# ScienceMode for the MOTIONSTIM8

Description and Protocol

*W. Scholz and K. Busch*

Medel GmbH
Poppenbütteler Bogen 11, 22399 Hamburg
E-mail: kbusch@krauth-timmermann.de

*N.-O. Negård and T. Schauer*

Max Planck Institute for Dynamics of Complex Technical Systems
Systems and Control Theory Group
Sandtorstr. 1, 39106 Magdeburg
E-mail: schauer@mpi-magdeburg.mpg.de

10th February 2004

# 1 Introduction

ScienceMode is a seriel communication protocol to control the 8-channel stimulator MOTIONSTIM8 by Medel GmbH (http://www.medel-hamburg.de) directly from an external device, preferably a PC, via the standard RS232 interface. The ScienceMode offers great flexibility whereas two different stimulation strategies can be distinguished:

1. *Single Pulse Mode:* Sending a command to the stimulator causes a single pulse been sent out on a specific channel with desired current amplitude and pulse width. The stimulator will generate the pulse immediately after processing the command. Complex stimulation patterns may be generated by sending more than one command. The external device is responsible for controlling the stimulation timing, i.e. the stimulation pulse interval.

2. *Channel List Mode:* Using this mode, the generation of complex patterns is greatly simplified. The stimulator is responsible for controlling the stimulation timing. The user can specify a list of stimulation channels, on which repeatedly pulses or even pulse groups (doublets or triplets) will be generated. The stimulation frequency of each channel can be chosen out of two specified stimulation frequencies whereas the higher available stimulation frequency is an integer of the lower available stimulation frequency. Such a setup is handy when applying mixed reflex and muscle stimulation.

   To enter the Channel List Mode an initialisation command must be issued by the external device. Another command is used the exit the Channel List Mode. While the Channel List Mode is active, the pulse parameters (pulse width, current amplitude and stimulation form mode (single pulse, doublet, or triplet) can be altered by sending a pulse parameter update command. The new parameters will be used from the next processing of the channel list on.

The communication between the stimulator and the external device runs with a speed of 115200Bit/s (no parity Bit, 8 Bit). The 8 channels of the stimulator are multiplexed, sharing one current source.

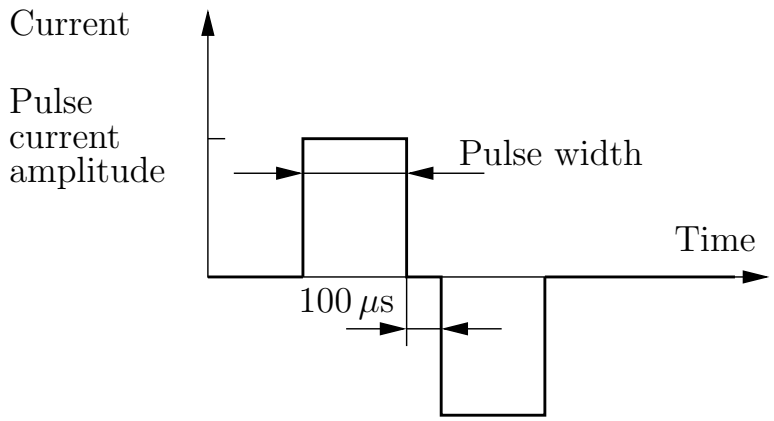


Figure 1: Bipolar pulse with definition of pulse width and current amplitude

All pulses are bipolar (cf. Fig. 1) with the following parameter ranges:

Pulse width:

- Range: 0 and 10-500 $\mu$s
- Resolution 1 $\mu$s

Current:

- Range: 0-127 mA
- Resolution 1 mA

Note that the pulse form will differ from the ideal rectangular bipolar pulse for higher charges as the capacitor of the output stage is not large enough. The Figures **??** and **??** show the measured current and voltage with simulated skin as load for a specified current of 127 mA and a pulse width of 500$\mu$s.

The stimulator will provide automatically a trigger signal at the digital output no. X on the XX conector to mute EMG amplifiers while a stimulation pulse is send (cf. Fig. 1). The mute signal raises from 0 V to 3.3 V just before the stimulation pulse is generated and lasts for 1.5 ms.

Via the stimulator menu the user has further the option to activate a skin resistance check for safety reasons. The resistance is determined by analysing the effect of a small test impulse which is send before each stimulation pulse. If the resistance is not inside normal ranges then stimulation pulse will not be generated.

For using the ScienceMode, a ready to use C++ library and a Matlab/Simulink interface for Linux are available.

# 2 How to activate/deactivate the ScienceMode

The menus of the stimulator are currently only in German. To enter the ScienceMode follow these steps:

1. Hold the keys ’P’ and ’E’ pressed while switching the stimulator on to obtain the menus for “therapeuts” (and engineers, scientists etc.).
2. Press once ’P’ and scroll down the menu to the item ’Einstellungen - Abfragen’ then press ’E’.
3. Scroll down to the item ’Science - Mode’, then press ’E’.
4. Select ’Ausschalten’ and press ’E’ to deactivate the ScienceMode or select ’Einschalten’ and press ’E’ to activate the Sciencemode.
5. Switch the device normally on again to enter the ScienceMode now if selected or run the normal stimulator mode if ScienceMode was deactivated.

English menus will be provided soon by Medel GmbH. The required new firmware can be easily transfered to the stimulator via the serial link.

# 3 Channel List Mode

## 3.1 Initialisation of the Channel List Mode

Before starting the **C**hannel **L**ist **M**ode (CLM), an initialisation of the stimulator has to take place. During this initialisation, a list of stimulation channels has to be specified whereas each selected physical stimulator channel may only appear once in this list.

We have coded the channel list in a Byte `Channel_Stim` whereas the LSB is related to the stimulator channel 1 and the MSB is related to the channel 8. By setting the relevant Bits, stimulation channels are included in the channel list.

While running the stimulation later, the stimulator will generate cyclically stimulation bursts with the time period $t_{s_1}$ on the selected channels.

A stimulation burst may be a single pulse, a doublet or triplet.

*Doublets*: two pulses with a short inter-pulse interval $t_{s_2}$, possessing the same pulse width and current amplitude

*Triplets*: three pulses with a short inter-pulse interval $t_{s_2}$ between each, possessing the same pulse width and current amplitude

From the selected channels in the channel list `Channel_Stim`, one can choose a sub set of channels which will be accessed for stimulation only every $N$-th time the stimulator goes through the channel list. The integer $N$ will be declared as variable `N_Factor` within the protocol. Hence, the stimulation bursts will be generated with a larger time period $N \cdot t_{s_1}$ for this set of channels.

The channels with stimulation period $N \cdot t_{s_1}$ are coded in a byte `Channel_Lf` similar to `Channel_Stim` where `Lf` stands for low frequency. Relevant Bits, set in the Byte, represent channels stimulated with lower frequency. Note that for operating a channel with lower stimulation frequency, the corresponding Bit must be set in both Bytes `Channel_Stim` and `Channel_Lf`!

The time periods $t_{s_1}$ and $t_{s_2}$ are specified by two positive integers `Main_time` and `Group_time` as follows:

- $t_{s_1} =$ `Main_Time` $\cdot\, 0.5\,\mathrm{ms} + 1\,\mathrm{ms}$ with `Main_Time` $= 1 \ldots 2047$
- $t_{s_2} =$ `Group_Time` $\cdot\, 0.5\,\mathrm{ms} + 1.5\,\mathrm{ms}$ with `Group_Time` $= 0 \ldots 31$

The case `Main_Time` = 0 is an exception. Using this setting, the channel list will be processed only once, every time a pulse parameter update command is sent.

## 3.2 Realisation of the Stimulation Sequence

The stimulator has to ensure, that

1. doublets and triplets on all channels are generated with inter-pulse interval $t_{s_2}$,
2. stimulation bursts on every channel are generated with constant period:
   a) $t_{s_1}$ for channels specified in `Channel_Stim` and *not* in `Channel_Lf`,
   b) $N \cdot t_{s_1}$ for channels specified in `Channel_Stim` *and* in `Channel_Lf`.

The following nested strategy is applied within the stimulator in order to realise these requirements (Note, that the special case `Main_Time` $== 0$ is not represented here.):

set $i$ = 1 %Counter for low frequency
if initialisation command enter main loop
  begin
    start timer 0 %timer for $t_{s_1}$
    for $j$ = 0 to largest `Mode` of all channels %loop for generating doublets and triplets if necessary
      begin
        start timer 1 %timer for $t_{s_2}$
        for $k$ = 1 to 8
          begin
            if ((`Channel_stim`[$k$] == 1) and
                ((`Channel_Lf`[$k$] == 0) or
                  ((`Channel_Lf`[$k$] == 1) and ($i$ == `N_Factor`))) then
              begin
                start timer 2
                if ($j$ == `Mode` channel $k$) then send pulse channel $k$
                wait until timer 2 == 1.5 ms
              end
          end
          wait until timer 1 == `Group_Time` $\cdot\, 0.5\,\mathrm{ms} + 1.5\,\mathrm{ms}$
      end
  if $i$ == `N_Factor` then set $i$ = 1
    else increment $i$ by one
  update pulse parameters (pulse width, current, mode) if receiving command
  wait until timer 0 == `Main_Time` $\cdot\, 0.5\,\mathrm{ms} + 1\,\mathrm{ms}$
  exit main loop if receiving quit command
  end

The stimulator will always wait for 1.5 ms before switching to the next channel even if the actual pulse duration is shorter. This guarantees that the stimulation frequencies for the channels will not alter by varying pulse durations. The $1.5\,\mathrm{ms}$ are an integer of the internal stimulator time base of $500\,\mu\mathrm{s}$ and have been chosen to cover the maximal possible duration of a bipolar pulse $(2 \cdot 0.5\,\mathrm{ms} + 0.1\,\mathrm{ms}) = 1.1\,\mathrm{ms})$.

The `Mode` of a channel defines, if a single pulse, a doublet or a triplet is generated.

`Mode` is defined as follows

- `Mode = 0`: Send single pulse
- `Mode = 1:` Send doublet
- `Mode = 2:` Send triplet

There are certain limitations for the selectable time periods using the implementation shown above:

- $t_{s_1} \geq t_{s_2} \cdot \max(\texttt{Mode}) + t_c$
- $t_{s_2} \geq noc \cdot 1.5\,\text{ms}$

where $noc$ is the number of active channels (set Bits) in `Channel_Stim` and $t_c$ is a time interval needed for communication and command processing by the stimulator. $\max(\texttt{Mode})$ is the largest mode used by any of the stimulated channels. The times $t_{s_1}$ and $t_{s_1}$ must further realisable by choosing appreciate positive integers `Main_Time` and `Group_Time`.

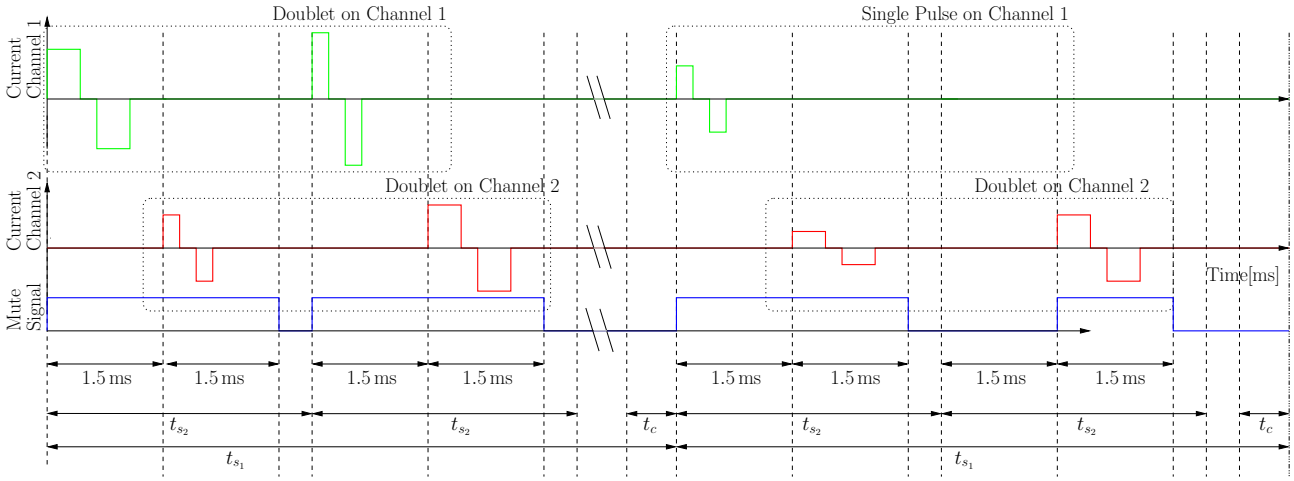


Figure 2: Example for the channel list mode: In the channel list `Channel_Stim` only the channels 1 and 2 are set. None of these channels is used for stimulation with a larger time period than $t_{s_1}$. As a consequence no channels are marked in `Channel_Lf`. During the time period $t_c$ the pulse parameters for the next execution of the channel list are updated. Note that first a doublet is sent on channel 1 and then a single pulse in the next cycle.

!!! The user is responsible for selecting logical values
for `Mode`, `Main_Time` and `Group_Time` !!!

### 3.3 Pulse Parameter Update

The pulse width and current amplitude can be updated for each channel specified in `Channel_Stim` as well as the stimulation mode `Mode` by the pulse parameter update command.
The pulse width can be set 0 or in the range 10–500 $\mu$s. The current setting may be in the rang 0-127 mA with 1 mA resolution.

The updated parameters are valid within the next execution of the channel list.

## 4 Single Pulse Mode

Within the Single Pulse Mode there is a single command to generate a single pulse on one specificed channel with desired pulsewidth and current amplitude. The pulse will be generated directly after the command was processed. Entering the ScienceMode, the Single Pulse Mode will be activated.

## 5 Protocol

### 5.1 Serial Settings

Table 1: Serial settings

| Parameter | Value |
|---|---|
| Baudrate | 115200 |
| parity | no |
| data bits | 8 |
| stop bit | 1 |

Caution: The stimulator serial interface does not provide galvanic isolation. Medel GmbH offers a special cable with built-in opto-couplers.

### 5.2 Commands

All information is contained in packets of one or more bytes conforming with some predefined sequence. In each packet all bytes other than the first have bit 7 clear; the starting byte has bit 7 set.

Table 2: Commands with their 2 bit identification number `Ident`

| No. | Command | Identification number `Ident` |
|---|---|---|
| 1 | Channel list mode initialisation | 00 |
| 2 | Channel list mode update | 01 |
| 3 | Channel list mode stop | 10 |
| 4 | Single pulse generation | 11 |

Table 3: Variables used for channel list mode initialisation command

| Variable | Bits | Value/Range | Description |
|---|---|---|---|
| `Ident` | 2 | 0 | Command identification number |
| `Check` | 3 | 0..7 | Checksum = sum of all logical variables Modulo 8 = (`N_Factor`+`Channel_Stim` +`Channel_Lf`+ `Group_Time`+`Main_Time`) modulo 8 |
| `N_Factor` | 3 | 0..7 | Defines how many times the stimulation is skipped for channels specified in `Channel_Lf` as well as in `Channel_Stim`.<br>0 = no skip<br>1 = skip once<br>... 7 = skip seven times |
| `Channel_Stim` | 8 | 0..256 | Defines the actives channels.<br>Bit 0 corresponds to channel 1.<br>⋮<br>Bit 7 corresponds to channel 8.<br>A bit set activates a channel. |
| `Channel_Lf` | 8 | 0..256 | Defines the low frequency channels.<br>Bit 0 corresponds to channel 1.<br>⋮<br>Bit 7 corresponds to channel 8.<br>A bit set activates a channel for low frequency stimulation but only if the same bit is also set in `Channel_Stim`. |
| `Group_Time` | 5 | 0..31 | Defines the interpulse-interval $t_{s_2}$ by $t_{s_2} = \texttt{Group\_Time} \cdot 0.5\,\text{ms} + 1.5\,\text{ms}$ |
| `Main_Time` | 11 | 0..2047 | Defines the main time period $t_{s_1}$ by $t_{s_1} = \texttt{Main\_Time} \cdot 0.5\,\text{ms} + 1\,\text{ms}$ |

Table 4: Definition of the channel mode list initialisation command

| Byte | Bits | Value | Variable | Bit no. with respect to the variable |
|---|---|---|---|---|
| Byte 1 | 7 | 1 | | |
| | 6 | 0 | Ident | 1 |
| | 5 | 0 | Ident | 0 |
| | 4 | | Check | 2 |
| | 3 | | Check | 1 |
| | 2 | | Check | 0 |
| | 1 | | N_Factor | 2 |
| | 0 | | N_Factor | 1 |
| Byte 2 | 7 | 0 | | |
| | 6 | | N_Factor | 0 |
| | 5 | | Channel_Stim | 7 |
| | 4 | | Channel_Stim | 6 |
| | 3 | | Channel_Stim | 5 |
| | 2 | | Channel_Stim | 4 |
| | 1 | | Channel_Stim | 3 |
| | 0 | | Channel_Stim | 2 |
| Byte 3 | 7 | 0 | | |
| | 6 | | Channel_Stim | 1 |
| | 5 | | Channel_Stim | 0 |
| | 4 | | Channel_Lf | 7 |
| | 3 | | Channel_Lf | 6 |
| | 2 | | Channel_Lf | 5 |
| | 1 | | Channel_Lf | 4 |
| | 0 | | Channel_Lf | 3 |
| Byte 4 | 7 | 0 | | |
| | 6 | | Channel_Lf | 2 |
| | 5 | | Channel_Lf | 1 |
| | 4 | | Channel_Lf | 0 |
| | 3 | X | | |
| | 2 | X | | |
| | 1 | | Group_Time | 4 |
| | 0 | | Group_Time | 3 |
| Byte 5 | 7 | 0 | | |
| | 6 | | Group_Time | 2 |
| | 5 | | Group_Time | 1 |
| | 4 | | Group_Time | 0 |
| | 3 | | Main_Time | 10 |
| | 2 | | Main_Time | 9 |
| | 1 | | Main_Time | 8 |
| | 0 | | Main_Time | 7 |
| Byte 6 | 7 | 0 | | |
| | 6 | | Main_Time | 6 |
| | 5 | | Main_Time | 5 |
| | 4 | | Main_Time | 4 |
| | 3 | | Main_Time | 3 |
| | 2 | | Main_Time | 2 |
| | 1 | | Main_Time | 1 |
| | 0 | | Main_Time | 0 |

Table 5: Variables used for channel list mode update command

| Variable | Bits | Value/Range | Description |
|---|---|---|---|
| `Ident` | 2 | 1 | Command identification number |
| `Check` | 5 | 0..31 | Checksum = sum of all logical variables Modulo 32 = (`Mode`+`Pulse_Width`+`Pulse_Current`) modulo 32 |
| `Mode` | 2 | 0..2 | `Mode` = 0: generate single pulse<br>`Mode` = 1: generate doublet<br>`Mode` = 2: generate triplet |
| `Pulse_Width` | 9 | 0,10..500 | Pulse width in $\mu$s |
| `Pulse_Current` | 7 | 0..127 | Current in mA |

Table 6: Definition of the channel mode update command

| Byte | Bits | Value | Variable | Bit no. with respect to the variable |
|---|---|---|---|---|
| Byte 1 | 7 | 1 | | |
| | 6 | 0 | Ident | 1 |
| | 5 | 1 | Ident | 0 |
| | 4 | | Check | 4 |
| | 3 | | Check | 3 |
| | 2 | | Check | 2 |
| | 1 | | Check | 1 |
| | 0 | | Check | 0 |
| For each channel activated in the channel list, the next three bytes are send in increasing order with respect to the channel number. | | | | |
| Byte 2 | 7 | 0 | | |
| | 6 | | Mode | 1 |
| | 5 | | Mode | 0 |
| | 4 | X | | |
| | 3 | X | | |
| | 2 | X | | |
| | 1 | | Pulse_Width | 8 |
| | 0 | | Pulse_Width | 7 |
| Byte 3 | 7 | 0 | | |
| | 6 | | Pulse_Width | 6 |
| | 5 | | Pulse_Width | 5 |
| | 4 | | Pulse_Width | 4 |
| | 3 | | Pulse_Width | 3 |
| | 2 | | Pulse_Width | 2 |
| | 1 | | Pulse_Width | 1 |
| | 0 | | Pulse_Width | 0 |
| Byte 4 | 7 | 0 | | |
| | 6 | | Pulse_Current | 6 |
| | 5 | | Pulse_Current | 5 |
| | 4 | | Pulse_Current | 4 |
| | 3 | | Pulse_Current | 3 |
| | 2 | | Pulse_Current | 2 |
| | 1 | | Pulse_Current | 1 |
| | 0 | | Pulse_Current | 0 |
| ⋮ | | | | |

Table 7: Variables used for channel list mode stop command

| Variable | Bits | Value/Range | Description |
|---|---|---|---|
| `Ident` | 2 | 2 | Command identification number |
| `Check` | 5 | 0..31 | Checksum = sum of all logical variables Modulo 32 = (0) modulo 32 = 0 |

Table 8: Definition of the channel mode stop command

| Byte | Bits | Value | Variable | Bit no. with respect to the variable |
|---|---|---|---|---|
| Byte 1 | 7 | 1 | | |
| | 6 | 1 | `Ident` | 1 |
| | 5 | 0 | `Ident` | 0 |
| | 4 | 0 | `Check` | 4 |
| | 3 | 0 | `Check` | 3 |
| | 2 | 0 | `Check` | 2 |
| | 1 | 0 | `Check` | 1 |
| | 0 | 0 | `Check` | 0 |

Table 9: Variables used for single pulse generation command

| Variable | Bits | Value/Range | Description |
|---|---|---|---|
| `Ident` | 2 | 3 | Command identification number |
| `Check` | 5 | 0..31 | Checksum = sum of all logical variables Modulo 32 = (`Channel_Number` + `Pulse_Width` + `Pulse_Current`) modulo 32 |
| `Channel_Number` | 3 | 0..7 | `Channel_Number` = 0 is channel no. 1<br>⋮<br>`Channel_Number` = 7 is channel no. 8 |
| `Pulse_Width` | 9 | 0,10..500 | Pulse width in $\mu$s |
| `Pulse_Current` | 7 | 0..127 | Current in mA |

Table 10: Definition of the single pulse generation command

| Byte | Bits | Value | Variable | Bit no. with respect to the variable |
|---|---|---|---|---|
| Byte 1 | 7 | 1 | | |
| | 6 | 1 | Ident | 1 |
| | 5 | 1 | Ident | 1 |
| | 4 | | Check | 4 |
| | 3 | | Check | 3 |
| | 2 | | Check | 2 |
| | 1 | | Check | 1 |
| | 0 | | Check | 0 |
| Byte 2 | 7 | 0 | | |
| | 6 | | Channel_Number | 2 |
| | 5 | | Channel_Number | 1 |
| | 4 | | Channel_Number | 0 |
| | 3 | X | | |
| | 2 | X | | |
| | 1 | | Pulse_Width | 8 |
| | 0 | | Pulse_Width | 7 |
| Byte 3 | 7 | 0 | | |
| | 6 | | Pulse_Width | 6 |
| | 5 | | Pulse_Width | 5 |
| | 4 | | Pulse_Width | 4 |
| | 3 | | Pulse_Width | 3 |
| | 2 | | Pulse_Width | 2 |
| | 1 | | Pulse_Width | 1 |
| | 0 | | Pulse_Width | 0 |
| Byte 4 | 7 | 0 | | |
| | 6 | | Pulse_Current | 6 |
| | 5 | | Pulse_Current | 5 |
| | 4 | | Pulse_Current | 4 |
| | 3 | | Pulse_Current | 3 |
| | 2 | | Pulse_Current | 2 |
| | 1 | | Pulse_Current | 1 |
| | 0 | | Pulse_Current | 0 |

All received frames MOTIONSTIM8 acknowledges with one byte:

Table 11: Acknowledgement byte

| Byte | Bits | Value | Variable | Bit no. with respect to the variable |
|---|---|---|---|---|
| Byte 1 | 7 | | `Ident` | 1 |
| | 6 | | `Ident` | 0 |
| | 5 | | | |
| | 4 | | | |
| | 3 | | | |
| | 2 | | | |
| | 1 | | | |
| | 0 | 1 = OK, 0 = error | `Error_Code` | 0 |

*Single pulse example 1*

Sending a pulse on channel 3 with a pulse width of $200\,\mu$s (binary 011001000) and a current of 120 mA (binary 1111000). Hence, `Channel_Number` is 2 (binary 010). In this case, the checksum is $(2 + 200 + 120)\,\mathrm{modulo}\,32 = 2$ (binary 00010). In binary format, the modulo 32 operation can be easily performed by just taking the 5 LSB of the sum 2+200+120 (binary 101000010). The byte sequence for the command is:

Byte 0 – 11100010
Byte 1 – 0010XX01
Byte 2 – 01001000
Byte 3 – 01111000

The bits with XX do not have a meaning. In hex code the command is E2 21 48 78.
The return value would be in hex C1 (binary 11000001) if no error occurred else C0 (11000000).

*Single pulse example 2*

Sending a pulse on channel 6 with a pulse width of $221\,\mu$s (binary 011011101) and a current of 55 mA (binary 0110111). Hence, `Channel_Number` is 5 (binary 101).The checksum is in this case $(5 + 221 + 51)\,\mathrm{modulo}\,32 = 25$ (binary 11001). In binary format, the modulo 32 operation can be easily performed by just taking the 5 LSB of the sum 5+221+55 (binary 100011001). The byte sequence for the command is:

Byte 0 – 11111001
Byte 1 – 0101XX01
Byte 2 – 01011101
Byte 3 – 00110111

The bits with XX do not have a meaning. In hex code the command is F9 51 5D 37.

The return value would be in hex C1 (binary 11000001) if no error occurred else C0 (11000000).

*Channel list mode initialisation example 1*

The channel list mode shall be initialised with the following parameters:
`Main_Time`=98 (binary 00001100010) gives $t_{s_1}$=50 ms
`Group_Time`=7 (binary 00111) gives $t_{s_2}$=5 ms
`N_Factor`=1 (binary 001)
Channels to be activated are 1,2 and 5 whereas channel 5 runs with lower frequency. This leads to `Channel_Stim`= (binary 0001 0011) = 19 and `Channel_Lf`= (binary 0001 0000) = 16. The checksum is given as 98+7+1+19+16 modulo 8 = 5 (binary 101). In binary format, the modulo 8 operation can be easily performed by just taking the 3 LSB of the sum 98+7+1+19+16 = 141 (binary 10001101). The command byte sequence is then

Byte 1 – 10010100
Byte 2 – 01000100
Byte 3 – 01100010
Byte 4 – 0000XX00
Byte 5 – 01110000
Byte 6 – 01100010

In hex code the command is 94 44 62 0 70 62.

*Channel list mode initialisation example 2*

The channel list mode shall be initialised with the following parameters:
`Main_Time`=31 (binary 00000011111) gives $t_{s_1}$=16.5 ms
`Group_Time`=9 (binary 01001) gives $t_{s_2}$=6 ms
`N_Factor`=2 (binary 010)
Channels to be activated are 2,3,6 and 8 whereas the channel 2 and 3 run with lower frequency. This leads to `Channel_Stim`= (binary 1010 0110) = 166 and `Channel_Lf`= (binary 0000 0110) = 6. The checksum is given as 31+9+166+6+2 modulo 8 = 6 (binary 110). In binary format, the modulo 8 operation can be easily performed by just taking the 3 LSB of the sum 31+9+166+6+2 = 141 (binary 11010110). The command byte sequence is then

Byte 1 – 10011001
Byte 2 – 00101001
Byte 3 – 01000000
Byte 4 – 0110XX01
byte 5 – 00010000
byte 6 – 00011111

In hex code the command is 99 29 40 61 10 1F.

*Channel list mode update example*

This example is for the initialisation example 2 above.

Pulse width channel 2: 100 (001100100)
Pulse width channel 3: 200 (011001000)
Pulse width channel 6: 300 (100101100)
Pulse width channel 8: 400 (110010000)

Current amplitude channel 2: 52 (0110100)
Current amplitude channel 3: 55 (0110111)
Current amplitude channel 6: 72 (1001000)
Current amplitude channel 8: 92 (1011100)

Mode channel 2: 0 (00)
Mode channel 3: 2 (10)
Mode channel 6: 1 (01)
Mode channel 8: 1 (01)

The checksum is given by 100+200+300+400+52+55+72+92+0+2+1+1 modulu 32 = 1+2+8+16 = 28 (binary 11011).

In binary format, the modulo 32 operation can be easily performed by just taking the 5 LSB of the sum 100+200+300+400+52+55+72+92+0+2+1+1 = 1275 (10011111011).

The command byte sequence is then

Byte 0 – 10111011
Byte 1 – 000XXX00
Byte 2 – 01100100
Byte 3 —00110100
Byte 4 – 010XXX01
Byte 5 – 01001000
Byte 6 – 00110111
Byte 7 – 001XXX10
Byte 8 – 00101100
Byte 9 – 01001000
Byte 10 – 001XXX11
Byte 11 – 0010000
Byte 12 – 01011100

In hex code the command is BB 00 64 34 41 48 37 22 2C 48 23 10 5C.

# 6 C++ Library and Tools

Available at sciencestim.sourceforge.net

## 6.1 Matlab Stimulink Interface

## 6.2 Stimulator Test Programm

# 7 Stimulator Specifications

The stimulator possesses two sockets and one infrared port to connect to external sensors or control devices. The description of the related plugs is given in the Tables 12 and 13.

Table 12: Mini DIN plug, 7-pin

| Pin | Description | Label |
|---|---|---|
| 1 | Analog input 3 (max. 10 Bit in the range 0-2.5V) | S3 |
| 2 | Analog input 2 (max. 10 Bit in the range 0-2.5V) | S2 |
| 3 | Analog input 4 (max. 10 Bit in the range 0-2.5V) | S4 |
| 4 | Analog input 1 (max. 10 Bit in the range 0-2.5V) | S1 |
| 5 | Power supply for external devices (sensors etc.) (8.4 V, max. 100 mA, battery voltage) | |
| 6 | Ground | |
| 7 | Analog input 5 (max. 10 Bit in the range 0-2.5V) | S5 |

# References

Table 13: RJ45 plug, 8-pin

| Pin | Description | Label |
|---|---|---|
| 1 | RX3 (serial port used for ScienceMode) | |
| 2 | TX3 (serial port used for ScienceMode) | |
| 3 | RX1 (serial port used for firmware update, etc.) | |
| 4 | TX1 (serial port used for firmware update, etc.) | |
| 5 | digital I/O (low $<$ 0.5 V, high $>$ 2.5 V) | |
| 6 | digital I/O (low $<$ 0.5 V, high $>$ 2.5 V) | |
| 7 | digital I/O (low $<$ 0.5 V, high $>$ 2.5 V) | |
| 8 | GND | |