

# Timer Request Protocol

# Timer Request Protocol

## 1.1 Revision History

| No. | Date               | Subject  |
|-----|--------------------|--|
| 1.0 | June 26, 2012      | Initial release.   |
| 1.1 | July 11, 2012      | Document reformatted.  |
| 1.2 | May 14, 2014       | 4.4.3: Set.Group added.<br>4.5.6: Get.Group commands added.<br>4.8: Error codes added.   |
| 1.3 | May 23, 2014       | 3.2.1: Negative timer values possible in "Basic" mode.<br>3.2.1: Timer Strings in "Full" mode are variable in length.<br>3.2.2: Replaced status "Blinking" by "Flashing".                      |
| 1.4 | May 27, 2014       | 4.8: Error code added: IntegerTooSmall.  |
| 1.5 | June 12, 2014      | 3.2.2: Clarified order of status words.  |
| 1.6 | June 27, 2014      | 4.1: Device name is now separate parameter.  |
| 1.7 | July 4, 2014       | 2.3: Clarified capabilities of standard Rubidium Ethernet firmware.<br>3.2.2: Renamed <FlashStatus> to <DisplayStatus>.<br>3.2.2: Added "Off" status.  |
| 2.0 | March 16, 2015     | Added timer manipulation commands:<br>4.6 Control<br>4.4.4 Set.TimerAll<br>4.5.7 Get.TimerAll ... 4.5.14 Get.DownOverflow<br>4.8: Error codes added.   |
| 2.1 | March 23, 2015     | 4.5.8: Get.MainTimer replaced by Get.Main.<br>4.6.17: Control.MainTimer replaced by Control.Main.<br>4.8: Error codes added.   |
| 2.2 | April 2, 2015      | Removed quotes from TimeFormat and DateFormat.<br>Renamed DisplayDelimiter to Delimiter and changed parameter.<br>3.2.1 Added "Status" timer format mode.<br>4.7 Renamed configuring commands. |
| 2.3 | April 8, 2015      | 4.4.3 Clarified Set.Group.   |
| 2.4 | July 9, 2019       | Fixes some typos.<br>3.2.1 Added "RunStatus" timer format mode.  |
| 2.5 | August 26, 2019    | Changed address of Plura Europe GmbH.  |
| 2.6 | September 16, 2020 | Clarified that Rubidium options R3 and R10 require options M and N.  |

## 1.2 Copyright

Copyright © PLURA Europe GmbH 2012-2020. All rights reserved. No part of this publication may be reproduced, translated into another language, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written consent of PLURA Europe GmbH.

Printed in Germany.

Technical changes are reserved.

All brand and product names mentioned herein are used for identification purposes only and are trademarks or registered trademarks of their respective holders.

Information in this publication replaces all previously published information. PLURA Europe GmbH assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

For further information please contact your local dealer or:

PLURA Europe GmbH

Binger Weg 12

55437 Ockenheim

Phone: +49 - 6725 - 91 80 06 - 70

E-Mail: [info@plurainc.com](mailto:info@plurainc.com)

Internet: <https://www.plurainc.com>

## 1.3 Table of Contents

1

|                                     |           |
|-------------------------------------|-----------|
| <b>Timer Request Protocol .....</b> | <b>2</b>  |
| 1.1 Revision History .....          | 2         |
| 1.2 Copyright .....                 | 3         |
| 1.3 Table of Contents.....          | 4         |
| <br>                                |           |
| <b>2 Overview.....</b>              | <b>6</b>  |
| 2.1 Introduction .....              | 6         |
| 2.2 Communication .....             | 6         |
| 2.3 Client / Server .....           | 6         |
| <br>                                |           |
| <b>3 Communication.....</b>         | <b>7</b>  |
| 3.1 Format .....                    | 7         |
| 3.2 Timers.....                     | 7         |
| 3.2.1 Timer Format Mode .....       | 7         |
| 3.2.2 Timer Status .....            | 8         |
| 3.2.3 Control Time .....            | 9         |
| 3.2.4 Time Format .....             | 9         |
| 3.2.5 Date Format .....             | 10        |
| 3.2.6 Delimiter .....               | 10        |
| 3.2.7 On Off.....                   | 10        |
| <br>                                |           |
| <b>4 Commands .....</b>             | <b>11</b> |
| 4.1 Hello.....                      | 11        |
| 4.2 Subscribe.....                  | 11        |
| 4.2.1 Subscribe.Timer .....         | 11        |
| 4.2.2 Subscribe.Status .....        | 12        |
| 4.2.3 Subscribe.All .....           | 12        |
| 4.3 Unsubscribe .....               | 13        |
| 4.3.1 Unsubscribe.Timer .....       | 13        |
| 4.3.2 Unsubscribe.Status .....      | 13        |
| 4.3.3 Unsubscribe.All.....          | 14        |
| 4.4 Set .....                       | 14        |
| 4.4.1 Set.Refresh .....             | 14        |
| 4.4.2 Set.Format.....               | 15        |
| 4.4.3 Set.Group .....               | 15        |
| 4.4.4 Set.TimerAll.....             | 15        |
| 4.5 Get.....                        | 16        |
| 4.5.1 Get.Timer .....               | 16        |
| 4.5.2 Get.Status .....              | 16        |
| 4.5.3 Get.Version .....             | 17        |
| 4.5.4 Get.Refresh .....             | 17        |
| 4.5.5 Get.Format .....              | 17        |
| 4.5.6 Get.Group.....                | 18        |
| 4.5.7 Get.TimerAll .....            | 18        |

# Timer Request Protocol

---

|          |   |           |
|----------|---|-----------|
| 4.5.8    | Get.Main .....  | 19        |
| 4.5.9    | Get.DisplayFormat.....                                    | 19        |
| 4.5.10   | Get.Delimiter.....  | 19        |
| 4.5.11   | Get.LeadingZeros.....                                     | 20        |
| 4.5.12   | Get.ZerosOnZero .....                                     | 20        |
| 4.5.13   | Get.FlashOnNegative .....                                 | 21        |
| 4.5.14   | Get.DownOverflow .....                                    | 21        |
| 4.6      | Control .....   | 21        |
| 4.6.1    | Control.Start .....                                       | 22        |
| 4.6.2    | Control.Stop .....  | 22        |
| 4.6.3    | Control.Reset.....  | 22        |
| 4.6.4    | Control.Hold .....  | 23        |
| 4.6.5    | Control.Up .....  | 23        |
| 4.6.6    | Control.Down.....   | 23        |
| 4.6.7    | Control.DownStart .....                                   | 24        |
| 4.6.8    | Control.Due .....   | 24        |
| 4.6.9    | Control.DueTc .....                                       | 25        |
| 4.6.10   | Control.OffsetTime .....                                  | 25        |
| 4.6.11   | Control.DiffTime.....                                     | 25        |
| 4.6.12   | Control.OffsetTc .....                                    | 26        |
| 4.6.13   | Control.DiffTc.....                                       | 26        |
| 4.6.14   | Control.SetTime.....                                      | 27        |
| 4.6.15   | Control.SetDate .....                                     | 27        |
| 4.6.16   | Control.ShutOff.....                                      | 27        |
| 4.6.17   | Control.Main .....  | 28        |
| 4.7      | Configure .....   | 28        |
| 4.7.1    | Configure.DisplayFormat.....                              | 28        |
| 4.7.2    | Configure.Delimiter.....                                  | 29        |
| 4.7.3    | Configure.LeadingZeros .....                              | 29        |
| 4.7.4    | Configure.ZerosOnZero.....                                | 29        |
| 4.7.5    | Configure.FlashOnNegative .....                           | 30        |
| 4.7.6    | Configure.DownOverflow.....                               | 30        |
| 4.8      | Error.....  | 31        |
| <b>5</b> | <b>Examples .....</b>                                     | <b>33</b> |
| 5.1      | Subscribing to Timer A.....                               | 33        |
| 5.2      | Subscribing to Timer A with full status and refresh ..... | 33        |

## 2 Overview

### 2.1 Introduction

The Timer Request protocol allows 3<sup>rd</sup> party equipment to show and control timers and status information of a Plura timer system over an Ethernet connection.

It is possible (and recommended) to subscribe to specific timers to get any change of values or status automatically. This prevents the need for polling. The 3<sup>rd</sup> party equipment can select what information it needs and how detailed, depending on its ability and complexity.

### 2.2 Communication

The communication parameters are:

- Port 8851
- TCP
- IP
- 10/100BaseT Ethernet

### 2.3 Client / Server

The Plura timer system acts as a server. Depending on its hardware and/or license it can serve 1 up to 10 clients simultaneously. The Rubidium Ethernet (IE or PM) modules with option R3 can handle 3 clients simultaneously; option R10 can handle 10 clients. Both R3 and R10 require options N and M to be installed, too. SPT's option R can handle only one client.

After connecting to the server, the client will receive the hello string as defined in chapter 4.1.

## 3 Communication

Server and client communicate in ASCII. The client sends commands to the server, the server answers with replies. Commands to the server have to be terminated by <cr> (carriage return, ^M, \r, 0x0D). An extra <lf> (line feed, ^J, \n, 0x0A) is optional and will be ignored. Commands are case-insensitive.

Replies to the client are terminated by <cr><lf>. This allows communicating with the server using Telnet-style client software.

### 3.1 Format

Communication format is as follows. Optional parts are enclosed in square brackets [ ].

#### *Client to server*

```
Command[.SubCommand][:Value[,Value]]<cr>[<lf>]
```

#### *Server to client*

```
Reply[.SubReply]:Value[,Value]<cr><lf>
```

SubCommand is required with most commands. Most commands require or return one or more values.

Values that have blank, dot, colon, comma or semicolon characters included have to be enclosed in quotation marks:

```
Command[.SubCommand]:"Value" [, "Value"]
```

Several commands can be joined in a single line by using a semicolon as a delimiter. The maximum line length is 100 characters, not including the <cr> line termination.

### 3.2 Timers

The following timers are available in a Plura timer system:

```
TimerA  
TimerB  
TimerC  
TimerD  
TimerE  
TimerF  
Time  
Date  
TC
```

The timer names can be used in some of the commands described in chapter 4.

#### 3.2.1 Timer Format Mode

The timers can be requested in one of the following format modes:

```
Basic  
Full  
Status  
RunStatus
```

In “Basic” format mode the timer values are always in format “hh:mm:ss” or “-h:mm:ss”, regardless of the format defined in the timer system. The date will have the fixed format

“dd.mm.yy”. This is the default mode. Enhanced formats like 12h/24h are not transferred in this mode; on the other hand, this mode is easy to implement.

In “Full” format mode the timer values are formatted as defined in the timer system. This includes display formats like “hh.mm.ss”, “mm:ss:ff” and “mm:ss.z” or the “leading zeroes” feature. The values are formatted as an ASCII string with a maximum length of 9 characters. Delimiters like “.” (dot) or “:” (colon) may appear in every column, as well as the “-” (minus) sign. The client should not try to interpret the timer value in this mode but simply display it as an ASCII string.

In “Status” format mode the timer values are formatted as defined in “Full” format mode, but timer status is more detailed (see next paragraph).

In “RunStatus” format mode the timer values are formatted as defined in “Full” format mode, but timer status is again more detailed as in “Status” format mode (see next paragraph).

## 3.2.2 Timer Status

Additional display status attributes are available in the timer system that can be requested if needed, like display status or color.

In “Basic” and “Full” format modes, two attributes are sent:

```
<DisplayStatus>,<Color>
```

<DisplayStatus> is the display status of the timer. It can be one of:

```
Flashing  
Steady  
Off
```

<Color> is the color of the timer. It can be one of:

```
Red  
Green  
Yellow
```

Additional status words may be added in future definitions of this protocol. This may include Up/Down or Stop/Go indicators. The client should silently ignore all unknown status words.

In “Status” format mode, three attributes are sent:

```
<DisplayStatus>,<Color>,<DisplayMode>
```

<DisplayMode> is the display mode of a timer. It can be one of:

```
Up  
Down  
Due  
DueUp  
DueEnd  
DueTc  
DueTcUp  
DueTcEnd  
OffsetTime  
OffsetTc  
DiffTime  
DiffTc  
Timezone  
Date
```



In “RunStatus” format mode, four attributes are sent:

`<DisplayStatus>,<Color>,<DisplayMode>,<RunStatus>`

`<RunStatus>` shows if the timer is currently counting up or down or if it’s stopped. It can be one of:

RunUp  
RunDown  
Stop  
Split

In “Split” status display was stopped using the HOLD key, but the timer continues running in background.

### 3.2.3 Control Time

Some of the commands in chapter 4.6 need to enter a control time parameter:

`<ControlTime>`

`<ControlTime>` can be one of:

`<HH>:<MM>:<SS>`  
`<MM>:<SS>`  
`<SS>`

`<HH>` is a number from 0 to 23, `<MM>` and `<SS>` are numbers from 0 to 59. Leading zeroes are allowed, but not required.

#### *Example*

These are examples of valid control times:

"1:30:00"  
"2:34"  
"5"

### 3.2.4 Time Format

Some of the commands in chapter 4.7 need a time format parameter:

`<TimeFormat>`

`<TimeFormat>` can be one of:

HHMMSS  
HMMSS  
HMMSS12  
SSSSSS  
MMSSFF  
MMMMSS  
MMSST  
SSSST

HH is a placeholder for hours, MM for minutes, SS for seconds, FF for frames and T for tenth of seconds.

HMMSS is used for real-time modes, while HHMMSS is for stop timers. The difference is the handling of leading zeroes. With real-time modes, hours and minutes are always shown, even if they are zero. With stop-timers, hours and minutes may be hidden. HMMSS12 is like HMMSS, but in am/pm mode.

## 3.2.5 Date Format

Some of the commands in chapter 4.7 need a date format parameter:

`<DateFormat>`

`<DateFormat>` can be one of:

DDMMYY

MMDDYY

YYMMDD

DD is a placeholder for day, MM for month and YY for year.

## 3.2.6 Delimiter

Some of the commands in chapter 4.7 need a delimiter parameter:

`<Delimiter>`

`<Delimiter>` can be one of:

Colon

Dot

Blank

That defines possible delimiter used with timers.

## 3.2.7 On Off

Some of the commands in chapter 4.7 need an on or off parameter:

`<OnOff>`

`<OnOff>` can be one of:

On

Off

## 4 Commands

Commands are formatted as defined in chapter 3.1.

### 4.1 Hello

*Command*

```
Hello
```

*Reply*

```
Hello:<HelloString>,<DeviceName>
```

*Description*

This command will request the hello string and the device name. The hello string contains unspecified information about the server like manufacturer and device type. The device name is used-defined.

The hello string together with the device name will also be sent to the client after establishing a connection.

*Example*

To request the hello string, the client would send this command to the server:

```
Hello<cr>
```

The server would reply with:

```
Hello:"PLURA MTD via RUB IE","OB Van 1"<cr><lf>
```

### 4.2 Subscribe

The subscribe commands allow the client to automatically receive every change of the requested information. This prevents the need for polling. It's highly recommended to subscribe to information that will change frequently.

#### 4.2.1 Subscribe.Timer

*Command*

```
Subscribe.Timer:<Timer>
```

*Reply*

```
Subscribing.Timer:<Timer>
```

*Description*

This command subscribes to timer values. The value field contains the timer name as defined in chapter 3.2; it can be set to "All" to subscribe to all available timers. In that case the server will reply several times with all timers that are currently not subscribed.

*Example*

To subscribe to the values of Timer A, the client would send this command to the server:

```
Subscribe.Timer:TimerA<cr>
```

The server would reply with:

```
Subscribing.Timer:TimerA<cr><lf>  
Timer.TimerA:"10:23:56"<cr><lf>
```

## 4.2.2 Subscribe.Status

### *Command*

```
Subscribe.Status:<Timer>
```

### *Reply*

```
Subscribing.Status:<Timer>
```

### *Description*

This command subscribes to the timer status. The value field contains the timer name as defined in chapter 3.2; it can be set to "All" to subscribe to all available timers. In that case the server will reply several times with all timers that are currently not subscribed. See chapter 0 for a description of the possible status values.

### *Example*

To subscribe to the status of Timer A, the client would send this command to the server:

```
Subscribe.Status:TimerA<cr>
```

The server would reply with:

```
Subscribing.Status:TimerA<cr><lf>  
Status.TimerA:Steady,Green<cr><lf>
```

## 4.2.3 Subscribe.All

### *Command*

```
Subscribe.All:<Timer>
```

### *Reply*

```
Subscribing.All:<Timer>
```

### *Description*

This command subscribes to both timer values and status. The value field contains the timer name as defined in chapter 3.2; it can be set to "All" to subscribe to all available timers. In that case the server will reply several times with all timers that are currently not subscribed.

### *Example*

To subscribe to both values and status of Timer A, the client would send this command to the server:

```
Subscribe.All:TimerA<cr>
```

The server would reply with:

```
Subscribing.All:TimerA<cr><lf>  
Timer.TimerA:"10:23:56"<cr><lf>  
Status.TimerA:Steady,Green<cr><lf>
```

## 4.3 Unsubscribe

The unsubscribe command allows the client to cancel the subscription to the requested information.

### 4.3.1 Unsubscribe.Timer

#### *Command*

```
Unsubscribe.Timer:<Timer>
```

#### *Reply*

```
Unsubscribing.Timer:<Timer>
```

#### *Description*

This command cancels subscription to the timer values. The value field contains the timer name as defined in chapter 3.2; it can be set to “All” to cancel subscription to all available timers. In that case the server will reply several times with all timers that are currently subscribed.

#### *Example*

To cancel the subscription to the values of Timer A, the client would send this command to the server:

```
Unsubscribe.Timer:TimerA<cr>
```

The server would reply with:

```
Unsubscribing.Timer:TimerA<cr><lf>
```

### 4.3.2 Unsubscribe.Status

#### *Command*

```
Unsubscribe.Status:<Timer>
```

#### *Reply*

```
Unsubscribing.Status:<Timer>
```

#### *Description*

This command cancels subscription to the status of a timer. The value field contains the timer name as defined in chapter 3.2; it can be set to “All” to cancel subscription to all available timers. In that case the server will reply several times with all timers that are currently subscribed.

#### *Example*

To cancel the subscription to the status of Timer A, the client would send this command to the server:

```
Unsubscribe.Status:TimerA<cr>
```

The server would reply with:

```
Unsubscribing.Status:TimerA<cr><lf>
```

## 4.3.3 Unsubscribe.All

### *Command*

```
Unsubscribe.All:<Timer>
```

### *Reply*

```
Unsubscribing.All:<Timer>
```

### *Description*

This command cancels subscription to both values and status of a timer. The value field contains the timer name as defined in chapter 3.2; it can be set to “All” to cancel subscription to all available timers. In that case the server will reply several times with all timers that are currently subscribed.

### *Example*

To cancel the subscription to both values and status of Timer A, the client would send this command to the server:

```
Unsubscribe.All:TimerA<cr>
```

The server would reply with:

```
Unsubscribing.All:TimerA<cr><lf>
```

## 4.4 Set

This command will set parameters of the server. The settings will be specific for the current connection, i.e. different connections can work with different settings. After a connection has been established, all parameters will be set to their default values. After a connection loss the client has to set specific values again to their desired values.

### 4.4.1 Set.Refresh

#### *Command*

```
Set.Refresh:<Seconds>
```

#### *Reply*

```
Setting.Refresh:<Seconds>
```

#### *Description*

If subscribed to a timer, the client will only receive timer or status values if it has changed. By setting the refresh value the client can receive unchanged timer values or status after a specific number of seconds. This can be used to detect missing subscriptions with a timeout on client's side. Any change of timer values or status will still be sent to the client immediately.

This command will set the refresh timeout for timer values to the specified number of seconds. Setting it to zero will switch off the refresh timeout; this is the default value.

#### *Example*

To set the refresh timeout for timer values to 10 seconds, the client would send this command to the server:

```
Set.Refresh:10<cr>
```

The server would reply with:

```
Setting.Refresh:10<cr><lf>
```

## 4.4.2 Set.Format

### *Command*

```
Set.Format:<Mode>
```

### *Reply*

```
Setting.Format:<Mode>
```

### *Description*

This command will set the timer format mode. See chapter 3.2.1 for a description of the possible modes.

### *Example*

To set the timer's format mode to "Full", the client would send this command to the server:

```
Set.Format:Full<cr>
```

The server would reply with:

```
Setting.Format:Full<cr><lf>
```

## 4.4.3 Set.Group

### *Command*

```
Set.Group:<Group>
```

### *Reply*

```
Setting.Group:<Group>
```

### *Description*

This command will set the group number that will be used for timer and status requests. The default group is that one with the lowest number currently handled by the device. If the selected group becomes unavailable (e.g. by a re-configuration), the new group is selected automatically and can be queried with Get.Group, see 4.5.6.

### *Example*

To set the group number to 1, the client would send this command to the server:

```
Set.Group:1<cr>
```

The server would reply with:

```
Setting.Group:1<cr><lf>
```

## 4.4.4 Set.TimerAll

### *Command*

```
Set.TimerAll:<Timer>,...
```

### *Reply*

```
Setting.Timerall:<Timer>,...
```

## *Description*

This command will set the “TimerAll” timer. “<Timer>,...” is a list one or more timers. See chapter 4.5.8 for a description of TimerAll.

## *Example*

To set the TimerAll timer to timers A, B and D, the client would send this command to the server:

```
Set.TimerAll:TimerA,TimerB,TimerD<cr>
```

The server would reply with:

```
Setting.TimerAll:TimerA,TimerB,TimerD<cr><lf>
```

## **4.5 Get**

This command will request values from the servers. This may be timer values or status as well as values that can be set with the “Set” command.

### **4.5.1 Get.Timer**

#### *Command*

```
Get.Timer:<Timer>
```

#### *Reply*

```
Timer.<Timer>:<Value>
```

#### *Description*

This command requests the value of a specific timer. The value field contains the timer name as defined in chapter 3.2. It is only intended to get a quick prevue of the value; if a timer has to be displayed continuously by the client, it should be subscribed with the “Subscribe.Timer” command described in chapter 4.2.1.

#### *Example*

To get the value if Timer A, the client would send this command to the server:

```
Get.Timer:TimerA<cr>
```

The server would reply with:

```
Timer.TimerA:"10:23:56"<cr><lf>
```

### **4.5.2 Get.Status**

#### *Command*

```
Get.Status:<Timer>
```

#### *Reply*

```
Status.<Timer>:<Value>
```

#### *Description*

This command requests the status of a specific timer. The value field contains the timer name as defined in chapter 3.2. It is only intended to get a quick prevue of the status; if a timer has to be displayed continuously by the client, it should be subscribed with the “Subscribe.Status” command described in chapter 4.2.2.



## *Example*

To get the status of Timer A, the client would send this command to the server:

```
Get.Status:TimerA<cr>
```

The server would reply with:

```
Status.TimerA:Steady,Green<cr><lf>
```

### **4.5.3 Get.Version**

#### *Command*

```
Get.Version
```

#### *Reply*

```
Get.Version:<Version>
```

#### *Description*

This command will request the software version of the server.

#### *Example*

To request the software version of the server, the client would send this command to the server:

```
Get.Version<cr>
```

The server would reply with:

```
Get.Version:"2.15.7"<cr><lf>
```

### **4.5.4 Get.Refresh**

#### *Command*

```
Get.Refresh
```

#### *Reply*

```
Get.Refresh:<Seconds>
```

#### *Description*

This command will request the refresh timeout in seconds.

#### *Example*

To request the refresh timeout, the client would send this command to the server:

```
Get.Refresh<cr>
```

The server would reply with:

```
Get.Refresh:10<cr><lf>
```

### **4.5.5 Get.Format**

#### *Command*

```
Get.Format
```

## *Reply*

```
Getting.Format:<Mode>
```

## *Description*

This command will request the timer format mode. See chapter 3.2.1 for a description of the possible modes.

## *Example*

To request the timer's format mode, the client would send this command to the server:

```
Get.Format<cr>
```

The server would reply with:

```
Getting.Format:Full<cr><lf>
```

## **4.5.6 Get.Group**

### *Command*

```
Get.Group
```

### *Reply*

```
Getting.Group:<Group>
```

### *Description*

This command will request the group number. See chapter 4.4.3 for a description of groups.

### *Example*

To request the current group, the client would send this command to the server:

```
Get.Group<cr>
```

The server would reply with:

```
Getting.Group:1<cr><lf>
```

## **4.5.7 Get.TimerAll**

### *Command*

```
Get.TimerAll
```

### *Reply*

```
Getting.TimerAll:<Timer>, ...
```

### *Description*

This command will request the TimerAll timer. See chapter 4.5.8 for a description of TimerAll.

### *Example*

To request the TimerAll timer, the client would send this command to the server:

```
Get.TimerAll<cr>
```

The server would reply with:

```
Getting.TimerAll:TimerA,TimerB,TimerD<cr><lf>
```

## 4.5.8 Get.Main

### *Command*

```
Get.Main<n>
```

### *Reply*

```
Getting.Main<n>:<Timer>
```

### *Description*

This command will get the selected “main time”. <n> can be 1, 2 or 3. <Timer> can be TimerA to TimerF, Time or Date.

### *Example*

To get main timer 1, the client would send this command to the server:

```
Get.Main1<cr>
```

The server would reply with:

```
Getting.Main1:TimerA<cr><lf>
```

## 4.5.9 Get.DisplayFormat

### *Command*

```
Get.DisplayFormat:<Timer>
```

or

```
Get.DisplayFormat:Date
```

### *Reply*

```
Getting.DisplayFormat:<Timer>,<TimeFormat>
```

or

```
Getting.DisplayFormat:Date,<DateFormat>
```

### *Description*

This command will query the display format of the selected timer. <Timer> can be TimerA to TimerF, Time or TC. If “Date” is used, the date format is queried.

### *Example*

To query the display format of Timer A, the client would send this command to the server:

```
Get.DisplayFormat:TimerA<cr>
```

The server would reply with:

```
Getting.DisplayFormat:TimerA,HHMMSS<cr><lf>
```

## 4.5.10 Get.Delimiter

### *Command*

```
Get.Delimiter:<Timer>
```

### *Reply*

```
Getting.Delimiter:<Timer>,<Delimiter>
```

## *Description*

This command will query the display delimiter of the selected timer. <Timer> can be TimerA to TimerF, Time, Date or TC.

## *Example*

To query the display delimiter of Timer A, the client would send this command to the server:

```
Get.Delimiter:TimerA<cr>
```

The server would reply with:

```
Getting.Delimiter:TimerA,Blank<cr><lf>
```

## **4.5.11 Get.LeadingZeros**

### *Command*

```
Get.LeadingZeros:<Timer>
```

### *Reply*

```
Getting.LeadingZeros:<Timer>,<OnOff>
```

### *Description*

This command will query the “leading zero” display function of the selected timer. <Timer> can be TimerA to TimerF, Time, Date or TC.

### *Example*

To query leading zeros of Timer A, the client would send this command to the server:

```
Get.LeadingZeros:TimerA<cr>
```

The server would reply with:

```
Getting.LeadingZeros:TimerA,On<cr><lf>
```

## **4.5.12 Get.ZerosOnZero**

### *Command*

```
Get.ZerosOnZero:<Timer>
```

### *Reply*

```
Getting.ZerosOnZero:<Timer>,<OnOff>
```

### *Description*

This command will query the “zeros on zero” display function of the selected timer. <Timer> can be TimerA to TimerF.

### *Example*

To query leading zeros on zero of Timer A, the client would send this command to the server:

```
Get.ZerosOnZero:TimerA<cr>
```

The server would reply with:

```
Getting.ZerosOnZero:TimerA,Off<cr><lf>
```

## 4.5.13 Get.FlashOnNegative

### *Command*

```
Get.FlashOnNegative:<Timer>
```

### *Reply*

```
Getting.FlashOnNegative:<Timer>,<OnOff>
```

### *Description*

This command will query the “flash on negative” display function of the selected timer. <Timer> can be TimerA to TimerF.

### *Example*

To query flashing on negative values of Timer A, the client would send this command to the server:

```
Get.FlashOnNegative:TimerA<cr>
```

The server would reply with:

```
Getting.FlashOnNegative:TimerA,On<cr><lf>
```

## 4.5.14 Get.DownOverflow

### *Command*

```
Get.DownOverflow:<Timer>
```

### *Reply*

```
Getting.DownOverflow:<Timer>,<OnOff>
```

### *Description*

This command will query the “down overflow” function of the selected timer. <Timer> can be TimerA to TimerF.

### *Example*

To query the down overflow function of Timer A, the client would send this command to the server:

```
Get.DownOverflow:TimerA<cr>
```

The server would reply with:

```
Getting.DownOverflow:TimerA,On<cr><lf>
```

## 4.6 Control

This command will control timers. This will allow the client to send commands like Start, Stop or Reset, or to change setup of timers like display format. Unlike “Set” commands that are local to the current connection, the “Control” commands affect the timers themselves and therefore all other displays and connections.

Some commands allow to control a “TimerAll” timer. This is a combination of one or more timers that can be set with Set.TimerAll command. When using TimerAll, the command controls any of the selected timers simultaneously.

## 4.6.1 Control.Start

### *Command*

```
Control.Start:<Timer>
```

### *Reply*

```
Controlling.Start:<Timer>
```

### *Description*

This command will start the selected timer. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To start Timer A, the client would send this command to the server:

```
Control.Start:TimerA<cr>
```

The server would reply with:

```
Controlling.Start:TimerA<cr><lf>
```

## 4.6.2 Control.Stop

### *Command*

```
Control.Stop:<Timer>
```

### *Reply*

```
Controlling.Stop:<Timer>
```

### *Description*

This command will stop the selected timer. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To stop Timer A, the client would send this command to the server:

```
Control.Stop:TimerA<cr>
```

The server would reply with:

```
Controlling.Stop:TimerA<cr><lf>
```

## 4.6.3 Control.Reset

### *Command*

```
Control.Reset:<Timer>
```

### *Reply*

```
Controlling.Reset:<Timer>
```

### *Description*

This command will reset the selected timer. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To reset Timer A, the client would send this command to the server:

```
Control.Reset:TimerA<cr>
```

The server would reply with:

```
Controlling.Reset:TimerA<cr><lf>
```

## 4.6.4 Control.Hold

### *Command*

```
Control.Hold:<Timer>
```

### *Reply*

```
Controlling.Hold:<Timer>
```

### *Description*

This command will hold the selected timer, i.e. take a lap time. The timer's display will stop, but the timer continues counting internally and can be shown again e.g. using the "Control.Start" command. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To hold Timer A, the client would send this command to the server:

```
Control.Hold:TimerA<cr>
```

The server would reply with:

```
Controlling.Hold:TimerA<cr><lf>
```

## 4.6.5 Control.Up

### *Command*

```
Control.Up:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.Up:<Timer>,<ControlTime>
```

### *Description*

This command will prepare the selected timer for up counting. The timer will remain in stop mode and can be started later by an extra "Control.Start" command. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To prepare Timer A to count up from 1:30, the client would send this command to the server:

```
Control.Up:TimerA,"1:30"<cr>
```

The server would reply with:

```
Controlling.Up:TimerA,"0:01:30"<cr><lf>
```

## 4.6.6 Control.Down

### *Command*

```
Control.Down:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.Down:<Timer>,<ControlTime>
```

## *Description*

This command will prepare the selected timer for down counting. The timer will remain in stop mode and can be started later by an extra "Control.Start" command. <Timer> can be TimerA to TimerF or TimerAll.

## *Example*

To prepare Timer A to count down from 1:30, the client would send this command to the server:

```
Control.Down:TimerA,"1:30"<cr>
```

The server would reply with:

```
Controlling.Down:TimerA,"0:01:30"<cr><lf>
```

## **4.6.7 Control.DownStart**

### *Command*

```
Control.DownStart:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.DownStart:<Timer>,<ControlTime>
```

### *Description*

This command starts the selected timer down counting. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To start Timer A down counting from 1:30, the client would send this command to the server:

```
Control.DownStart:TimerA,"1:30"<cr>
```

The server would reply with:

```
Controlling.DownStart:TimerA,"0:01:30"<cr><lf>
```

## **4.6.8 Control.Due**

### *Command*

```
Control.Due:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.Due:<Timer>,<ControlTime>
```

### *Description*

This command will prepare the selected timer for DUE mode. The timer will remain in stop mode and can be started later by an extra "Control.Start" command. <Timer> can be TimerB to TimerF.

### *Example*

To prepare Timer C to DUE mode, starting from 1:30, the client would send this command to the server:

```
Control.Due:TimerC,"1:30"<cr>
```

The server would reply with:



```
Controlling.Due:TimerC,"0:01:30"<cr><lf>
```

## 4.6.9 Control.DueTc

### *Command*

```
Control.DueTc:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.DueTc:<Timer>,<ControlTime>
```

### *Description*

This command will prepare the selected timer for DUE TC mode. The timer will remain in stop mode and can be started later by an extra "Control.Start" command. <Timer> can be TimerB to TimerF.

### *Example*

To prepare Timer C to DUE TC mode, starting from 1:30, the client would send this command to the server:

```
Control.DueTc:TimerC,"1:30"<cr>
```

The server would reply with:

```
Controlling.DueTc:TimerC,"0:01:30"<cr><lf>
```

## 4.6.10 Control.OffsetTime

### *Command*

```
Control.OffsetTime:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.OffsetTime:<Timer>,<ControlTime>
```

### *Description*

This command will set the selected timer to the "offset time" mode. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To set Timer A to the "offset time" mode with an offset of one hour, the client would send this command to the server:

```
Control.OffsetTime:TimerA,"1:00:00"<cr>
```

The server would reply with:

```
Controlling.OffsetTime:TimerA,"1:00:00"<cr><lf>
```

## 4.6.11 Control.DiffTime

### *Command*

```
Control.DiffTime:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.DiffTime:<Timer>,<ControlTime>
```

## *Description*

This command will set the selected timer to the “diff time” mode. <Timer> can be TimerA to TimerF or TimerAll.

## *Example*

To set Timer A to the “diff time” mode with a difference to 9 o'clock, the client would send this command to the server:

```
Control.OffsetTime:TimerA,"9:00:00"<cr>
```

The server would reply with:

```
Controlling.OffsetTime:TimerA,"9:00:00"<cr><lf>
```

## **4.6.12 Control.OffsetTc**

### *Command*

```
Control.OffsetTc:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.OffsetTc:<Timer>,<ControlTime>
```

### *Description*

This command will set the selected timer to the “offset tc” mode. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To set Timer A to the “offset tc” mode with an offset of one hour, the client would send this command to the server:

```
Control.OffsetTc:TimerA,"1:00:00"<cr>
```

The server would reply with:

```
Controlling.OffsetTc:TimerA,"1:00:00"<cr><lf>
```

## **4.6.13 Control.DiffTc**

### *Command*

```
Control.DiffTc:<Timer>,<ControlTime>
```

### *Reply*

```
Controlling.DiffTc:<Timer>,<ControlTime>
```

### *Description*

This command will set the selected timer to the “diff tc” mode. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To set Timer A to the “diff tc” mode with a difference to 9 o'clock, the client would send this command to the server:

```
Control.OffsetTc:TimerA,"9:00:00"<cr>
```

The server would reply with:

```
Controlling.OffsetTc:TimerA,"9:00:00"<cr><lf>
```

## 4.6.14 Control.SetTime

### *Command*

```
Control.SetTime:<ControlTime>
```

### *Reply*

```
Controlling.SetTime:<ControlTime>
```

### *Description*

This command will set real-time (local time).

### *Example*

To set real-time to 14:30:10, the client would send this command to the server:

```
Control.SetTime:"14:30:10"<cr>
```

The server would reply with:

```
Controlling.SetTime:"14:30:10"<cr><lf>
```

## 4.6.15 Control.SetDate

### *Command*

```
Control.SetDate:<Date>
```

### *Reply*

```
Controlling.SetDate:<Date>
```

### *Description*

This command will set real-time (local time). Date is in Format "YYYY-MM-DD".

### *Example*

To set date to February 23, 2020, the client would send this command to the server:

```
Control.SetDate:"2020-02-23"<cr>
```

The server would reply with:

```
Controlling.SetDate:"2020-02-23"<cr><lf>
```

## 4.6.16 Control.ShutOff

### *Command*

```
Control.ShutOff:<Timer>
```

### *Reply*

```
Controlling.ShutOff:<Timer>
```

### *Description*

This command will shut off the display of the selected timer. <Timer> can be TimerA to TimerF or TimerAll.

### *Example*

To shut off Timer A, the client would send this command to the server:

```
Control.ShutOff:TimerA<cr>
```

The server would reply with:

```
Controlling.ShutOff:TimerA<cr><lf>
```

## 4.6.17 Control.Main

### *Command*

```
Control.Main<n>:<Timer>
```

### *Reply*

```
Controlling.Main<n>:<Timer>
```

### *Description*

This command will set the selected “main time” to the selected timer. <n> can be 1, 2 or 3. <Timer> can be TimerA to TimerF, Time or Date.

### *Example*

To set main timer 1 to Timer A, the client would send this command to the server:

```
Control.Main1:TimerA<cr>
```

The server would reply with:

```
Controlling.Main1:TimerA<cr><lf>
```

## 4.7 Configure

This command will configure timers. This will allow the client to change the behavior like display format, zeros, flashing, etc. Unlike “Set” commands that are local to the current connection, the “Configure” commands affects the timers themselves and therefore all other displays and connections.

Configuration of timers is normally done by configuring the timer central unit itself. Using the commands in this chapter is designed for special applications that require configuration changes in actual operation.

### 4.7.1 Configure.DisplayFormat

#### *Command*

```
Configure.DisplayFormat:<Timer>,<TimeFormat>
```

or

```
Configure.DisplayFormat:Date,<DateFormat>
```

#### *Reply*

```
Configuring.DisplayFormat:<Timer>,<TimeFormat>
```

or

```
Configuring.DisplayFormat:Date,<DateFormat>
```

#### *Description*

This command will set the display format of the selected timer. <Timer> can be TimerA to TimerF, Time, Date or TC. If “Date” is used, only date formats can be set.

## *Example*

To set the display format of Timer A to HHMMSS, the client would send this command to the server:

```
Configure.DisplayFormat:TimerA,HHMMSS<cr>
```

The server would reply with:

```
Configuring.DisplayFormat:TimerA,HHMMSS<cr><lf>
```

## **4.7.2 Configure.Delimiter**

### *Command*

```
Configure.Delimiter:<Timer>,<Delimiter>
```

### *Reply*

```
Configuring.Delimiter:<Timer>,<Delimiter>
```

### *Description*

This command will set the display delimiter of the selected timer. <Timer> can be TimerA to TimerF, Time, Date or TC.

### *Example*

To remove the display delimiter of Timer A (set it to blank), the client would send this command to the server:

```
Configure.Delimiter:TimerA,Blank<cr>
```

The server would reply with:

```
Configuring.Delimiter:TimerA,Blank<cr><lf>
```

## **4.7.3 Configure.LeadingZeros**

### *Command*

```
Configure.LeadingZeros:<Timer>,<OnOff>
```

### *Reply*

```
Configuring.LeadingZeros:<Timer>,<OnOff>
```

### *Description*

This command will set the “leading zero” display function of the selected timer. <Timer> can be TimerA to TimerF, Time, Date or TC.

### *Example*

To enable leading zeros of Timer A, the client would send this command to the server:

```
Configure.LeadingZeros:TimerA,On<cr>
```

The server would reply with:

```
Configuring.LeadingZeros:TimerA,On<cr><lf>
```

## **4.7.4 Configure.ZerosOnZero**

### *Command*

```
Configure.ZerosOnZero:<Timer>,<OnOff>
```

## ***Reply***

```
Configuring.ZerosOnZero:<Timer>,<OnOff>
```

## ***Description***

This command will set the “zeros on zero” display function of the selected timer. <Timer> can be TimerA to TimerF.

## ***Example***

To disable leading zeros on zero of Timer A, the client would send this command to the server:

```
Configure.ZerosOnZero:TimerA,Off<cr>
```

The server would reply with:

```
Configuring.ZerosOnZero:TimerA,Off<cr><lf>
```

## **4.7.5 Configure.FlashOnNegative**

### ***Command***

```
Configure.FlashOnNegative:<Timer>,<OnOff>
```

### ***Reply***

```
Configuring.FlashOnNegative:<Timer>,<OnOff>
```

### ***Description***

This command will set the “flash on negative” display function of the selected timer. <Timer> can be TimerA to TimerF.

### ***Example***

To enable flashing on negative values of Timer A, the client would send this command to the server:

```
Configure.FlashOnNegative:TimerA,On<cr>
```

The server would reply with:

```
Configuring.FlashOnNegative:TimerA,On<cr><lf>
```

## **4.7.6 Configure.DownOverflow**

### ***Command***

```
Configure.DownOverflow:<Timer>,<OnOff>
```

### ***Reply***

```
Configuring.DownOverflow:<Timer>,<OnOff>
```

### ***Description***

This command will set the “down overflow” function of the selected timer. <Timer> can be TimerA to TimerF.

### ***Example***

To enable the down overflow function of Timer A, the client would send this command to the server:

```
Configure.DownOverflow:TimerA,On<cr>
```

The server would reply with:

```
Configuring.DownOverflow:TimerA,On<cr><lf>
```

## 4.8 Error

### *Command*

```
<Any unknown or malformed command>
```

### *Reply*

```
Error.<Reason>:<Number>
```

### *Description*

Any communication error, like unknown commands, unknown or out-of-range values are replied with an error reply. The reason describes the kind of error, the number tells the exact cause.

The reason "Info" (numbers 201 to 299) is no error, but just informational.

| Reason  | Number |                    | Description   |
|---------|--------|--------------------|---|
| Unknown | 5      | UnknownCommand     | The command is unknown.   |
|         | 6      | UnknownSubCommand  | The sub-command is unknown for this command.  |
|         | 7      | UnknownParameter   | The parameter is unknown for this command or sub-command.                           |
| Format  | 101    | LineTooLong        | The command line is too long.   |
|         | 102    | CommandMissing     | No command found in the command line.   |
|         | 103    | SubCommandMissing  | The command requires a sub-command.   |
|         | 104    | ParameterMissing   | The command or sub-command requires a parameter.                                    |
|         | 105    | ExtraParameter     | The command or sub-command does not allow a parameter.                              |
|         | 106    | DotMissing         | The sub-command has to be separated from the command by a dot ('.').                |
|         | 107    | ColonMissing       | The parameter has to be separated from the command or sub-command by a colon (':'). |
|         | 108    | IntegerTooLong     | The integer parameter is too long.  |
|         | 109    | IntegerExpected    | The command or sub-command requires an integer parameter.                           |
|         | 110    | IntegerTooBig      | The integer parameter value is too big.   |
|         | 111    | IntegerTooSmall    | The integer parameter value is too small.   |
|         | 112    | CommaMissing       | Parameter lists have to be separated by a comma (',').                              |
|         | 113    | DuplicateParameter | Duplicate parameter found in list.  |
|         | 114    | QuoteMissing       | Parameter has to be enclosed with double quotes ("").                               |
|         | 115    | MinusMissing       | Date parameter has to be separated by minus ('-').                                  |

# Timer Request Protocol

| Reason | Number | Description          |  |
|--------|--------|----------------------|--|
| Info   | 201    | NoTimerSubscribed    | All timers are already subscribed.                           |
|        | 202    | NoStatusSubscribed   | All status information is already subscribed.                |
|        | 203    | NothingSubscribed    | All information is already subscribed.                       |
|        | 204    | NoTimerUnsubscribed  | All timers are already unsubscribed.                         |
|        | 205    | NoStatusUnsubscribed | All status information is already unsubscribed.              |
|        | 206    | NothingUnsubscribed  | All information is already unsubscribed.                     |
| Timer  | 301    | GroupNotServiced     | Currently the device does not know about the selected group. |
|        | 302    | TimerNotAllowed      | This timer is not allowed with this command.                 |
|        | 303    | NoTimerSelected      | At least one timer has to be used with this command.         |
|        | 304    | ControlTimeout       | Timeout occurred; command not executed.                      |
|        | 305    | AllTimerNotAllowed   | TimerAll is not allowed with this command.                   |
|        | 306    | StatusNotAvailable   | The requested status is currently not available.             |
|        | 307    | StatusUnknown        | The requested status is unknown.                             |

## *Example*

To test the error reply, the client would send this command to the server:

```
UnknownCommand<cr>
```

The server would reply with:

```
Error.Unknown:5<cr><lf>
```



## 5 Examples

### 5.1 Subscribing to Timer A

To subscribe to Timer A, the client would send the following command to the server:

```
Subscribe.Timer:TimerA<cr>
```

The server would reply with:

```
Subscribing.Timer:TimerA<cr><lf>  
Timer.TimerA:"00:09:56"<cr><lf>  
Timer.TimerA:"00:09:57"<cr><lf>  
Timer.TimerA:"00:09:58"<cr><lf>  
...
```

The timer is shown in “Basic” format per default, that is “hh:mm:ss”.

The server will continue to send the updated timer information in the above form until the client unsubscribes as described in section “Unsubscribe” or closes the connection.

### 5.2 Subscribing to Timer A with full status and refresh

To switch timer format mode to “Full”, subscribe to Timer A’s values and status and set the refresh timeout to 6 seconds, the client would send the following command to the server:

```
Set.Format:Full;Subscribe.All:TimerA;Set.Refresh:6<cr>
```

This is a multi-part command written in a single line. The server would reply with:

```
Setting.Format:Full<cr><lf>  
Subscribing.All:TimerA<cr><lf>  
Timer.TimerA:"9:56"<cr><lf>  
Status.TimerA:Steady,Green<cr><lf>  
Setting.Refresh:6<cr><lf>  
Timer.TimerA:"9:57"<cr><lf>  
Timer.TimerA:"9:58"<cr><lf>  
Timer.TimerA:"9:59"<cr><lf>  
Timer.TimerA:"10:00"<cr><lf>  
Timer.TimerA:"10:01"<cr><lf>  
Timer.TimerA:"10:02"<cr><lf>  
Status.TimerA:Steady,Green<cr><lf>  
Timer.TimerA:"10:03"<cr><lf>  
Timer.TimerA:"10:03.3"<cr><lf>  
Timer.TimerA:"10:03.4"<cr><lf>  
Timer.TimerA:"10:03.5"<cr><lf>  
...
```

In this example Timer A was configured not to show leading zeroes. It was switched from “HH:MM:SS” format to “MM:SS.Z” by another device somewhat after 10:03.

The server will continue to send the updated timer information in the above form until the client unsubscribes as described in section “Unsubscribe” or closes the connection. If the timer’s value or status will not change for more than 6 seconds, the information will be repeated.

**plura**<sup>®</sup>

[www.plurainc.com](http://www.plurainc.com)

U.S.A. . Germany . U.A.E. . S. Korea