=OrigB1, ... B6=OrigB6. Unused bytes may have arbitrary value.

- The last data byte (B7) of the broadcast message is the ID of the original message: B7=OrigID.
- The 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 OrigCnt ≤ 7, you cannot use them, e.g., for the commands Set Motor Positions (Section 6.3) or Set Autotuning Control Parameters (Section 7.1) because these commands need all 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 = **ci09_BrCast** = 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

The following COM port settings are fixed:

- 8 data bits
- 1 stop bit
- No parity

Variable COM port settings are defined by **Rs232.cfg** file stored inside Homer. The only relevant quantity is the baudrate. The factory default is **115200 bits/s**.

3.2 Transmitting

3.2.1 Byte Types

The transmitted bytes fall into the following categories:

- **Command codes** (message codes). A command code is a byte that identifies the action that should be executed, or identifies the nature of the data that have just been transmitted. It may have any value from 0 to 127.
- **Data bytes**. A data byte is interpreted as a number or as a character.
- **Command Label**. There is one special byte called *Command Label Byte*; its value is `CmndLbl` = 128 = 0x80.

3.2.2 Sending Commands

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

```
CmndLbl, CommandCode
```

i.e.,

```
128, CommandCode
```

Example: To send Data Begin command (command code `msDataBegin` = 28 = 0x1C), the two bytes `CmndLbl, msDataBegin` are transmitted, i.e., the bytes `128, 28`.

3.2.3 Sending Data Bytes

Two cases must be distinguished: the case when a data byte (let us denote it `DataByte`) differs from `CmndLbl` = 128 = 0x80 and the case when a data byte is equal to `CmndLbl`.

- If `DataByte` differs from `CmndLbl`, it is simply transmitted as a single byte.
- If `DataByte` = `CmndLbl`, it is doubled, i.e., it is transmitted 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 general 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 blocks:

1. **Data Begin** message, which is a two-byte sequence: `CmndLbl` = 128 = 80h and `msDataBegin` = 28 = 0x1C, i.e., `128, 28`.

2. **Data** itself (note that each byte equal to `CmndLbl` must be doubled).
3. **End Message**, which is a two-byte sequence `CmndLbl, EndMsg`, where `EndMsg` is data identifier, indicating that (a) the data stream has ended, and (b) how it should be interpreted.

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

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

Data bytes are indicated here 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 `CmndLbl` = 128 = 0x80, this byte is interpreted as data byte equal to B1.
- When the received byte B1 = `CmndLbl` = 128 = 0x80, the byte following it (B2) must be received as well.
  - If the second byte B2 differs from `CmndLbl` = 128 = 0x80, this 2-byte incidence is interpreted as a reception of a command with command code equal to B2.
  - If this second byte B2 =B1 = `CmndLbl`, this 2-byte incidence is interpreted as the reception of a single *data* byte equal to `CmndLbl`.

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

- byte **`b`**: data byte or command code (depending on **`IsCommand`** flag).
- **`IsCommand`** : boolean flag. If TRUE, byte **`b`** is a command code; if FALSE, **`b`** is a data byte.
- (implicitly) **`Err00`**: code of communication error (**`Err00`** = 0 if no error has occurred 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 = 0x1C).
- 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 presently received data sequence.

See **`ReceiveDataObject`** function in HmRs232.pas file as an example implementation. The function input is **`wait_ms`**: the maximum wait time for receiving `msDataBegin` command. Once `msDataBegin` has arrived, the rest of the data object is received until an end message.

The function result is TRUE in case of success. The function also returns:

- **`Buffer`**: the buffer for received data bytes; it is implicitly typecast to a desired structure **`RxBuff`**.
- **`n`**: received byte count (valid data bytes are **`Buffer[0]`** to **`Buffer[n-1]`**).
- **`EndMsg`**: the 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 = 0x1C command.
2. Send the command parameters.
3. Send the `CommandCode` command.
4. Wait for a response from Homer.

Steps 1 and 2 are omitted if there are no parameters associated with the command (e.g., All Stubs Home command).
Step 4 is omitted if there is no response expected 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. This is in fact superfluous since it doubles the command code. However, it remains both for historical reasons, and because it 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 = 0x0D) + Line Feed (`LF` = 10 = 0x0A); it does not need to be separated from the rest. The terminator is mandatory for Server versions 59 and less, and optional for Server versions starting V60. In the examples that follow, the terminator will always be included.

The text representation of a number means its conversion from binary to string. As an example, the text representation of the number -1.37×10^{-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=0x48), 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 the 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,  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, while some do not return anything.
The returned data may contain 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 the end message code `msConfirm` = 4.

- The first data byte (B1) is the replica of the command code: B1 = `CommandCode`.
- The second data byte (B2) is, for most commands, an error byte. If the command has been executed successfully, then B2 = 0.
- There are a few exceptions, where B2 has a different meaning:
  - In the response to **Ping** command, B2 is equal to *Test Byte* sent by the **Ping** command (see Section 8.7).
  - In the response to **Autotune** commands, B2 represents the actual value of the parameter set or queried by the **Autotune** command, *or* an error code (see Section 7.3).

Example: Response to the above `ATC 0` and `ATC 1` commands (command code `msATunCmd`=72=48h) is, in case of success, respectively,

```
128,  28,  72,   0, 128,   4
128,  28,  72,   1, 128,   4
```

If Homer could not execute the commands successfully (if, for instance, the command string was **ATC 7**), the response could be, for instance,

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

where 3 is an 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 related 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 “de-embedded” reflection coefficient or reflection coefficient “behind the tuner”
- Homer Error Byte
- reflected power: only in case of Rectified and Pulsed sampling when averaged results are transmitted (i.e., not individual samples).

The 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 Homer Help or Homer Handbook for details regarding the reference planes.

Motors Data include

- actual positions,
- motors status.

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

4.1 Case of CAN Bus

In the case of CAN bus, **Homer Measurement Results** consist of three consecutive CAN messages, the first with ID = `ci0B_HmRes1` = 11 = 0x0B, the second with ID = `ci0C_HmRes2` = 12 = 0x0C, and the third with ID = `ci0D_HmRes3` = 13 = 0x0D.

4.1.1 Homer Measurement Results, Part 1

ID	Sender	Byte Count
`ci0B_HmRes1`=11=0x0B	Homer	8

Byte	Name	Meaning	Note
0	HST	Homer Status Byte (Section 4.2.3)	
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
`ci0C_HmRes2`=12=0x0C	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
`ci0D_HmRes3`=13=0x0D	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	

The sample serial number (SRL, SRH) is a valid quantity only if individual samples are transmitted in Rectified and Pulsed measurement modes. These modes are not covered in this document.

Notes:

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

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

- If both HST.0=0 ***and*** HST.1=0, then HST.6=0 when in CW sampling mode, and HST.6=1 when sending averaged results in Rectified or Pulsed sampling modes:
  - If HST.6=0 (CW sampling mode)
    - SRL, SRH not used (values irrelevant)
    - RE not used (value irrelevant)
  - If HST.6=1 (averaged results in Rectified or Pulsed sampling modes)
    - SRL Coded reflected power – LSB
    - SRH Coded reflected power – MSB
    - RE Coded reflected power – exponent
- If HST.0=1 ***or*** HST.1=1, then, regardless of HST.6:
    - SRL Sample serial number – LSB
    - SRH Sample serial number – MSB
    - RE not used (value irrelevant)

4.1.4 Motors Data

Motors Data occupy one CAN message with ID = `ci0F_MotStat` = 15 = 0x0F or `ci16_MotQry` = 22 = 0x16.

ID	Sender	Byte Count
`ci0F_MotStat`=15=0x0F	Homer	8
`ci16_MotQry`=22=0x16	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 = `ci16_MotQry` is sent *only* as a response to the motor query (Section 6.5) or single-shot commands (Section 8). In all other cases, the message with ID = `ci0F_MotStat` is sent.

4.2 Case of RS232

In the case of RS232, data are sent in the form of Data Objects with the end message **`msMeasObject`** = 16 = 0x10, 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 Nmax = 31, i.e., MDO[0] to MDO[30]. Note, however, that the count of the sent bytes (and thus the bytes which have to be received) may be greater than MDO length N (see Section 3.2.3).

One MDO may contain Homer Measurement Results, Motors Data, or both. The information about which components are present in an MDO and which are not is included in its first byte, MDO[0], referred to as *Homer Status Byte* (HST). Homer Status Byte is *always* present in an MDO; its structure is given below in Section 4.2.3.

4.2.1 MDO Data Bytes

For the specific case of CW sampling mode, see section Case of CW Sampling Mode.

Byte	Name	Meaning	Note
0	HST	Homer Status Byte (Section 4.2.3)	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 the frequency has been freshly measured; otherwise the value from the previous frequency measurement has been sent.

M Byte is present only if Motors Data are included (bit HST.4 set to 1).

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

- If both HST.0=0 ***and*** HST.1=0, then HST.6=0 when in CW sampling mode and HST.6=1 when sending averaged results in Rectified or Pulsed sampling modes:
  - If HST.6=0 (CW sampling mode)
    - SRL, SRH not present in MDO
    - RE not used (value irrelevant) but present if HMR are included in MDO
  - If HST.6=1 (averaged results in Rectified or Pulsed sampling modes)
    - SRL Coded reflected power – LSB
    - SRH Coded reflected power – MSB
    - RE Coded reflected power – exponent
- If HST.0=1 ***or*** HST.1=1, then, regardless of HST.6:
    - SRL Sample serial number – LSB
    - SRH Sample serial number – MSB
    - RE not used (value irrelevant) but present if HMR are included in MDO

4.2.2 Checksum Byte

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

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

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

4.2.3 Case of CW Sampling Mode

For the specific case of CW sampling mode (if HST.0, HST.1, and HST.6 are all zero):

- The MDO does not include bytes SRL and SRH, and
- byte RE is irrelevant.

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	Irrelevant	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	M1L	Motor 1 position – LSB	M
21	M1H	Motor 1 position – MSB	M
22	M2L	Motor 2 position – LSB	M
23	M2H	Motor 2 position – MSB	M
24	M3L	Motor 3 position – LSB	M
25	M3H	Motor 3 position – MSB	M
26	MS1	Motor Status, Part 1 (Section 4.5)	M
27	MS2	Motor Status, Part 2 (Section 4.6)	M
28	CS	Checksum byte	A

4.3 Homer Status Byte (HST)

Bits	Value	Meaning
0+1	00b=**0**	a) CW mode of sampling, or b) MDO carries *averaged* results in pulsed measurements (MDO is *not* a single sample)
	01b=**1**	MDO is the first sample in pulsed measurements
	10b=**2**	MDO is an intermediate sample in pulsed measurements
	11b=**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: a 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 a motor query or a single-shot command
6	**0**	a) if b0=0 and b1=0: Bytes SRL, SRH not present in MDO; byte RE irrelevant b) if b0=1 or b1=1: Bytes SRL, SRH used as sample S/No in pulsed measurements when individual samples are transferred; byte RE irrelevant
	1	Bytes RE, SRL, SRH used for reflected power transfer (only if b0=0 and b1=0)
7	-	Not used

Notes:

- *Pulsed measurements* above mean measurements using Rectified or Pulsed mode of sampling.
- 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 previous measurement is copied.

4.4 Homer Error Byte (HER)

Bit	Meaning
0	Measurement error in Rectified or Pulsed mode of sampling
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 a corresponding bit set to 1.
- A warning is not a fatal error: measurement accuracy may, however, be degraded.
- The 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 a terminal switch or was 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 (i.e., 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	-	not used
5	-	not used
6	-	not used
7	-	not used (starting Server version V59)

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 were hard-stopped (Section 6.6).

If MS2.0, MS2.1, and MS2.2 are all 0 (i.e., MS2 AND 111 (binary) = 0), then all motors are without error.

4.7 Decoding Data Bytes

A word of warning: Care must be taken with expressions like 256*B where B is a single byte (i.e., UINT8) and the result is expected to be a 2-byte variable (i.e., UINT16 or INT16). This may, depending on programming language, lead either to a run-time error or an incorrect result (e.g., 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*uint16(B). This similarly applies to UINT8 vs. INT32 and UINT16 vs. INT32.

Reflection Coefficient

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

- Create INT16 XS (i.e., with values between –32768 and 32767) from the bytes XL, XH, e.g., XS = int16(XL) + 256*int16(XH); see also Section 1.2.1.
- Similarly, create INT16 YS from the bytes YL, YH.
- Then
  X = XS/4096.0
  Y = YS/4096.0

Frequency

The frequency F in Hz is computed as follows:

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

Incident Power

The 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

The internal temperature T in degrees Celsius is computed as follows:

T = integer(TL + 256*TH)/10.0

4.7.1 Some Derived Quantities

Magnitude of reflection coefficient

$$M = \sqrt{X^2 + Y^2}$$

Return Loss

$$RL = -20 \log_{10}(M)$$

Voltage standing wave ratio

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

Reflected power in CW mode of sampling, or, generally, when bytes RE, SRL, SRH are ***not*** used for reflected power transfer (see note S in Section 4.2.1)

$$P_r = P_i M^2 = P_i \left(X^2 + Y^2\right)$$

Power absorbed in load

$$P_a = P_i - P_r$$

4.7.2 Phase of Reflection Coefficient

To obtain the correct phase φ of the reflection coefficient in all four quadrants, *do not* apply the standard one-dimensional *arctangent*, which is a function of the ratio Y/X of the imaginary (Y) and real (X) components of the reflection coefficient. In programming languages this *arctangent* function is usually denoted as `arctan(x)` (Pascal) or `atan(x)`(C).

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

$$\varphi_{rad} = \arctan 2(Y, X) \qquad \text{(radians)}$$

$$\varphi_{\text{deg}} = \frac{180}{\pi} \varphi_{rad} \qquad \text{(degrees)}$$

If your programming language supports only the standard one-dimensional *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 your programming language supports only the standard *arccosine* function, then use the following routine:

```
r := sqrt(X * X + Y * Y); { reflection coefficient magnitude }
if r < 1E-20 then
  phase := 0.0 { phase undefined }
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 that control behavior of the *measurement* portion of the Homer system (Homer Analyzer). Please refer to Section 4 for the measurement results structure.

5.1 Start Continuous Measurement

Description: Homer starts making measurements and associated actions (*Measure-Tune-Send Sequences,* or *MTSS*) repeatedly in that it sets its *Running* and *Sending* flags to TRUE. Please refer also to Set/Get Running and Sending States.

Measurement can be asynchronous in the sense that a MTSS is immediately followed by a next one. Another option is to start a new MTSS only after a defined time has elapsed after the start of the previous MTSS (see Section 10.1.1).

Duration of an MTSS depends on the system settings (such as measurement time, frequency counting interval) and other circumstances (e.g., whether the tuning stubs are moved, etc.).

The actions executed in one MTSS depend on **Autotune** flag setting (see Sections 7.3); they can include:

#	Action	Executed
1	Measurement	Always
2	Computing Tune Positions	if **Autotune** = TRUE
3	Moving stubs to Tune Positions	if **Autotune** = TRUE
4	Sending Homer Measurement Results	Always
5	Sending Actual Positions	Always

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
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSweepStart=17=11h

Homer Response:

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

ID	Sender	Byte Count
ci12_HomResp=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 meaning of the data bytes contained in the individual messages):

ID	Sender	Note
ci0B_HmRes1=11=0Bh	Homer	
ci0C_HmRes2=12=0Ch	Homer	
ci0D_HmRes3=13=0Dh	Homer	
ci0F_MotStat=15=0Fh	Homer	Actual Positions (AP)

Notes:

- The byte count of all messages is 8.
- If **Autotune =** TRUE, Homer Measurement Results correspond to stub positions before their shifting to new positions while the sent AP 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:

```
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...

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 = 20, 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:

```
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`).

5.2 Stop Measurement

Description: Homer stops continuous measurement and associated actions (such as 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
`ci0A_HmStop`=10=0Ah	PC	1

Data bytes:

Byte	Value/Meaning
0	`msSweepStop`=18=12h

Note that the command has a special ID (`ci0A_HmStop` = 10 = 0Ah rather than = 16 = 10h).

Homer Response:

Homer responds with a message identical with the command:

ID	Sender	Byte Count
`ci0A_HmStop`=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)

Example:

The command is transmitted as the byte sequence

```
128, 18
```

The response can be

```
128, 28, 18, 0, 128, 4
```

The first data byte (18) is the replica of the command code msSweepStop=18=12h. The second data byte (0) is the error code, in this case indicating no error.

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* applies only to CW mode of sampling: it means the sample count taken for one measurement result. (Setting of sampling frequency is described in Section 5.6.)

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10h	PC	5

Data bytes:

Byte	Value/Meaning
0	msHmAvrg=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
ci12_HomResp=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.

Example:

To set Vavrg = 256, Tavrg = 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 (Vavrg)
	Temperature measurement averaging (Tavrg)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set Vavrg = 256, Tavrg = 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 mode of signal sampling.
- Turns Homer frequency counter ON or OFF (for any mode of signal sampling).

Count time for CWsampling mode is expressed in microseconds as a 4-Byte signed integer (longint). The accepted count time range is 16 microseconds to 1 second.

Please note that the count time is quantized in steps of 16 µs. The actual, and hence returned ,value of count time will be the highest multiple of 16 µs that does not exceed the set value. Thus, for instance, if the set value is 1000 µs, the actual value will be 992 µs.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10 h	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
ci12_HomResp=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 µs 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 µs 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 that Homer will send in a case when:

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

CAN

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

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=18=12h	Homer	1

Data bytes:

Byte	Value/Meaning
0	msSetFsubst=7

Example:

To set Fsubst = 2450 MHz = 2450000 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	msSetFsubst=7
Label	FRE
Parameters	Substitute frequency in kHz (Fsubst)
Response	Command Execution Confirmation (Section 3.4.3)

Example:

To set Fsubst = 2450 MHz = 2450000 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 Fsampl 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 signed integer (longint). Homer responds with a one-data-byte message, returning the command byte.

Command Message:

ID	**Sender**	**Byte Count**
ci10_HomCmd=16=10 h	PC	5

Data bytes:

Byte	**Value/Meaning**
0	msSetFsmplCw=75=4Bh
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**
ci12_HomResp=18= 12h	Homer	1

Data bytes:

Byte	**Value/Meaning**
0	msSetFsmplCw=75=4Bh

Example:

To set Fsampl = 100 kHz = 100000 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	msSetFsmplCw=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 a measured frequency from the 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 signed integer (longint). Homer responds with a one-data-byte message, returning the command byte.

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=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 the mode of sampling of the signal waveform (CW, Rectified, Pulsed).

CAN

Command Message:

ID	**Sender**	**Byte Count**
ci10_HomCmd=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**
ci12_HomResp=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, 49, 13, 10, 128, 53
```

Response:

```
128, 28, 53, 0, 128, 4
```

6. STEPPER MOTORS

This section describes messages enabling to set the tuning stubs to defined positions (in terms of motor steps from 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) is a byte that 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 6.5).

CAN

Command Message:

ID	Sender	Byte Count
`ci0A_HmStop`=10=0Ah or ~~`ci0E_MotCmd`~~	PC	1 or 2 (see Notes)

Data bytes:

Byte	Value/Meaning
0	`msMotInit`=69=45h
1	Motors Selection Map MSM (see Notes)

Notes:

- Use messages with ID = `ci0A_HmStop` = 10 = 0Ah. The case ID = `ci0E_MotCmd` = 14 = 0Eh serves for backward compatibility only.
- New command version with Byte Count = 1 can be used and is encouraged for firmware (Server) versions starting V59.
- Old command version with Byte Count = 2 serves for backward compatibility. To initialize all motors (All Stubs Home, Reset Stubs) in Server versions V58 and less, set Byte 1 equal to `7`. Use this command alternative if you want to control both old and new Server versions.

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

ci0A_HmStop=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.

Server versions starting V59:

Command message

```
10:   69
```

Response (success)

```
10:   69  0
```

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

```
10:  197  1
```

Old Server versions (V58 and lower):

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: (this command is only possible with Server versions V58 and lower)

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

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

Command Code	msMotInit=69=45h
Label	none
Parameters	none
Response	• Server V52 and less: None • Server V53 and more: Command Execution Confirmation (Section 3.4.3)

Example:

To initialize all motors, send only the msMotInit command:

```
128, 69
```

The response for Server V53 and more may be

```
128, 28, 69, 0, 128, 4
```

The first data byte (69) is the replica of the command code. The second data byte (0) is error code, in this case indicating no error.

6.3 Read Maximal Motors Step Count and Step Size

Description: This command reads from the Homer device

– the maximal step count of the stepper motors (corresponding to the maximal tuning stub insertions), and
– the step size in the units of tens of nanometers (1E-8 meters or 1E-5 millimeters).

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMotMaxStepsAndStepSize=62=3Eh

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=18=12h	Homer	5

Data bytes:

Byte	Value/Meaning
0	msMotMaxStepsAndStepSize=62=3Eh
1	**Maximal Motors Step Count** – LSB
2	**Maximal Motors Step Count** – MSB
3	**Step Size** in tens of nanometers – LSB
4	**Step Size** in tens of nanometers – MSB

Example:

Command:

```
16:   62
```

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

```
18:   62 188  17 244   1
```

Then:

MaxSteps = 188 + 256*17 = 4540
StepSize_mm = (244 + 256*1)*1E-5 = 500*1E-5 = 0.005 mm
MaxInsertion_mm = MaxSteps*StepSize_mm = 4540*0.005 = 22.7 mm

RS232

Command Code	msMotMaxStepsAndStepSize =62=3Eh
Label	none
Parameters	none
Response	4-byte Data Object (Section 3.3.2)

Data bytes:

Byte	Value/Meaning
0	**Maximal Motors Step Count** – LSB
1	**Maximal Motors Step Count** – MSB
2	**Step Size** in tens of nanometers – LSB
3	**Step Size** in tens of nanometers – MSB

Example:

The complete command is transmitted as the sequence

```
128, 62
```

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

```
128, 28, 188, 17, 244, 1, 128, 62
```

MDO starts by `msDataBegin` command (`128, 28`), followed by the four data bytes. MDO is terminated by `msMotMaxStepsAndStepSize`=62=3Eh command (`128, 62`).

Then:

MaxSteps = 188 + 256*17 = 4540
StepSize_mm = (244 + 256*1)*1E-5 = 500*1E-5 = 0.005 mm
MaxInsertion_mm = MaxSteps*StepSize_mm = 4540*0.005 = 22.7 mm

6.4 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* (i.e., not an increment from the current 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
`ci0E_MotCmd`=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
`ci16_MotQry`=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 an 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 4.2.3 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
```

6.5 Read Motor Positions

Description: Queries for current motor positions and status.

CAN

Command Message:

ID	Sender	Byte Count
ci16_MotQry=22=16 h or ci0E_MotCmd	PC	1

Note: Using message with ID = ci16_MotQry is preferable in order not to possibly overwrite a previous message with ID = ci0E_MotCmd.

Data bytes:

Byte	Value
0	msMotRead=74=4Ah

Homer Response:

Motors Data – a message with ID = ci16_MotQry = 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 4.2.3 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
```

6.6 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 was lost. There is no response from Homer to this command.

CAN

Command Message:

ID	Sender	Byte Count
`ci0E_MotCmd`=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 the parameters controlling the autotuning process. For their meaning, refer to Homer Help or Homer Handbook, topic *Autouner > System Info Window > Autotune Page*.

Accepted range of values:

- Target 0 to 1000
- Tolerance 0 to 1000
- Tuning pause 0 to 255 (Applies to Server versions starting V60)
- Skipped measurements 0 to 255 (Applies to Server versions V55 to V59)
- Smoothing count 1 to 255
- Tuning delay 0 to 31 (5 bits)
- Hysteresis 0 to 255

Notes:

- Tuning pause has unit of 10 milliseconds. For example, to apply actual Tuning pause 150 ms, set the value to 15.
- This command cannot be broadcast because the message contains more than 7 data bytes (see Section 2.6).
- The case for old Server versions (V54 and less) is not documented here. Please refer to older vresions of this document.

7.1.1 Server Versions Starting V60

CAN

Command Message:

ID	Sender	Byte Count
ci11_AtCmd=17=11h	PC	8

Data bytes:

Byte	Value/Meaning	
0	msCanSetAtPar=3	
1	Tolerance in milliunits – LSB	
2	Tolerance in milliunits – MSB	
3	Tuning pause in units of 10 milliseconds	
4	Smoothing count	
5	b7 – b3	Tuning 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
ci13_AtResp=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 (the rightmost **1**) represents WaitRF.
Response example:

```
19:    3
```

RS232

Command Code	msATunPar=73=49h
Label	ATP
Parameters	**Toler**: Tolerance in milliunits (0 to 1000)
	Pause: Tuning pause in units of 10 milliseconds (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, Pause = 10 (i.e., actual tuning pause = 100 ms), 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 Server Versions V55 to V59

CAN

Command Message:

ID	Sender	Byte Count
ci11_AtCmd=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 count	
5	b7 – b3	Tuning 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
ci13_AtResp=19=13 h	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 (the rightmost **1**) 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
```

7.2 Hysteresis

Description: Sets **Hysteresis** parameter for the autotuning in degrees. For the meaning of Hysteresis, refer to Homer Help or Homer Handbook, topic *Autouner* > *Hysteresis*.

CAN

Command Message:

ID	Sender	Byte Count
ci11_AtCmd=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
ci13_AtResp=19=13h	Homer	3

Data bytes: The same as above; the actual Hysteresis value is 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 (or after skipping the specified number of measurements). Whether or not this command will be actually executed depends on the measured reflection coefficient and the conditions defined by Autotuning Control Parameters.

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 the 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 sets the *Running* flag, bit 1 sets the *Sending* flag (see Section 11.1). For details about **Hom.cfg** file, refer to Homer Help or Homer Handbook, 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
ci11_AtCmd=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 (starting Server version V54)

Homer Response (Server versions starting V59):

ID	Sender	Byte Count
ci13_AtResp=19=13 h	Homer	2

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 higher):

```
17:    5
```

Possible responses:

```
19:    5    0    (Autotune = FALSE)
19:    5    1    (Autotune = TRUE)
19:  133    0    (command fail, Byte 2 irrelevant)
```

Homer Response (Server versions V58 and less):

Case of Setting Autotuning ON and OFF

ID	Sender	Byte Count
ci13_AtResp=19=13 h	Homer	1

Data bytes:

Byte	Value/Meaning
0	Same as in the command (msAtOFF=0 or msAtON=1)

Case of Querying Autotuning State

ID	Sender	Byte Count
ci13_AtResp=19=13 h	Homer	2

Data bytes:

Byte	Value/Meaning
0	Same as in the command (i.e., msGetAtune=5)
1	Undefined

Unfortunately, due to an error in the code of Server V58 and less, Byte 1 is undefined, and therefore the Query command is meaningless.

RS232

Notes:

- Starting with Server version V54 (Jan-2011), this command also enables *querying* current **Autotune** state without affecting it.
- Because of the response ambiguity, the RS232 response has been modified starting with Server version V59. Now, in all three command variations (ATC 0, ATC 1, ATC 2), the values **0** and **1** of the response byte B2 mean the actual **Autotune** state.

Command Code	msATunCmd=72=48h
Label	ATC
Parameters	0 Autotuning OFF 1 Autotuning ON 2 Query Autotuning state (only Server V54 and higher)
Response	Command Execution Confirmation (Section 3.4.3)
Server V59 and higher	Modified Command Execution Confirmation. Meaning of byte B2: 0: **Autotune** = FALSE 1: **Autotune** = TRUE **Other**: Error executing the command
Server V54 to V58	**Commands ATC 0 and ATC 1:** Standard Command Execution Confirmation; B2 = 0. **Command ATC 2:** Modified Command Execution Confirmation. Meaning of byte B2: 0: **Autotune** = FALSE 1: Error executing the command 2: **Autotune** = TRUE
Server less than V54	Standard Command Execution Confirmation, where B2 is an error code (B2 = 0 in case of success)

Example 1:

To switch continuous autotuning ON, the command string is "ATC 1". The complete command is transmitted as the following byte sequence:

```
128, 28, 65, 84, 67, 32, 49, 13, 10, 128, 72
```

Responses may be:

Server V59 and higher

```
128, 28, 72, 1, 128, 4
```
(command execution OK; actual **Autotune** = TRUE)

`128, 28, 72, 3, 128, 4` (command execution fail)

Server V54 to V58

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

Example 2:

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`

Responses may be:

Server V59 and higher

`128, 28, 72, 0, 128, 4` (**Autotune** = FALSE)
`128, 28, 72, 1, 128, 4` (**Autotune** = TRUE)
`128, 28, 72, 3, 128, 4` (command execution fail)

Server V54 to V58

`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.

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 a measurement. The same holds for a load that has changed much since the last autotuning step. For fine-tuning, one command will usually do.
- Consult Section 8.3 on how to launch measurement + autotuning.

CAN

Command Message:

ID	Sender	Byte Count
`ci11_AtCmd`=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 = `ci16_MotQry` = 22 = 16h (see Section 4.1.4) and motors position after the tuning step
- Command confirmation message as follows:

ID	Sender	Byte Count
`ci13_AtResp`=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 = ci16_MotQry = 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	• MDO with motors data, bits 4 and 5 of HST set to 1; *followed by* • 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 4.2.3 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)
```

8. SINGLE-SHOT COMMANDS

This section describes a group of commands which give the user control over the tuning process, including its timing. The commands trigger execution of one Measure-Tune-Send Sequence (MTSS); 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	Computing Tune Positions
3	Moving stubs to Tune Positions
4	Sending Homer Measurement Results
5	Sending Actual Positions

The outcome of *Measurement* action is Homer Measurement Results (HMR). Input data for the Tune Positions (TP) computation are HMR and Actual Positions (AP) 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, HMR+AP.

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. Caution: If Homer is in *Running* state, repeated call of this command gives different results.	HMR
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 6.5), and Single Autotuning Step (Section 7.4).

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 the received 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
ci10_HomCmd=16=10 h	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
ci0B_HmRes1=11=0B h	Homer	8	
ci0C_HmRes2=12=0C h	Homer	8	
ci0D_HmRes3=13=0D h	Homer	8	
ci16_MotQry=22=16 h	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 ci16_MotQry 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*
`ci10_HomCmd`=16=10 h	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
`ci0B_HmRes1`=11=0B h	Homer	8	
`ci0C_HmRes2`=12=0C h	Homer	8	
`ci0D_HmRes3`=13=0D h	Homer	8	
`ci16_MotQry`=22=16 h	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 `ci16_MotQry` 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 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 the final stub settings whereas the measurement results correspond to the starting stub positions.

CAN

Command Message:

ID	Sender	Byte Count
ci11_AtCmd=17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeaTun=88=58h

Homer Response:

ID	Sender	Byte Count	Note
ci0B_HmRes1=11=0Bh	Homer	8	
ci0C_HmRes2=12=0Ch	Homer	8	
ci0D_HmRes3=13=0Dh	Homer	8	
ci16_MotQry=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
```

8.4 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 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
ci11_AtCmd=17=11h	PC	1

Data bytes:

Byte	Value/Meaning
0	msMeaTunMea=89=59h

Homer Response:

ID	Sender	Byte Count	Note
ci0B_HmRes1=11=0Bh	Homer	8	
ci0C_HmRes2=12=0Ch	Homer	8	
ci0D_HmRes3=13=0Dh	Homer	8	
ci16_MotQry=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
```

8.5 ClrFifo Command

Description: Clear Homer‘s internal buffer (FIFO) of received messages (CAN) or bytes (RS232). The command ensures Homer’s immediate response to the subsequent command.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msClrFifo=84=54h

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=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
```

8.6 Get Timeouts

The execution time of a command issued to Homer can vary widely from tens of milliseconds to several seconds, depending, for instance, on:

- Homer settings (e.g., sampling rate and number of samples to be taken for one measurement point).

- Whether individual samples should be transmitted or not in Rectified and Pulsed sampling modes.
- Homer model (e.g., speed of its motors and maximum tuning stub extension).
- Load reflection coefficient (determining whether the stubs are to be moved or not).
- Whether the Homer is in the *Running* state or idle.

After issuing a command, the controller (your PC or PLC) 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 practical 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 an appropriate *measurement* timeout and *motors movement* timeout, and and report them back. The command response 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. In the case of analyzer only, the returned value is zero and should be ignored).

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msGetTmouts=61=3Dh

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=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`).

8.6.1 Using MeasTimeout and MotorsTimeout

An actual timeout (*Tout*) that the controller should apply for a given command depends on a particular situation, for instance:

- whether Homer continuous measurement is running (*Running* flag),
- whether autotuning is activated (*Autotune* flag),
- whether the command is a *motors command* (one which can result in tuning stubs movement).

Motors commands include Stepper Motors commands and single-shot MeaTun and MeaTunMea commands.

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

Running	Autotune	Motors Command	Tmout (ms)	Note
NO	-	NO	1000	A 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, because 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.

8.6.2 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.

8.7 Ping Message

Description: The **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
ci10_HomCmd=16=10h	PC	2

Data bytes:

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

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=18=12h	Homer	2

Data bytes:

Byte	Value/Meaning
0	msPingPong=20=14h
1	Ping Byte *B* (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 *S* representing Ping Byte *B*
Response	1-byte Data Object (Section 3.3.2) with End Message = msPingPong

Data byte:

Byte	Value/Meaning
0	Returned Ping Byte *B*

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 a Homer to a predefined power-up state.

9.1 Restart Server, Halt Server, Change Server

Description:

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

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

The **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, EtherNet/IP)
- Restarts itself with *the same* communication mode and protocol (the same effect as **Restart Server**)
- Starts the RS232 file transfer program SrvDld.exe (the same effect as **Halt Server**)

Obviously, **Change Server** functionality includes those of the **Restart Server** and **Halt Server** commands.

The response depends on the Server version: the details 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
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSrvRst=80=50h

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=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
ci10_HomCmd=16=10h	PC	1

Data bytes:

Byte	Value/Meaning
0	msSrvHalt=34=22h

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=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
ci10_HomCmd=16=10h	PC	2

Data bytes:

Byte	Value/Meaning
0	msSrvRst=80=50h
1	91: Start Server in EtherNet/IP communication mode 95: Exit to operating system 96: Start RS232 file transfer program SrvDld.exe 97: Start Server in RS232 communication mode 98: Start Server in CAN communication mode 99: Restart the currently running Server

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=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	• 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	91: Start Server in EtherNet/IP communication mode 95: Exit to operating system 96: Start RS232 file transfer program SrvDld.exe 97: Restart Server in RS232 communication mode 98: Restart Server in CAN communication mode 99: Restart currently running Server
Response	• 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: This command returns Homer fully to the state it had when it left the factory (out-of-box default) in that it copies all firmware files to the Homer internal working directory from a specific *Factory Default* location. This helps resolve unwanted and 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 the user, with the corresponding firmware files stored in a specific *User Default* location. The **User-Defined Reset** reset is accomplished by executing the **Make User-Defined Reset** command, which is described in the next section.

After performing these types of resets, Homer DC power ought to be cycled.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=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 which were not transferred succesfully (a bit set to 1 means a file transfer fail). The usual reason is that the corresponding file was not found in the source directory. This is because, depending on the Homer options, not all potentially possible files need to be present in *Factory Default* or *User Default* locations. B

elow is the full list of the potentially available files and the corresponding File List bits :

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

The blank cells were occupied by now obsolete files (other files may take their place in future).
As an example, if DAQ .CFG, SRVDLD.EXE and AUTOEXEC.BAT files were missing (**1** in the column *Missing* in the table above), the *Failed File List* = 67712 = 1 00001000 10000000 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 *not* 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* = 67712:

```
18:    8 128   8   1   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* = 67712 (only bytes **B[n]** and *Reset Result*):

```
128, 8, 1, 0, 0
```

9.3 Create User-Defined Reset

Description:

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

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=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* = 67712:

```
18:    8   128   8   1   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* = 67712 (only bytes **B[n]** and *Reset Result*):

```
128, 8, 1, 0, 0
```

10. MEASUREMENT SETUP PARAMETERS

All measurement results, such as the reflection coefficient and incident power, are obtained from the results of measurement of the following quantities:

- Four six-port reflectometer detector voltages when microwave power is applied to them (referred to as signals).
- The detector voltages when the microwave signal is *not* applied to them (referred to as offsets). Offsets are slowly varying, temperature-dependent, quantities, originating primarily from the post-detection DC amplification circuitry.
- Signal frequency.
- 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 signals. This may increase the overall measurement throughput. Other possible methods to increase measurement 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.
- Using fixed A/D converter ranges rather than autoranging.

This section describes settings giving the user more control over details of CW measurement process, like timing of particular measurements or setting of A/D converter ranges. Section 10.3.4 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 immediately after a single-shot command is issued. Timing of data transmitting is also covered here.

10.1.1 Signal Measurement Rate

Description: This setting 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 then a new measurement is started. 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 Rate

Description: This setting defines **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 then a new offset measurement is started. If OfsPeriod is set to zero, the offset is measured anytime the signal is 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: This setting defines **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 then a new frequency measurement is started. If FPeriod is set to zero, frequency is measured anytime the signal is 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 reasonable to let FPeriod_ms=0..

10.1.4 Temperature Measurement Period

Description: This setting defines **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 then a new temperature measurement is started. If TPeriod is set to zero, temperature is measured anytime the signal is 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: This setting defines **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 an 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: This setting defines **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 measurement ranges are not stipulated to be equal to the signal measurement ranges (see below).

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

10.2.3 Offset Ranges Equal to Signal Ranges

Description: This setting defines **OfsEqualOn** boolean flag, which, if TRUE, forces the offset measurement ranges to be the same as those used in the signal measurement. (The settings may be different in different channels if Autoranging was employed for *signals* measurement.) In this case, the 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 a Homer transmits Homer Measurement Results (HMR) and Motors Data (MD).

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 Sequence (MTSS), various events occur, affecting whether data transmission will take place or not. The occurrence of a particular event sets a corresponding bit in an 8-bit array called *Send Event Register* (SER). Before each intended transmission, SER is compared against another, similar, register, called *Send Mask Register* (SMR). The SMR enables for transmission only a selection from the events registered in SER. The Homer Measurement Results (HMR) transmission will only take place if the result of the (SER **and** SMR) bitwise operation is nonzero.

The individual bits of SER have the following meaning:

Bit	Meaning
0	HMR are available
1	HMR sending period has expired
2	Motors Refresh period has expired
3	(not used)
4	(not used)
5	(not used)
6	(not used)
7	(not used)

- **Bit 0** of SER is set to 1 after each successful measurement. If bit 0 of SMR mask is set o 1 as well, HMR are sent anytime a new set of HMR is 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.
- **Bits 3 to 7** of SMR are irrelevant.

10.3.2 Sending Period

Description: This setting defines **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: This setting defines **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 overloaded with motors data. The power-up value is governed by the following line in **Hom.cfg** file:

```
Mot_Refresh=1000
```

If the line is missing or incorrect, the default is also 1000 ms.

10.3.4 Measurement Rate vs. Results Transmitting Cadence

If you want the Homer internal measurement to be running fast but you need the results to be transmitted less frequently (for instance, to avoid overloading the RS232 or CAN interface), you have two choices:

1. Switch *Sending* state to FALSE and query for Homer Measurement Results and Motors data (MD) anytime you wish, using, for instance, the FetchLast Command.
2. Leave *Sending* = TRUE, and
   - Set *Sending Period* to a sufficiently long time.
   - Modify *Send Mask Register* SMR: Set only bit 1, so that SMR = 010 bin = 2.

11. MEASUREMENT SETUP COMMANDS

11.1 Set/Get Running and Sending States

Description: This command 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 DOES NOT transmit either measurement results or motors data (although it may make measurements and autotuning). When *Sending* = TRUE, Homer Measurement Results and Motors Data are transmitted, based on the states of the Send Event Register (SER) and Send Mask Register (SMR).
The power-up ***Running*** and ***Sending*** values are governed by the following line in **Hom.cfg** file:

```
AUTORUN=1
```

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 1 (i.e., *Running* = TRUE , *Sending* = FALSE).

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10 h	PC	3

Data bytes:

Byte	Value/Meaning
0	msSweepStart=17=11h
1	**Running** (0 = FALSE; 1 = TRUE; 2 = keep unchanged = query)
2	**Sending** (0 = FALSE; 1 = TRUE; 2 = keep unchanged = query)

Homer Response:

ID	Sender	Byte Count
ci12_HomResp=18=12h	Homer	3

Data bytes: Same as above; actual values are returned.

Note: The command with setting both *Running* = 1 and *Sending* = 1 is equivalent with the command Start Continuous Measurement.

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 = keep unchanged)
	Sending (`0` = FALSE; `1` = TRUE; `2` = Query = keep unchanged)
Response	**Query (`SRS 2 2`):** 2-byte Data Object (Section 3.3.2) with actual values
	Otherwise: Command Execution Confirmation (Section 3.4.3)

Note: The command with setting both *Running* = 1 and *Sending* = 1 is equivalent with the command Start Continuous Measurement.

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: This command sets **OnPeriod** and **OfsPeriod** for signal and offset measurement, respectively.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=18=12h	Homer	6

Data bytes: Same as above; actual values are 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: This command sets **FPeriod** and **TPeriod** for frequency and temperature measurement, respectively.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=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
ci12_HomResp=18=12h	Homer	6

Data bytes: Same as above; actual values are 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: This command sets **TxPeriod** and **Send Mask Register** SMR.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10 h	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
ci12_HomResp=18=12h	Homer	5

Data bytes: Same as above; actual values are 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: This command sets signal measurement range (**OnRnge**), offset measurement range (**OfsRnge**), and **OfsEqualOn** flag.

CAN

Command Message:

ID	Sender	Byte Count
ci10_HomCmd=16=10 h	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
ci12_HomResp=18=12h	Homer	6

Data bytes: Same as above; actual values are 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. For setting TRUE in OfsEqualOn, any normal text string starting with y, Y, t, T, 1, can be used, e.g., yes, TRUE, etc. For 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: This command sets **MotPeriod** in the range from 0 to 32767 ms or queries it.

To set: The argument (MotPeriod) must be in the interval 0 ≤ MotPeriod ≤ 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
ci10_HomCmd=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
ci12_HomResp=18=12h	Homer	3

Data bytes: Same as above; actual value is 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	msMotRefresh=76=4Ch
Label	XXX
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. MISCELLANEOUS COMMANDS

12.1 Set/Query Format of RS232 Results

Note: This command pertains only to the case of RS232 communication interface.

Description: This command sets or queries format in which the measurement results are transmitted from a Homer unit. This format can be:

- Binary: Each MDO has the form of a stream of bytes that must be decoded to arrive at actual results.
- ASCII: Each MDO has the form of a readable text string that can be easily parsed to arrive at the numerical results. The downside is the larger size of these MDO, and hence slower transmission cadence. For more details, please refer to Homer Help or Homer Handbook, topic *Homer Start* > *Format of RS232 Results*.

RS232

Command Code	`msRsltForm`=35=23h
Label	`RFM`
Parameters	`0` Binary results `1` ASCII results `2` (reserved) `3` Query
Response	Command Execution Confirmation (Section 3.4.3) where B2 is the actual results format value (`0` = Binary, `1` = ASCII)

Example:

To set ASCII format of RS232 results, the command string is "`RFM 1`"; the complete command is transmitted as the sequence

```
128, 28, 82, 70, 77, 32, 49, 13, 10, 128, 35
```

Response:

```
128, 28, 35, 1, 128, 4
```