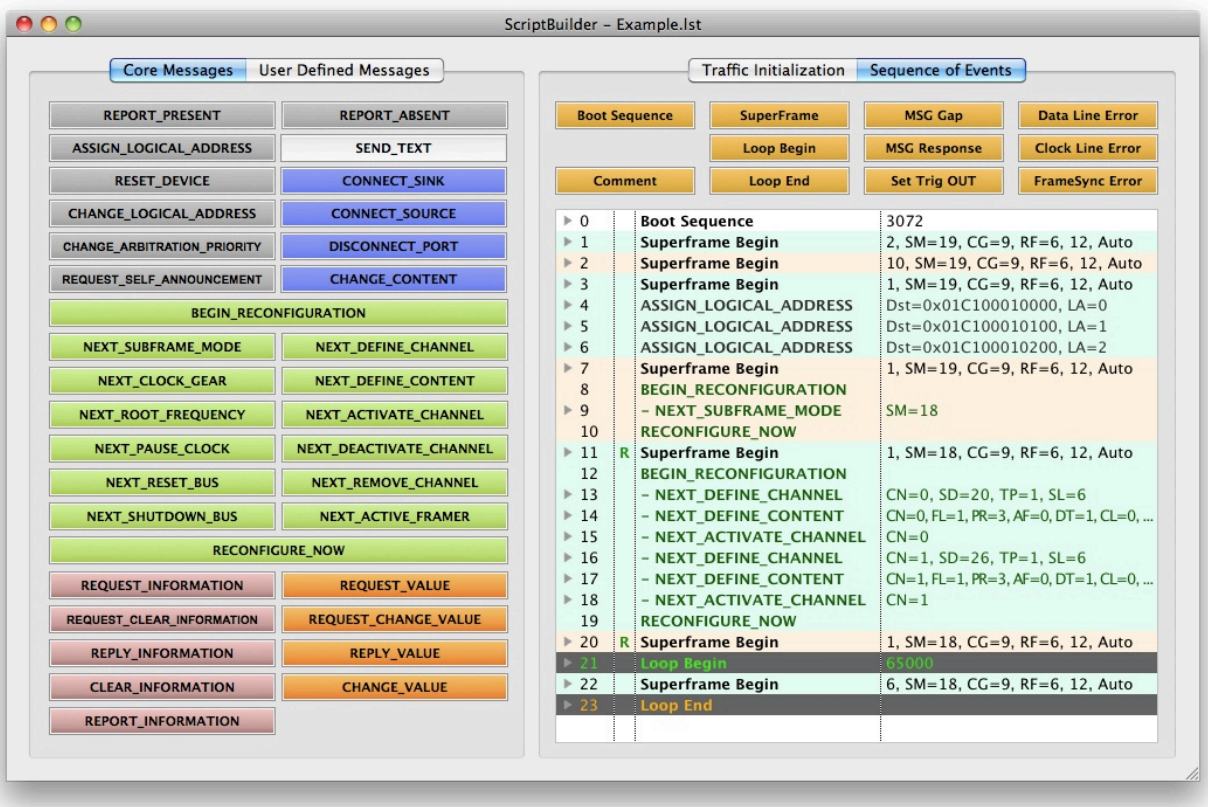
ScriptBuilder User Manual



LnK
44, rue des Combattants
B-4624 Romsée
Belgium
www.lnk-tools.com
info@lnk-tools.com

Table of Content

1. Introduction 3
2. Script Edition 4
   2.1. Traffic Initialization 5
      2.1.1. Hardware Configuration 5
      2.1.2. Framing Information 6
      2.1.3. HW Guide Byte 7
      2.1.4. Framer 7
      2.1.5. Data channels 7
      2.1.6. Message Response 11
      2.1.7. Traffic Generator Outputs 13
      2.1.8. Full Init XML Section 14
   2.2. Sequence of Event 15
      2.2.1. ScriptBuilder Specific Events 15
         2.2.1.1. Boot Sequence 15
         2.2.1.2. Superframe Begin 15
         2.2.1.3. Message Gap 16
         2.2.1.4. Message Response 18
         2.2.1.5. Set Trigger Output 19
         2.2.1.6. Data Line Error 19
         2.2.1.7. FrameSync Error 20
         2.2.1.8. Clock Line Error 21
         2.2.1.9. Loops 22
      2.2.2. Core Messages 24
      2.2.3. User Defined Messages 26
   2.3. XML Script Generation 29
3. Special Features 30
   3.1. Device Register Configuration 30
   3.2. Script Automator 32
   3.3. SLIMbus Protocol Checker 34
      3.3.1. Message parameter check 34
      3.3.2. Framing Information consistency check 34
      3.3.3. Message protocol check 35
      3.3.4. Message channel capacity overflow 36
   3.4. Import Message Capture 37
   3.5. SLIMbus Bandwidth Management Tool Box 39
4. Script Example 40
   4.1. Script Initialization 40
   4.2. Script Event List 42
5. Remote Control and DLL 45

1. Introduction

ScriptBuilder has 2 purposes:
- It is a SLIMbus protocol simulator, to some extent, allowing the user to see the effects of the reconfiguration sequences on the bus configuration and the data channels. It also spots protocol errors. Used together with the SLIMbus Bandwidth Management tool box (SLIMbusMan), it will display graphically the bus configuration over time.
- It generates from a given event list an XML script that can be directly fed in the SLIMbus Traffic Generator. The tool allows building a script by mostly using the mouse.

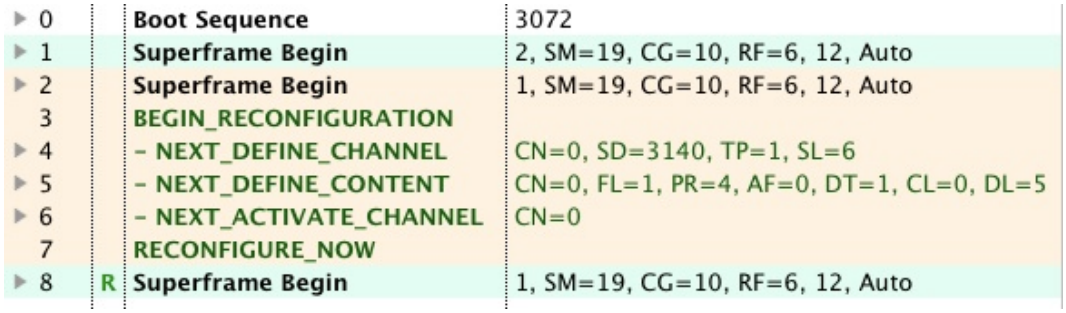
The main window of ScriptBuilder is split in 2 zones. The first one on the left side contains the Core (or User) messages that can be transmitted on SLIMbus. The right one contains the list of events or initialization parameters.

This manual will describe in details all the parameters the user can configure in a given entry of the Event list or in the Parameter list. For each entry, its translation onto the XML scripting language used by the SLIMbus Traffic Generator will be shown. For detailed information about the XML tags, refer to the Traffic Generator user manual.

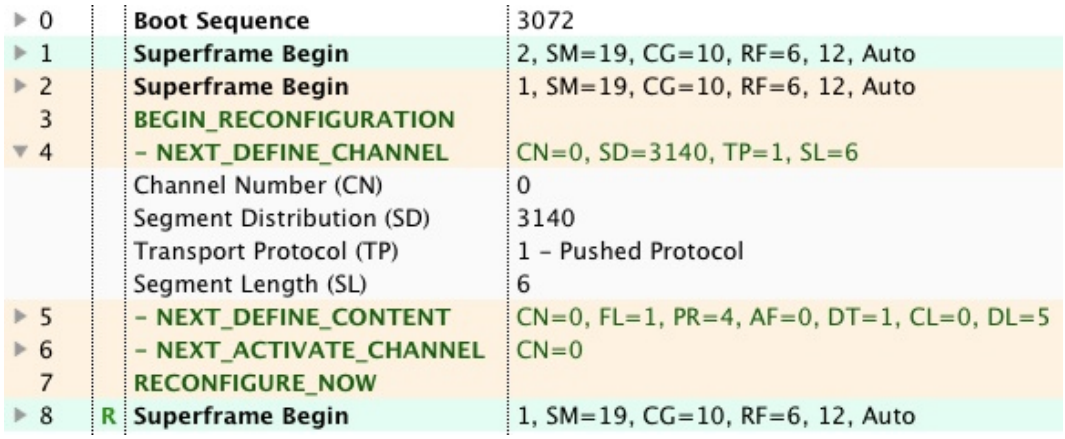
2. Script Edition

The user interface is very simple. The software makes intensive use of contextual menu to drive the user choice whenever possible. Activate the contextual menu by right-clicking in an editable cell. Mac OS users will have to either right-click or press the CTRL key while clicking.

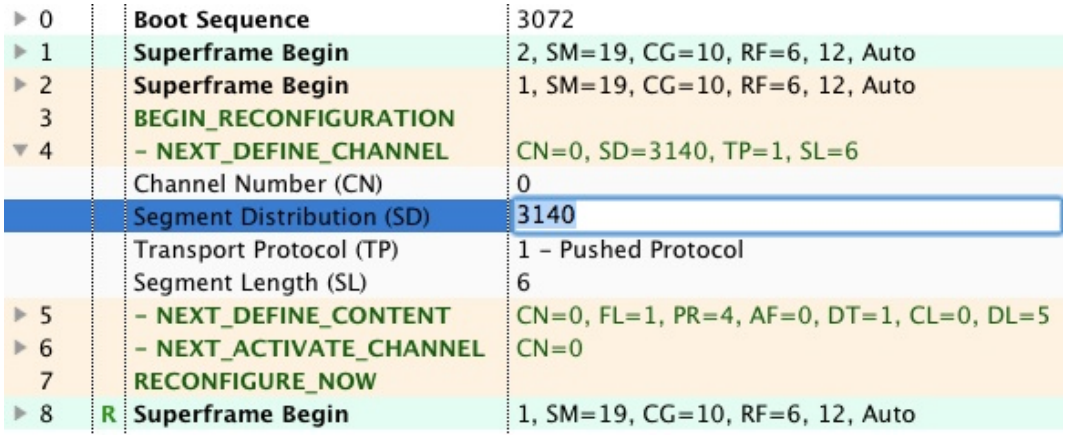
Items are listed in a listbox and can be expanded to access all the parameters of the item. To edit a parameter, expand the event and click on the cell containing the parameter value.



List of Events (all collapsed)



Expand the desired Event



Click the parameter value cell to edit

Values are entered as decimal value, hex value (0xYZ format) or obtained from the contextual menus.

2.1. Traffic Initialization

The Traffic Initialization listbox will display the items that shall appear in the <Init> section of the XML script. There are 4 permanent items: The hardware configuration, the Framing Information to be used during the boot sequence, the way the guide byte is handled by the hardware and the Framer mode (Enabled or Disabled).

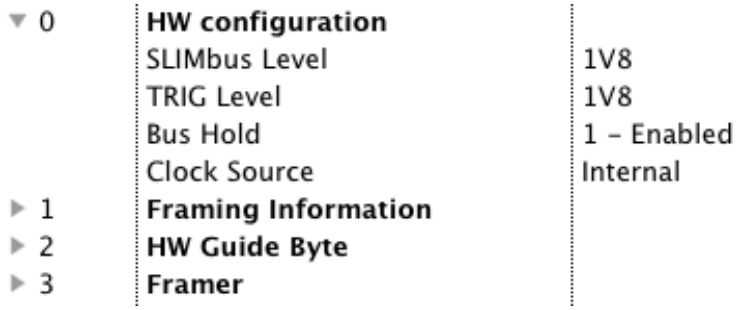
In addition, data channel information can be specified. These values are used when:
- A data stream is already active (the reconfiguration messages are not part of the script).
- The traffic generator feeds the data channel (sine wave or file content).

The data channel and initial bus configuration can also be imported from the SLIMbus Bandwidth Management Tool Box software.

The behavior of the hardware Message Response engine is programmed by adding an "Automatic MSG response" item in the list.

2.1.1. Hardware Configuration

There are four items in the Hardware Configuration entry:
- **SLIMbus Level** can take the value 1V2, 1V8 or 3V3. This is the signaling voltage used on the SLIMbus clock and data lines.
- **TRIG Level** can take the value 1V2, 1V8 or 3V3. This is the signaling voltage used on the trigger input and output.
- **Bus Hold** can take the value "0 - Disabled" or "1 - Enabled". When set to 1, the hardware bus hold will be active.
- **Clock Source** can take the value "Internal" or "External". When set to "Internal", the Traffic Generator hardware will use its internal PLL to generate the required SLIMbus clocks. When set to "External", the Traffic Generator hardware will use the clock signal fed on the "Framer Clock Input "connector. The frequency must be equal to the desired Root Frequency. The "External" option is not available on all hardwares.



XML translation

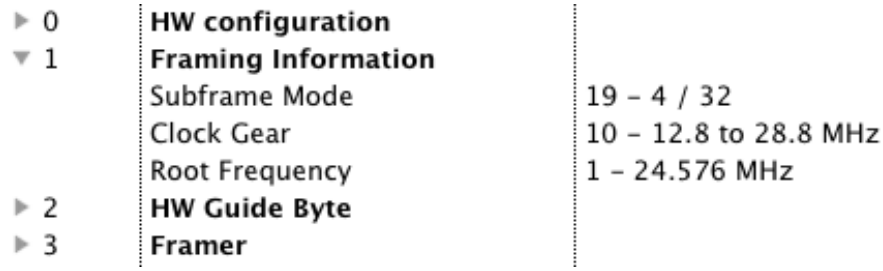
The XML tag is *<Board>*. The tag parameters are *SBLevel*, *TrigLevel*, *BusHold* and *ClkSrc*.

```
<!-- Hardware configuration parameters -->
<Board SBLevel="1V8"
       TrigLevel="1V8"
       BusHold="1"
       ClkSrc="Internal" />
```

2.1.2. Framing Information

There are three items in the Framing Information entry:

- **Subframe Mode** can take any of the subframe mode values (ranging from 0 to 31). This parameter defines the subframe mode used by the active framer during the boot sequence. The subframe mode describes the framing structure and in particular the partition between the control bandwidth and the data bandwidth. For instance, a value "19 - 4 / 32" will specify a subframe length of 32 slots, out of which 4 slots are used for the control space.
- **Clock Gear** can take any of the clock gear values (ranging from 0 to 15). This parameter defines the clock gear to be used during the boot sequence. For instance, a clock gear equal to 10 will set the framer to its maximum frequency.
- **Root Frequency** can take any of the root frequency values (ranging from 0 to 15). This parameter defines the root frequency to be used during the boot sequence. For instance, a root frequency equal to 1 will instruct the framer to use a clock of 24.576 MHz. Used together with the clock gear parameter, the root frequency will unambiguously define the bus frequency at boot time.



Please refer to the SLIMbus specification for all details about the framing information parameters.

XML Translation:

There are 2 XML tags for this functionality: *<Clock>* and *<FramingChannel>*. Clock is mainly there to specify a boot frequency. Both HW and SW traffic generation need to know accurately the frequency to be used at boot time. It is possible to define a "not indicated" root frequency for the Root Frequency. In this case, ScriptBuilder will set a value by default of 24 MHz for the root frequency. *<Clock>* will reflect it while *<FramingChannel>* will still show "Not Indicated".

```
<!-- Boot clock frequency -->
<Clock Freq="24576000" RF="1" CG="10" ClockGearFollow="1" />

<!-- Framer boot value for Framing Information -->
<FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                WhiteningStream="Auto" PhasingStream="Auto" />
```

The Whitening bit and Phasing stream are by default set to automatic generation. Leave it like this in the Init section.

2.1.3. HW Guide Byte

There is one item in the HW Guide Byte entry:

- **Mode** can take the value “Dynamic” or “Static”. When set to “Static”, the Guide Byte value is defined in the script and is not altered by any incoming message traffic. When set to “Dynamic”, the Guide byte is automatically generated by the hardware and is always taking into account any incoming message traffic.

XML Translation:

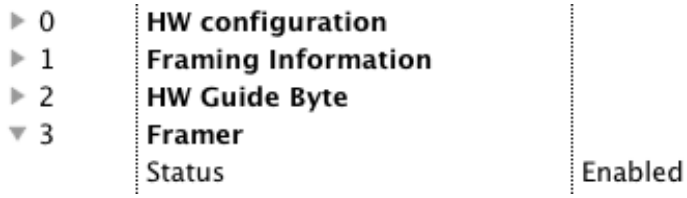
The XML tag is *<GuideByte>* and the parameter is *Val*.

```
<!-- HW mode of operation for Guide Byte dynamic update -->
<GuideByte Val="Dynamic" />
```

2.1.4. Framer

There is one item in the Framer entry:

- **Status** can take the value “Enabled” or “Disabled”. When set to “Enabled”, the traffic generator will build and write the framing channel to the bus. This includes the Framing Information bits and the Frame Sync symbol. When set to “Disabled”, the Traffic Generator will rely on an external framer to operate.



When disabling the framer, it is necessary to specify the bus configuration in the “Framing Information” entry. The Traffic Generator will rely on this information to place the messages and data in the right slots. Failing to give the right parameters will inevitably lead to collisions and disfunction of the bus.

XML Translation:

The XML tag is *<Framer>* and the parameter is *Status*.

```
<!-- Enable or disable the framer functionality of the TG -->
<Framer Status="Enabled" />
```

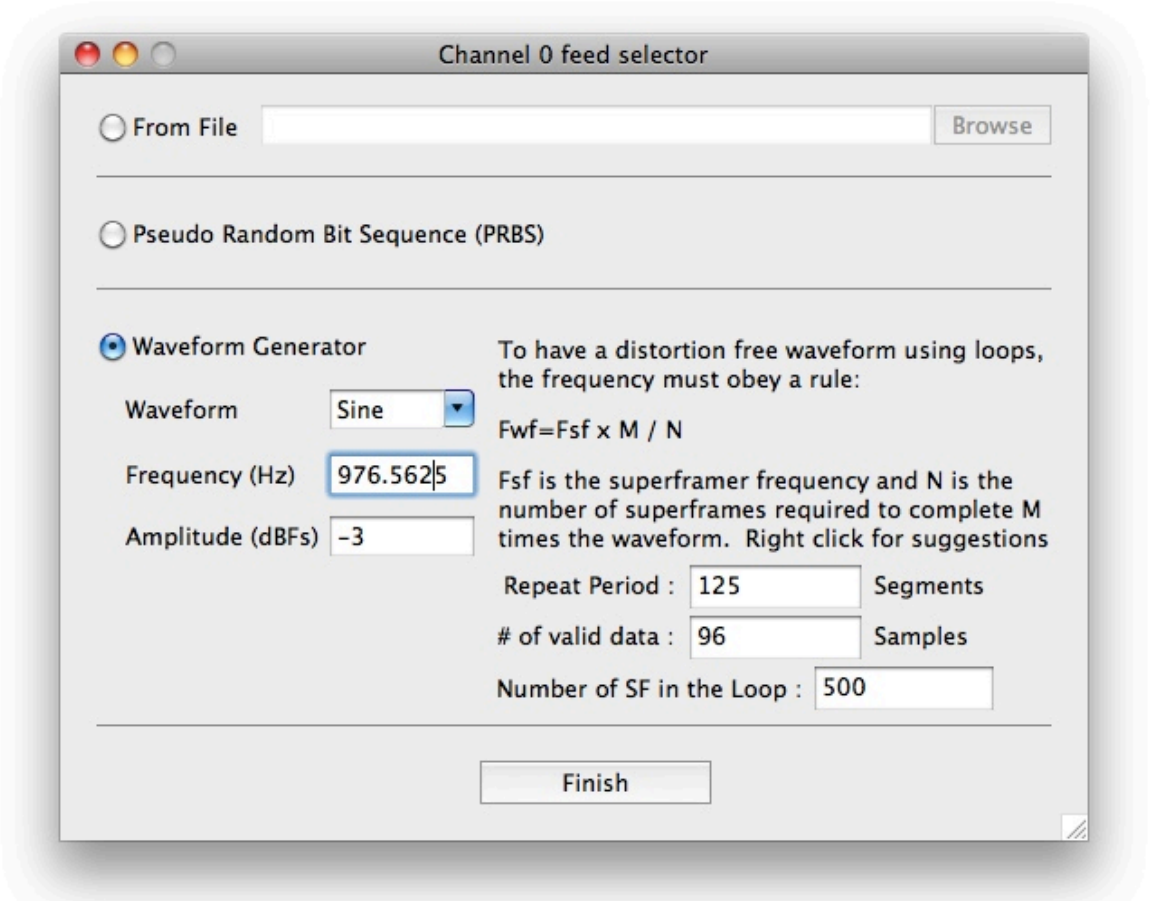
2.1.5. Data channels

This is an optional entry. To add it in the Traffic Initialization list, click on the “Channel Feed” button. An entry named “Channel Definition” will appear in the list. Note that there can be many of these entries, each one defining a specific data channel.

There are twelve items in the Channel Definition entry. However, only the first 2 are necessary.

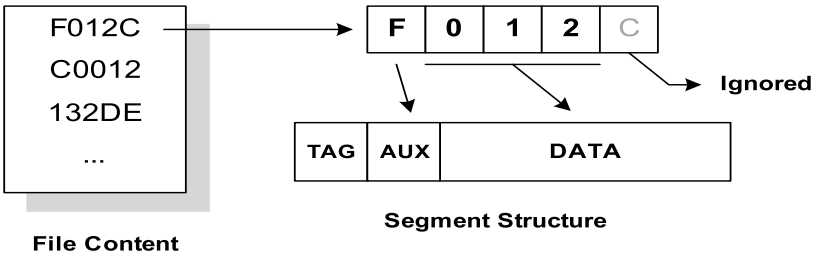
- **Channel Number** is the channel identifier used in SLIMbus. It ranges from 0 to 255. ScriptBuilder will automatically start numbering at 0 and will increase it by one unit each time a new entry is created.
- **Signal feed** actually mixes many thing. A right click in the parameter cell will show a signal source selector that will help the user to properly build the parameter value.

▸ 0	**HW configuration**	
▸ 1	**Framing Information**	
▸ 2	**HW Guide Byte**	
▸ 3	**Framer**	
▾ 4	**Channel Definition**	
	Channel Number	0
	Signal Feed	Sine: 1kHz; -3dBFs
	Segment Distribution	36
	Segment Length	7
	Transport Protocol	1 - Pushed Protocol
	Frequency Locked	0 - Frequency unlocked
	Presence Rate	3 - 48kHz
	AUX Format	0 - Not applicable
	Data Type	1 - LPCM audio
	Channel Link	0 - Channel not linked
	DataLength	5
	Sink	TG



The data channel can be fed by a wave form (Sine, Square, Dirac, Triangle), a pseudorandom bit sequence or the content of a data file. Right click on the frequency text field to get a list of suggested frequencies close to the entered one that will allow distortion free waves using the HW loops. When the Pushed and (to less extent) the Pulled protocol are used, the tool will take them into account to suggest the smallest number of superframes to put in the HW loop that will guarantee distortion free waves. The *Repeat Period* and the *Number of Valid Data* are given for information.

The file shall provide the content of the AUX bits (if any) and the content on the DATA bits. The TAG bits are handled by the traffic generator itself. The file format shall be .TXT. Each text line describes the content of a data sample, using hexadecimal characters only (0..9, A..F). Each character corresponds to a slot (4 bits). The data will be closed by a CR/LF sequence. If the number of characters in the text line in the file is longer than the Data+Aux field length, only the first characters (left justified) will be used. The last ones (at the right) will be ignored. If the text length in the file is shorter than the Data+Aux field length, the remaining slots are stuffed with 0's.
The following example shows a data channel segment having 1 AUX slot and 3 DATA slots. The text data has 5 characters (= 5 slots). Therefore, the last character is not used.



When using the internal waveform generator, it is important to know that the period of the wave must fit an integer number of times in an integer number of superframes.

$$F_{Sine}=F_{Superframe} \times M / N$$

M is the number of times the sine wave repeats over **N** superframes. M and N are both integers. This is the sine qua non condition to have a distortion free signal.

Example:

Let's compute the frequency of the sine waves that can be generated with the following bust configuration:
- Clock gear = 9 (6.4 to 14.4 MHz)
- Root Frequency = 1 (24.576 MHz)

The actual bus clock is equal to 24.576MHz/2=12.288 MHz
A superframe has 6144 bits. Its frequency is therefore equal to 12288000 / 6144 = 2000 Hz.
The sine wave reference frequency is therefore 2000 x M / N Hz.
To get a 1 kHz sine wave, we must set N=2 and M=1.
Note that the Subframe Mode does not play a role in the definition of the sine wave frequency.

An additional constraint is given by the use of the Pushed or Pulled protocols. The Presence bit or Sample Request bit pattern generated by the traffic generator is computed by using a fractional divider that uses two parameters:
- Repeat Period of the Segments

- The number of valid samples in these Segments.

remainder *should be initialized at 0;*
repeat_period *should be the number of samples at which the pattern repeats*
number_of_valid_samples *should be the number of valid samples inside the repeating pattern.*

```
bool valid_sample()
{
        remainder += number_of_valid_samples;
        if (remainder < repeat_period) return false;
        else
        {
                remainder -= repeat_period;
                return true;
        }
}
```

To get a clean repeatable P or SRQ bit pattern, we must make sure that an integer number of Repeat Period will fit in an integer number of superframes. The trick consists in finding the smallest number of superframes that satisfies both:
- an integer number of wave periods in an integer number of superframes
- an integer number of Repeat Period in an integer number of superframes.

The solution is the Least Common Multiple of the wave period and the Repeat Period expressed in slots.
ScriptBuilder computes that value based on the channel rate, the Presence Rate and the desired wave frequency.

The last parameter "**Sink**" is used to trigger a specific behavior of the traffic generator when it is set as a channel sink. If "Sink" is not equal to "TG", the traffic generator will not do anything special.
When **Sink**="TG" and the Transport Protocol is set to "Pulled", the Traffic generator will control (write) the SRQ bit only in the Data Channel Segment. If the source of the data is a file, the value of the SRQ bit will be extracted from the file content. If the source of the data is the sine wave generator, the SRQ bit will be generated by a fractional divider that will compute the optimal sample spread, based on the channel rate and the presence rate.
When "Sink"="TG" and the Transport Protocol is set to one of the Asynchronous protocols, the Traffic generator will control (write) the CTS bit only in the Data Channel Segment. The source of the data must be a file and the value of the CTS bit will be extracted from the file content.

The other parameters of the Channel Definition entry can be left blank if the channel is defined by the appropriate Reconfiguration Sequence in the core of the script. In this case, ScriptBuilder will automatically fill the empty cells.

If the script does not contain the channel definition sequence, the user must fill manually the channel parameters.
- Segment Distribution (SD), Segment Length (SL) and Transport Protocol (TP) are normally specified by the NEXT_DEFINE_CHANNEL message. Refer to section 2.2 of this document or to the SLIMbus specification (section 11) for more detailed information.

- Frequency Locked (FL), Presence Rate (PR), AUX format (AF), Data Type (DT), .Channel Link (CL) and Data Length (DL) are normally specified by the NEXT_DEFINE_CONTENT or CHANGE_CONTENT messages. Refer to section 2.2 of this document or to the SLIMbus specification (section 11) for more detailed information.

XML Translation:

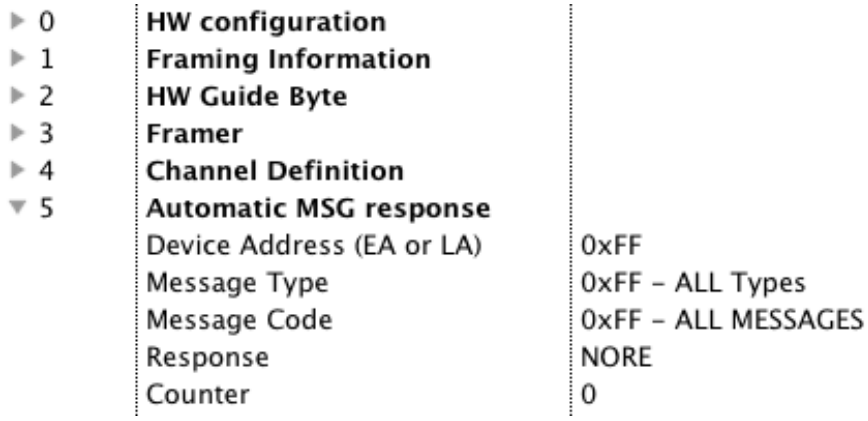
The XML tag is <DataStream>. The parameters are split into 2 categories: <Structure> and <Content>. The Structure does not rely on the Segment distribution but rater on a segment offset and segment interval. These two values are pointing to a unique Segment Distribution value.

```
<!-- Definition of the channel #0 data stream -->
<DataStream Id="0" Dst="TG" Wave="sine" Freq="1000" Amplitude="-3dBFs">
    <Structure Offset="4" Interval="64" Length="6" Protocol="0" />
    <Content FreqLock="0" PresenceRate="3" AuxBitFormat="0" DataType="1"
             ChannelLink="0" DataLength="6" />
</DataStream>
```

2.1.6. Message Response

There are five items in the Automatic Message Response entry:

- **Device Address** is the SLIMbus address of the device sending the message to the traffic generator. Address can be an Enumeration Address (EA) or a Logical Address (LA). Any other source addresses will be ignored by this entry.
- **Message Type** is a filter to apply on incoming messages. The parameter can be omitted. By default, there is no filtering applied on Message Type. When filtering is required, the value of the parameter has to be set accordingly to the SLIMbus specification.
- **Message Code** is a filter to apply on incoming messages. The parameter can be omitted. By default, there is no filtering on Message Code. When filtering is required, the value of the parameter has to be set accordingly to the SLIMbus specification.
- **Response** is the value written in the Response Slot of the incoming message. The value can be forced to any value (“PACK”, “NACK”, “NORE”...).
- **Counter** will set the number of time the hardware will generate the specified response to messages that correspond to the filters described above. Once the counter reach 0, the default answer is defined by the validity of the message (“PACK” if OK, “NACK” if CRC if failing).



There can be up to **15** entries for the Automatic Message Response engine.

XML Translation:

The XML tag is *<AutoResponse>*. The automatic message response is defined by 2 lines:

```
<!-- Force the HW to dynamically generate message responses -->
<AutoResponse Mode="Dynamic" />

<!-- New entry in the Automatic Message Response list -->
<AutoResponse   Address="0xFF"   MsgType="0xFF"   MsgCode="0xFF"   Resp="NORE"
Cntr="0" />
```

"Mode" enables or disables the automatic message response engine. The values can be either "Dynamic" or "Static". Dynamic means that the message response will be generated by the hardware. "Static" means that the message response is taken from the script itself. When using HW streaming, always enable the Automatic Message Response engine.

Note that when the Automatic Message Response engine is enabled, it will generate a message response to ALL incoming messages. The other entries are necessary to be more specific with the behavior of the engine.

Example 1:

Set the Automatic Message response entry in such a way that the messages transmitted by the hardware as Manager will not be acknowledged by the hardware itself. This setup is necessary not to hide the message answers of the other devices.

Device Address must be set to 0xFF (Logical Address of the Manager).
Message Type is set to ALL types.
Message Code is set to ALL messages.
Response is set to NORE (No Response).
Counter is left to 0 (no counter activated).

Example 2:

Set the Automatic Message response entry in such a way that the REPORT_PRESENT message sent by the device 0x001C100010000 will be NACKed 5 times and then PACKed.

Device Address must be set to 0x01C100010000 (Enumeration Address of the Device).
Message Type is set to 0x00 - Cores Message.
Message Code is set to 0x01 - REPORT_PRESENT.
Response is set to NACK (Negative ACKnowledgment).
Counter is set to 5.

2.1.7. Traffic Generator Outputs

ScriptBuilder writes all the necessary tags for the VCD output trace generation.

```
<!-- Output Format="VCD" FileName="Script.vcd" -->
<VCD Module="testbench1"
     NRZIDataChar="!"
     ClockChar="@"
     LogicalDataChar="$"
     DataTxEnaChar="_"
     ClockTxEnaChar="*"
     FrameSyncChar="+"
     SFrameStartChar="-" />
```

It is commented by default. Just uncomment the tag <Output> to enable the generation of a VCD file containing the clock and data streams. Do not modify the other lines.

```
<Output Format="VCD" FileName="Script.vcd" >
```

Make sure that the file name contains the full path.

The generation on the hardware is systematically enabled when the hardware is connected to the PC.

2.1.8. Full Init XML Section

The XML generation of the Traffic Initialization entries will look like the following listing. There is normally no reasons to do manual editing on that section of the XML script.

```
<Init>
    <!-- Boot clock frequency -->
    <Clock Freq="24576000" RF="1" CG="10" ClockGearFollow="1" />

    <!-- Framer boot value for Framing Information -->
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Hardware configuration parameters -->
    <Board SBLevel="1V8"
           TrigLevel="1V8"
           BusHold="1"
           ClkSrc="Internal" />

    <!-- Output Format="VCD" FileName="Script.vcd" -->
    <VCD Module="testbench1"
         NRZIDataChar="!"
         ClockChar="@"
         LogicalDataChar="$"
         DataTxEnaChar="_"
         ClockTxEnaChar="*"
         FrameSyncChar="+"
         SFrameStartChar="-" />

    <!-- Enable or disable the framer functionality of the TG -->
    <Framer Status="Enabled" />

    <!-- Definition of the channel #0 data stream -->
    <DataStream Id="0" Dst="" Wave="sine" Freq="1202.54" Amplitude="-3dBFs">
        <Structure Offset="4" Interval="128" Length="6" Protocol="0" />
        <Content FreqLock="1" PresenceRate="3" AuxBitFormat="0" DataType="1"
                 ChannelLink="0" DataLength="6" />
    </DataStream>

    <!-- New entry in the Automatic Message Response list -->
       <AutoResponse  Address="0xFF"  MsgType="0xFF"  MsgCode="0xFF"  Resp="NORE"
Cntr="0" />

    <!-- Force the HW to dynamically generate message responses -->
    <AutoResponse Mode="Dynamic" />

    <!-- HW mode of operation for Guide Byte dynamic update -->
    <GuideByte Val="Dynamic" />

</Init>
```

2.2. Sequence of Event

The event list will contain all the script events to happen on the SLIMbus.
Events are either messages (both Core and User defined) or the specific events like errors, start of superframes, ...

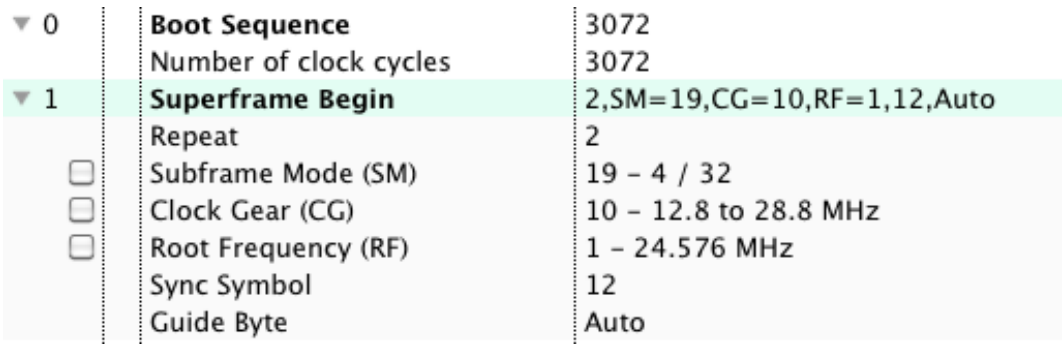
Events are all contained in a given superframe. This was the only method to be able to control the data stream down to the bit level. Therefore, a script must start with either a “Boot Sequence” or a “Superframe Begin”.

Note that ScriptBuilder monitor the bandwidth allocated to messages and will automatically insert a “Superframe Begin” event if the control space bandwidth is exceeded.

2.2.1. ScriptBuilder Specific Events

2.2.1.1. Boot Sequence

ScriptBuilder inserts 2 events in the list. The boot sequence itself, characterized by the number of clock cycles people want to use in the *StartingClock* state. The default value is 3072, according to the SLIMbus specification. However, this value can be set to any value to stimulate corner cases in the SLIMbus IP. The second entry is a group of 2 superframes with no other content than the framing channel.



In normal situation, none of the parameters of the boot sequence should be modified.

XML Translation:

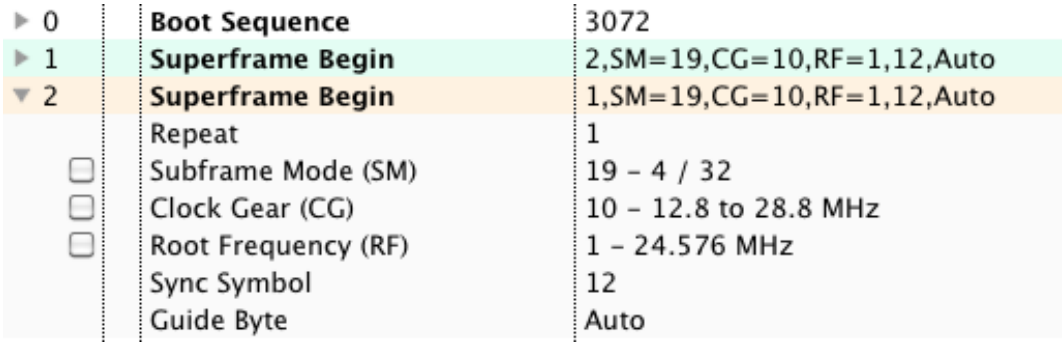
```
<!-- Boot sequence followed by 2 empty superframes -->
<Boot ClockToggle="3072" />
<Superframe Repeat="2" >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />
</Superframe>
```

2.2.1.2. Superframe Begin

To initiate any transaction, first place the beginning of a superframe in the sequence of events.

There are six parameters in the “Superframe Begin” event:

- **Repeat** defines the number of times the superframe and its content is replicated.
- **Subframe Mode** is the subframe mode (SM) value used in the Framing Information of the superframe. When the check box is not ticked, the value is automatically define by ScriptBuilder to keep it compliant with the SLIMbus Protocol. To manually force an arbitrary value, tick the check box and enter the desired value.
- **Clock Gear** is the clock gear (CG) value used in the Framing Information of the superframe. When the check box is not ticked, the value is automatically define by ScriptBuilder to keep it compliant with the SLIMbus Protocol. To manually force an arbitrary value, tick the check box and enter the desired value.
- **Root Frequency** is the root frequency (RF) value used in the Framing Information of the superframe. When the check box is not ticked, the value is automatically define by ScriptBuilder to keep it compliant with the SLIMbus Protocol. To manually force an arbitrary value, tick the check box and enter the desired value.
- **Sync Symbol** is the superframe punctured sync symbol. The default value is decimal 12. To generate a superframe sync error, change that value to something else.
- **Guide Byte** is the value of the guide byte used at the beginning of the superframe. WHen set to “auto”, it will be computed and generated by the Traffic Generator. Note that when the hardware guide byte engine is activated, this value is ignored.



XML Translation:

This is an example of an empty superframe. The XML container is *<Superframe>*. All events happening in that superframe must be enclosed by the <Superframe> and </Superframe> tags. In every superframe container, there must be a <FramingChannel> tag.

```
<Superframe Repeat="1" >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

</Superframe>
```

2.2.1.3. Message Gap

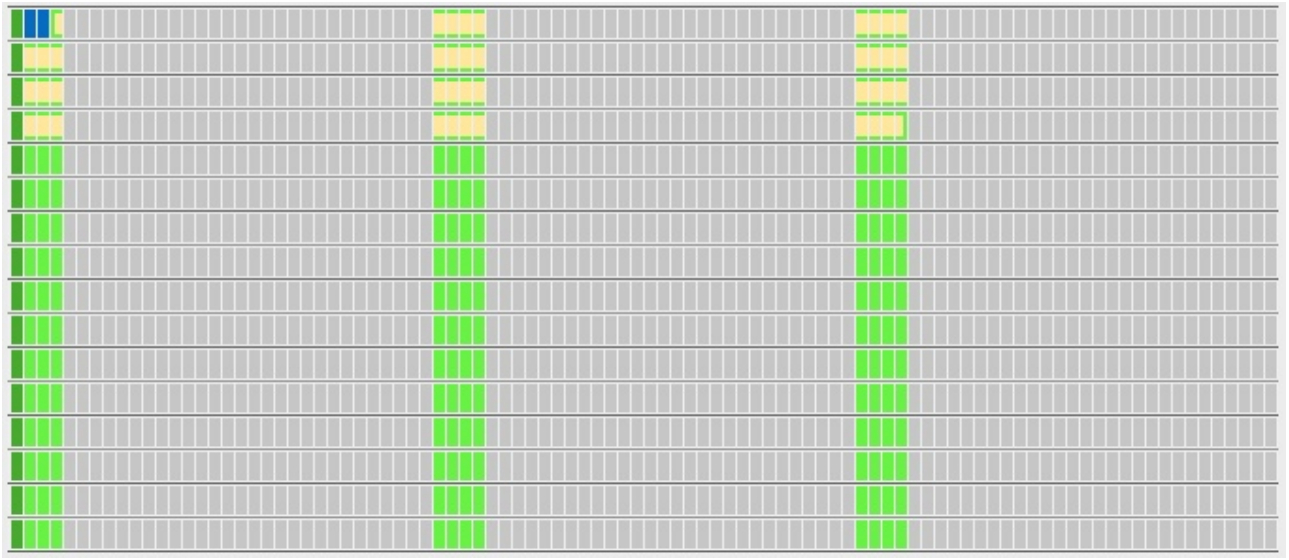
This is not an event as such bit it allows to add some delays between to messages. The gap is defined by a number of slots in the control space. Be sure to count control slots only. For instance, with a subframe mode equal to 19 (4 control slots per 32 slots), there are 192 control slots, out of which 16 are used for the Framing Channel (Framing Info + Frame sync symbol) and 2 are used for the Guide Byte. So there are 174 control slots left while a superframe has 1536 slots in total.

As an example, to force a message to start at the beginning of the third frame, a gap of 42 slots need to be inserted before the message. The value 42 is obtained with the following operation:

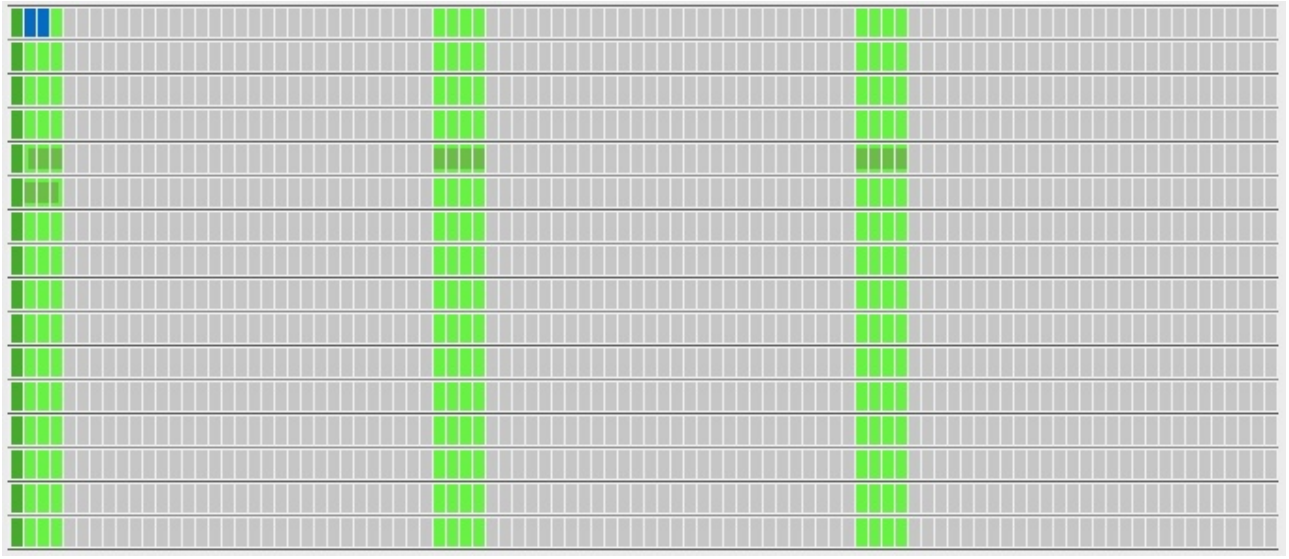
24 control slots per frame x 2 frames - 4 Framing Info slots - 2 Guide byte slots = 42 slots.

▶ 2	**Superframe Begin**	1,SM=19,CG=10,RF=1,12,Auto
▼ 3	**Message Channel Gap**	42
	Number of Slots	42
▶ 4	RESET_DEVICE	Dst=0x00

The following pictures show a complete superframe organized vertically with 8 frames, each of the frame being displayed by 2 strips of 96 slots, leading to 16 strips of 96 slots. The Guide byte slots are shown in blue and the Framing Inflo slots are shown in dark green on the left side of the pictures.



Gap slots are shown in yellow



Message slots (RESET_DEVICE in the example) are shown in dark green

There can be more than one Message Gap entry in the sequence of events. Message gaps are also used to force a message to cross the superframe boundary.

XML Translation:

The XML tag is *<Gap>* and the parameter is *NumberOfSLots*.

```
<!-- Event #2 : Superframe Begin -->
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : Message Channel Gap -->
    <Gap NumberOfSlots="42" />

</Superframe>
```

2.2.1.4. Message Response

When using the Traffic Generator to build a VCD stimulus file, it is necessary to manually place responses to incoming messages. The normal procedure is to run the simulation once to see where the incoming message is occurring. Then modify the script to include the message response at the right place. The procedure might take some time and iterations but offers a simple way to achieve interactivity in a static simulation trace.

The Message Response event has two parameters:

- **Slot Number** is the slot where the message response shall be written in the superframe. The value range from 0 to 1535. Obviously, the value 0 is forbidden as the message end will never happen in a frame sync slot.
- **Response** is the value that shall be written in the slot. It ranges from 0 to 15. The Traffic Generator will use the Logical OR signaling method to write that value.

▶ 2	**Superframe Begin**	1,SM=19,CG=10,RF=1,12,Auto
▶ 3	RESET_DEVICE	Dst=0x00
▼ 4	**Message Response**	129,PACK
	Slot Number	129
	Response	PACK

XML Translation:

The XML tag is *<MsgResp>* and the parameters are *Slot* and *Resp*.

```
<!-- Event #2 : Superframe Begin -->
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />
    <!-- Event #3 : Message Response -->
    <MsgResp Slot="129" Resp="PACK" />

</Superframe>
```

2.2.1.5. Set Trigger Output

It is possible to control the trigger output signal in the XML script. Use the SetTrigOut command.

The command has one parameter

- **Slot** is the slot where the trigger output pulse shall happen in the superframe. The value range from 0 to 1535. The pulse always starts at cell C3 of the slot. By default the pulse has one clock cycle duration but the duration can be modified in the Protocol Analyzer software.

XML Translation:

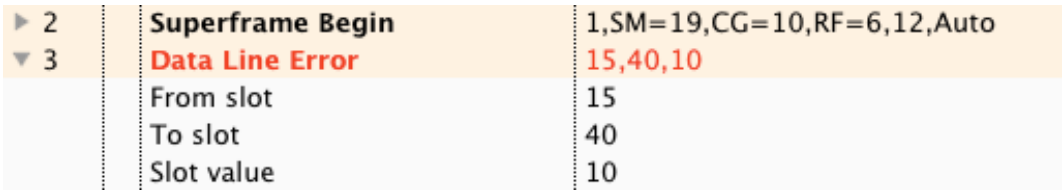
The XML tag is *<SetTrigOut>* and the parameter is *Slot*.

```
<!-- Event #2 : Superframe Begin -->
<Superframe >
    <FramingChannel ClockGear="9" FrameExt="0" RootFreq="6" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : Set Trig OUT -->
    <SetTrigOut Slot="0" />

</Superframe>
```

2.2.1.6. Data Line Error

This event writes an arbitrary value in one or more consecutive slots. As it is happening outside any protocol, it will be used to generate errors on the data line wherever the user needs. There are three parameters to define:

- **From Slot** is the slot number in the superframe of the first slot where data will get assigned a specific value. The value range from 0 to 1535.
- **To Slot** is the slot number in the superframe of the last slot where data will get assigned a specific value. The value range from 0 to 1535.
- **Slot Value** is the value written in the slots comprised between “From Slot” to “To Slot”. the value ranges from 0 to 15.



The following picture shows, as an example, the location of the “error” slots (as defined earlier in the event). In this example, the “error” slots are written both in data space (grey) and control space (green).

XML Translation:

The XML container is *<Error>*, the XML tag is *<EData>* and the parameters are *StartSlot*, *EndSlot* and *Val*.

```
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="6" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : Data Line Error -->
    <Error>
        <EData StartSlot="15" EndSlot="40" Val="10" />
    </Error>
</Superframe>
```

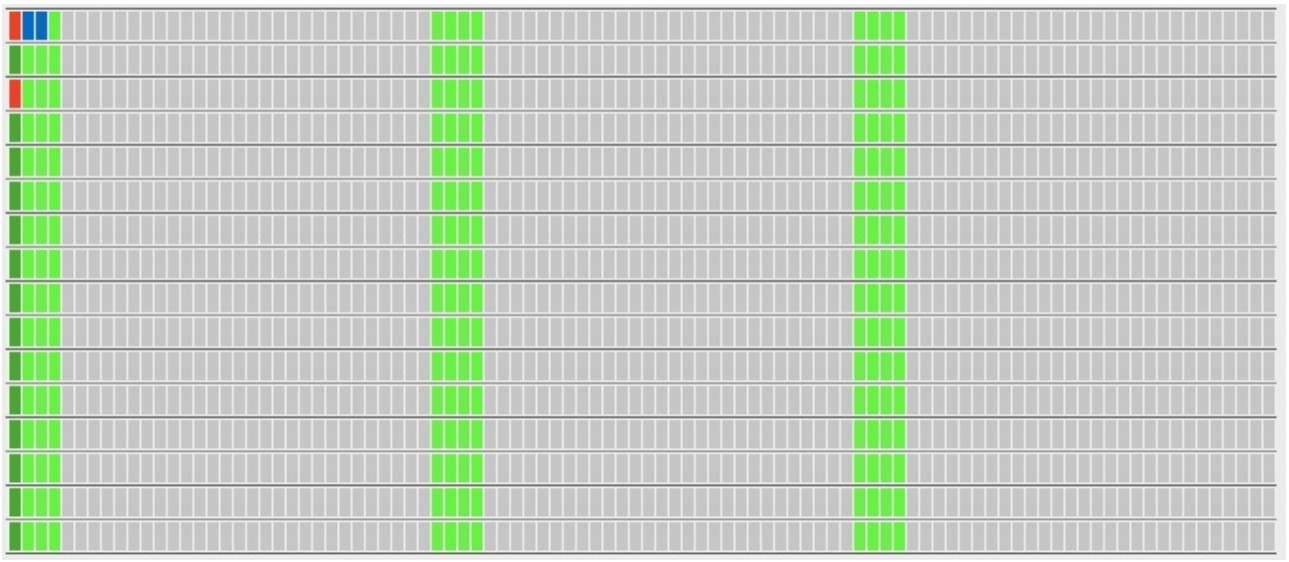
2.2.1.7. FrameSync Error

This event adds one or more successive frame sync errors. It is based on the more general data line error insertion method. There are two parameters to define:

- **Repeat** is the number frame sync symbols that will be corrupted in the superframe. The first error is always located at the beginning of the superframe. The value range from 1 to 8.
- **Sync Symbol Value** is the new value given to the sync symbol. The default value for the frame sync symbol is 0b1011 (11 decimal or 0xB). Any other values will corrupt the frame synchronization.

▶ 2	**Superframe Begin**	1,SM=19,CG=10,RF=6,12,Auto
▼ 3	**FrameSync Error**	2,6
	Repeat	2
	Sync symbol value	6

In this example, the two first frame sync symbols are corrupted (value equal to 6 instead of 11). The red slots are sowing the position of the corrupted frame sync symbols.



XML Translation:

The XML container is *<Error>*, the XML tag is *<EData>* and the parameters are *StartSlot*, *EndSlot* and *Val*.

```
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="6" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : FrameSync Error -->
    <Error>
        <EData StartSlot="0" EndSlot="0" Val="6" />
        <EData StartSlot="192" EndSlot="192" Val="6" />
    </Error>

</Superframe>
```

2.2.1.8. Clock Line Error

This event writes an arbitrary value in one or more consecutive slots. As it is happening outside any protocol, it will be used to generate errors on the data line wherever the user needs. There are three parameters to define:

- **From Slot** is the slot number in the superframe of the first slot where data will get assigned a specific value. The value range from 0 to 1535.
- **To Slot** is the slot number in the superframe of the last slot where data will get assigned a specific value. The value range from 0 to 1535.
- **Pattern Value** is the value written in the slots clock pattern comprised between “From Slot” to “To Slot”. the value ranges from 0 to 15. A value 0 will mean that all 4 clock pulses are swallowed. A value 15 will mean that all 4 clock cycles are present. A value 11 (0b1011) will mean that the second clock cycle of the slot is swallowed.

▶ 2	**Superframe Begin**	1,SM=19,CG=10,RF=6,12,Auto
▼ 3	**Clock Line Error**	60,60,7
	From slot	60
	To slot	60
	Pattern value	7

XML Translation:

The XML container is *<Error>*, the XML tag is *<EClk>* and the parameters are *StartSlot*, *EndSlot* and *Val*.

```
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="6" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : Clock Line Error -->
    <Error>
        <EClk StartSlot="60" EndSlot="60" Val="7" />
    </Error>

</Superframe>
```

2.2.1.9. Loops

Use the Loops to repeat part of a script when using the hardware to stream the SLIMbus traffic. Loops will not have any effect when generating a VCD file and shall be avoided. An integer number of superframes shall be included in the loop container. Looping part of a script is an easy way to achieve very long streams. When streaming data in a data channel, loops can achieve never-ending data streaming.

To specify which part of the script must be looped, insert a “Loop Begin“ event, place as many “Superframe Begin” events as desired and close the loop section by adding a “Loop End” event.

Loops will cause the hardware traffic generator to iterate a given number of time a part of the script stored in the internal memory of the hardware. Clever use of loops can significantly reduce the time needed by the computer to generate a script. When streaming with the hardware, use loops instead of the Repeat function of the superframe. Nested loops are not allowed. But many sequential loops can exist in a script.

The only parameter of “Loop Begin” is **Repeat**. It can take any value between 1 and 32767. Any value larger than 32767 but smaller than 65536 will cause the hardware to loop forever, resulting in a never ending streaming. This feature is especially useful when very long streaming is necessary to run audio tests, for instance.

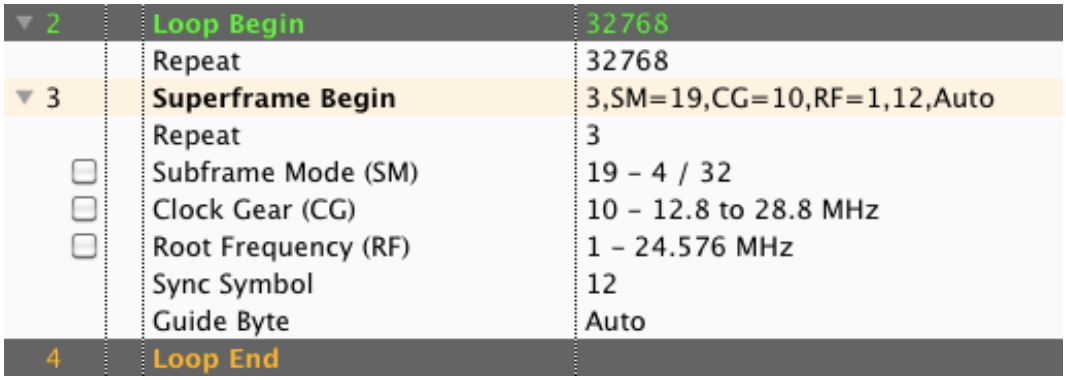
Loops and superframe repeat can be cleverly combined to achieve sine wave generation.

Distortionless sine wave generation requires the sine wave frequency to obey the following rule:

$$F_{Sine}=F_{Superframe} \times M / N$$

Where **M** is the number of times the sine wave repeats over **N** superframes.

If the sine wave period extends over 3 superframes (M=1, N=3), for instance, the Repeat parameter of the superframe tag will be set to 3, to allow the Traffic Generator to build a complete period and that period will be repeated as many times as necessary by the hardware by using a loop.



Loops Limitations

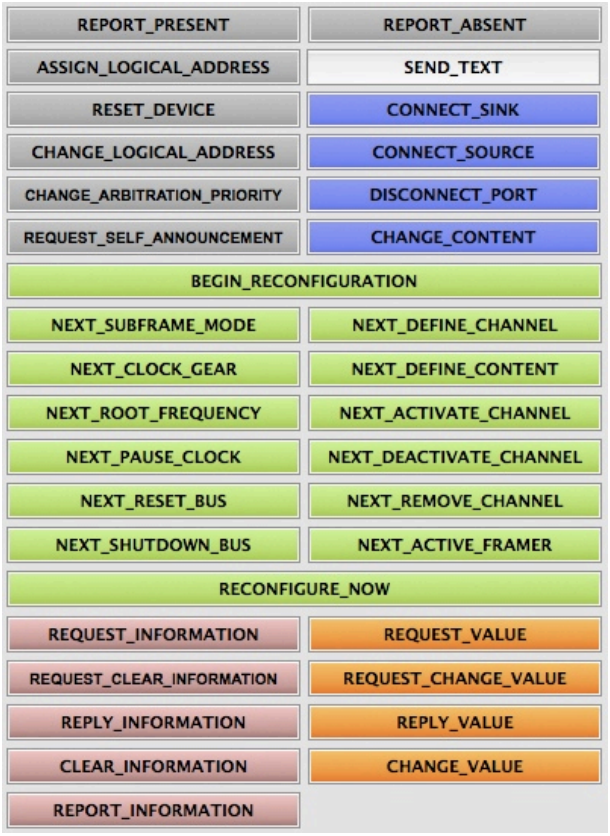
As the loops are repeating the content of N superframes, SLIMbus events that are spanning over more than N superframes will be corrupted. This is typically the case for the whitening stream that will not comply to the specification anymore. The Phasing stream will also broadcast erroneous values.

XML Translation:

```
<!-- Event #2 : Loop Begin -->
<Loop Repeat="32768" >

<!-- Event #3 : Superframe Begin -->
<Superframe Repeat="3" >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

</Superframe>

<!-- Event #4 : Loop End -->
</Loop>
```

2.2.2. Core Messages

The left panel shows all the SLIMbus Core messages. There are 34 messages defined, each with their own set of parameters.



The panel is split in five groups:
- Device Enumeration messages
- Data Port management messages
- Bus reconfiguration messages
- Information Elements messages
- Value Element messages.

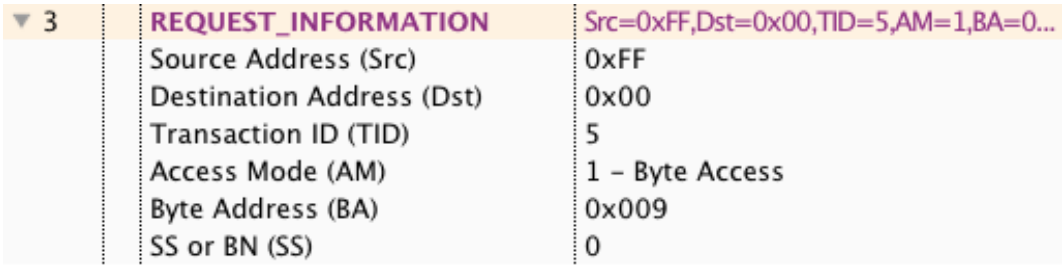
And additional User defined message is added in this list: SEND_TEXT.

This message has been defined by LnK and is recognized by the Protocol Analyzer which will display a text in the core of the decoded stream. It is useful to indicate what happens or what is expected to happen.

Refer to the SLIMbus specification for the exact meaning of the message parameters. ScriptBuilder tries whenever possible to respect the syntax described in the specification. The Core messages are described in the Section 11 of the SLIMbus specification.

Some messages needs additional parameters like the address of the sender (source address) or the address of the destination (destination address).

The messages dealing with Information Elements do not use the Element Code (EC) but rather its components: Access Mode, Byte Address and Slice Size. It is easier and more intuitive to use. But is will be converted to Element Code in the XML File.



For each message, ScriptBuilder will build a structure enclosed in the XML *<SMC>* container. There are 5 distinct entries in the structure:
- Arbitration
- Header
- Message name and parameters
- Message Integrity CRC
- Message Response

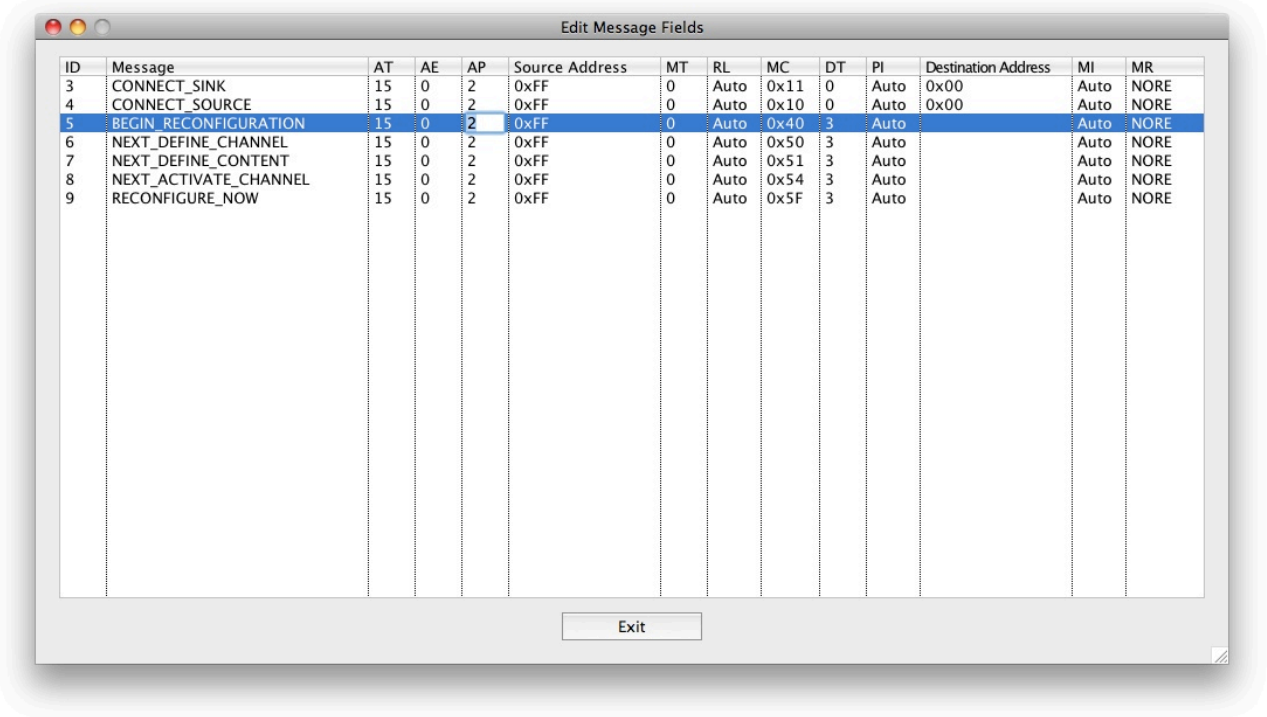
The parameters accessible int he event list will mainly affect the “Message name and parameters” entry. When the source address is required, it will affect the ‘Arbitration’ entry When the destination address is required, it will affect the “Header” entry

Note: the message code and message type are given in the “Header” entry. The message “Id” fond in the “Message name and parameters” entry is a duplicated information, the text name of the message instead of the message code. It is meant to improve readability of the XML scripts by human eyes. Obviously the Message code and Message ID must match.

XML Translation:

```
<Superframe Repeat="3" >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="1" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" />

    <!-- Event #3 : REQUEST_INFORMATION -->
    <SMC>
        <Arbitration ArbitrationType="15" PriorityCode="2" SrcAddress="0xFF" />
        <Header MessageType="0" RemainingLen="Auto" MessageCode="0x20"
                DstType="0" HeaderIntegrity="Auto" DstAddress="0x00"/>
        <Msg Id="REQUEST_INFORMATION" TID="5" EC="152" />
        <Integrity="Auto" />
        <Response="NORE" />
    </SMC>

</Superframe>
```

ScriptBuilder offers to the user a mean to modify the header parameters of the messages. Go to the menu “Edit/Message Fields”. Each messages that are listed in the sequence of event will appear in a dedicated window.



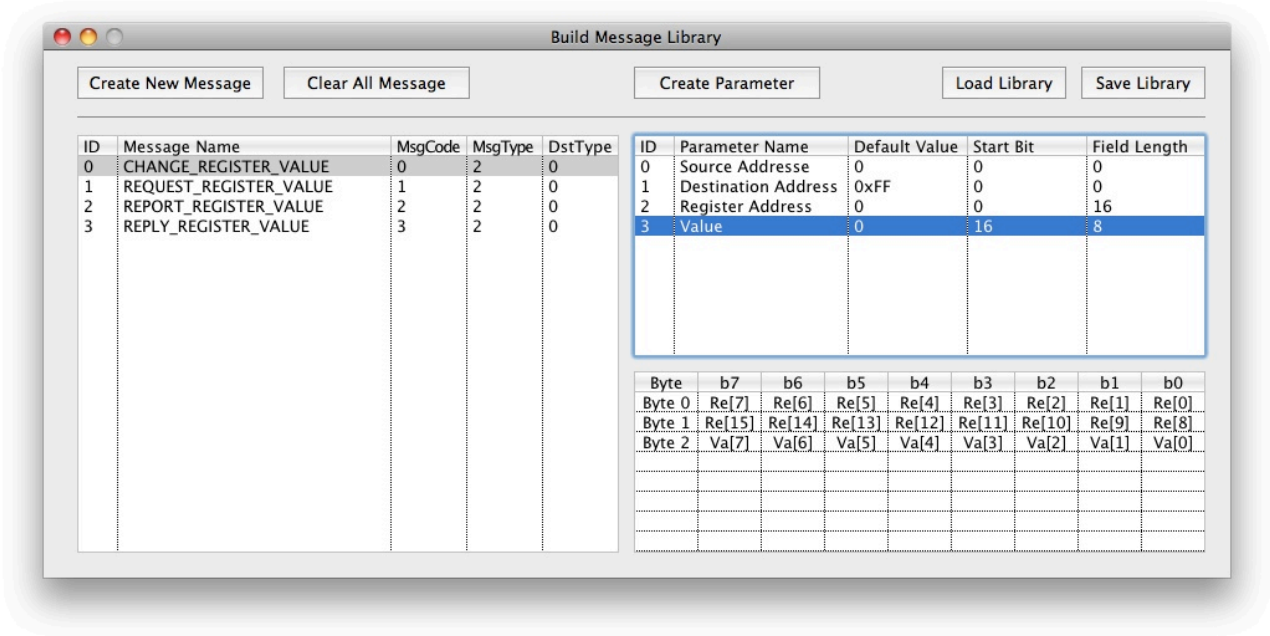
ID	Message	AT	AE	AP	Source Address	MT	RL	MC	DT	PI	Destination Address	MI	MR
3	CONNECT_SINK	15	0	2	0xFF	0	Auto	0x11	0	Auto	0x00	Auto	NORE
4	CONNECT_SOURCE	15	0	2	0xFF	0	Auto	0x10	0	Auto	0x00	Auto	NORE
5	BEGIN_RECONFIGURATION	15	0	2	0xFF	0	Auto	0x40	3	Auto		Auto	NORE
6	NEXT_DEFINE_CHANNEL	15	0	2	0xFF	0	Auto	0x50	3	Auto		Auto	NORE
7	NEXT_DEFINE_CONTENT	15	0	2	0xFF	0	Auto	0x51	3	Auto		Auto	NORE
8	NEXT_ACTIVATE_CHANNEL	15	0	2	0xFF	0	Auto	0x54	3	Auto		Auto	NORE
9	RECONFIGURE_NOW	15	0	2	0xFF	0	Auto	0x5F	3	Auto		Auto	NORE

All the header parameters are now editable at the exception of the Message Code (for obvious reasons). They will be translated in the XML script.

2.2.3. User Defined Messages

The SLIMbus specification allows the use of user defined messages. Obviously, as these messages are user defined, ScriptBuilder cannot have any “a priori” knowledge of them. Therefore, a message library editor has been added to the tool to allow the use of user defined messages.

Go to the menu “**Tools/Message Builder**”. A new window will appear.



The message list is on the left side. On the right, one can find the parameter list (associated to the selected message) and the bit organization of the parameters in the Message payload.
Click on the button “**Create New Message**” to add a message in the list. The message name, code and Type can be edited. The Destination Type is also editable. Some messages will be Broadcast while others are meant to be unicast.

Once the message is defined, add parameters by clicking on the “**Create Parameters**” button. A new parameter is added in the list. A maximum of 10 parameters per messages is allowed.
Edit the parameter by clicking on the column of the Parameter Name, Default Value, Start Bit and Field Length.

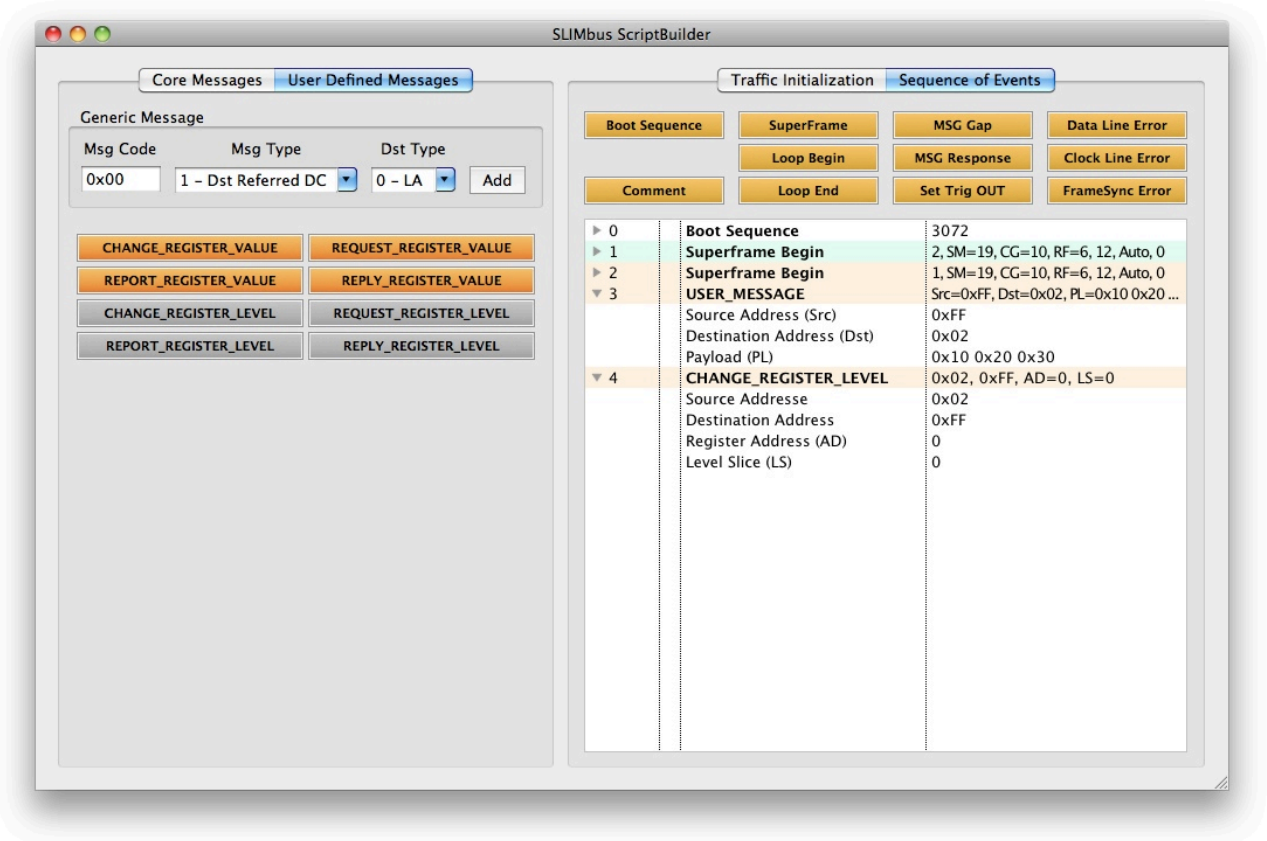
Start Bit indicates where in the payload the least significant bit (LSb) of the parameter will be located. **Field length** indicates how many bits are used for the parameter. As for all the message payloads, the parameters are stored most significant bit first, least significant byte first. The **Default Value** is the value that is used for the parameter when the message is inserted in the sequence of event.
To delete a message or a parameter, click on its **ID** and hit the “Del” key.

Note: For the messages that can be transmitted by any devices and that can have as a destination any devices, it is advised to add as the first parameter “Source Address” and “Destination Address” as second parameter. Set the **Field Length** to 0 and the parameter will not be included in the Message payload.

If the Source and Destination addresses are not included in the message parameter list, the User will have to use the Message Fields edit window to access the addresses.
In order to allow building consequent messages libraries, it is possible to merge existing libraries. Clicking on the “Load Library” button will not delete the messages already present in the list. The existing messages and the one of the loaded library will all appear in the list. The list can then be saved as a new library (with a .lib extension).

To be able to use the newly created message library, it mist first be saved as a message library file. Click on the “**Save Library**” button to do this operation.

Once this is done, the library can be loaded in the User Define message panel. Go to menu “**File/Import Message Library**” and select the desired library. The messages will be loaded and they will appear in buttons, just like the Core messages. Click on the desired message to insert it in the sequence of events.



For fast script generation, it is also possible to insert directly a generic USER_MESSAGE. The Message Code, Message Type and Destination Type must be defined before pressing the “**Add**” button. Then, in the Event list, the source address, the destination address and the payload content can be defined. Note that the payload is given in the form a series of hexadecimal bytes: for instance “0x10 0x20 0x30”.

XML Translation:

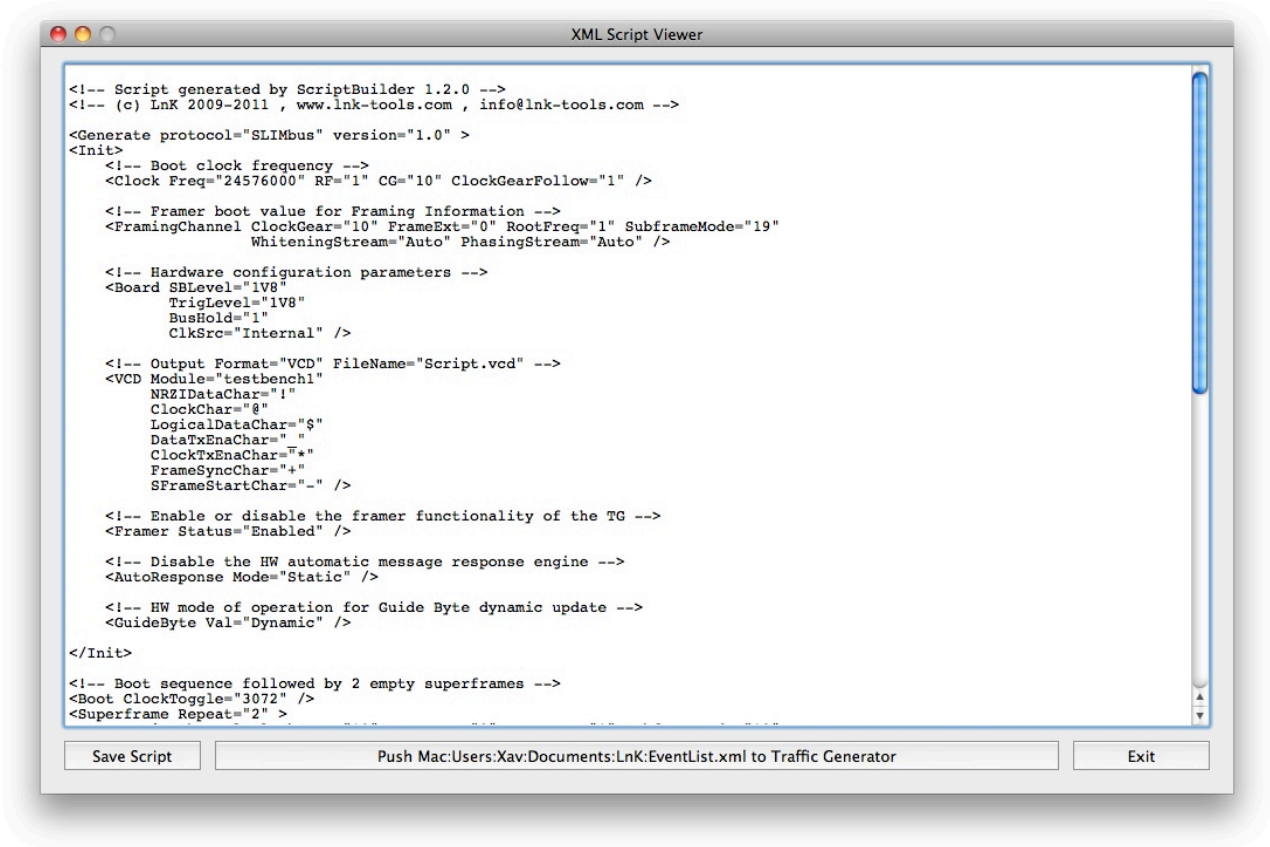
The User Defined messages are treated a bit differently than the Core messages. As the parameter names and definition are not known by the analyzer, the *<msg>* tag is replaced by the *<Payload>* tag with the parameter *Data*. ScriptBuilder generates the *Data* field out of the parameter list. In addition, but commented and thus not used, the *<msg>* tag is written for readability of the XML code.

The two user defined messages header parameters can be editied in the exact same way as the Core messages by using the Edit Message Field window.

```
<!-- Event #2 : Superframe Begin -->
<Superframe >
    <FramingChannel ClockGear="10" FrameExt="0" RootFreq="6" SubframeMode="19"
                    WhiteningStream="Auto" PhasingStream="Auto" Reserved="0" />

    <!-- Event #3 : USER_MESSAGE -->
    <SMC>
        <Arbitration ArbitrationType="15" PriorityCode="2" SrcAddress="0xFF" />
        <Header MessageType="1" RemainingLen="Auto" MessageCode="0x00"
                DstType="0" HeaderIntegrity="Auto" DstAddress="0x02"/>
        <!-- Msg Id="USER_MESSAGE" -->
        <Payload Data="0x10 0x20 0x30" />
        <Integrity="Auto" />
        <Response="NORE" />
    </SMC>

    <!-- Event #4 : CHANGE_REGISTER_LEVEL -->
    <SMC>
        <Arbitration ArbitrationType="15" PriorityCode="2" SrcAddress="0x02" />
        <Header MessageType="1" RemainingLen="Auto" MessageCode="0"
                DstType="0" HeaderIntegrity="Auto" Reserved="0" DstAddress="0xFF"/>
          <!-- Msg Id="CHANGE_REGISTER_LEVEL" Register Address (AD)="0" Level Slice (LS)
="0" -->
        <Payload Data="0x00 0x00 0x00" />
        <Integrity="Auto" />
        <Response="NORE" />
    </SMC>

</Superframe>
```

2.3. XML Script Generation

Once the script is ready to be used, it must be first entirely translated to the XML format. Go to menu “File/Export XML Script” to do this operation.
A new window will appear with the XML code that has been generated.



The text area of that window allows for last minute text edition. Once the XML script is ready, save it by clicking on the “**Save Script**” button or send it directly to the Traffic Generator by clicking on the “**Push *Filename* to Traffic Generator**” button. If the script does not get a name, it will be saved as “Default.xml” in the c:/sba/tmp directory and then will be pushed to the Traffic Generator.

Note that it is possible to bypass the XML Script Viewer window by using the menu “**File/ Send Script to Traffic Generator**”. In this case, the XML script will automatically be saved as “Default.xml” in the c:/sba/tmp directory and will be pushed to the Traffic Generator.

3. Special Features

3.1. Device Register Configuration

Most of the devices have a register map that allows for the configuration of the device functionalities. In case of large number of registers to configure, it may be unpractical to generate the script manually. ScriptBuilder has a special tool to turn a csv file into SLIMbus messages.

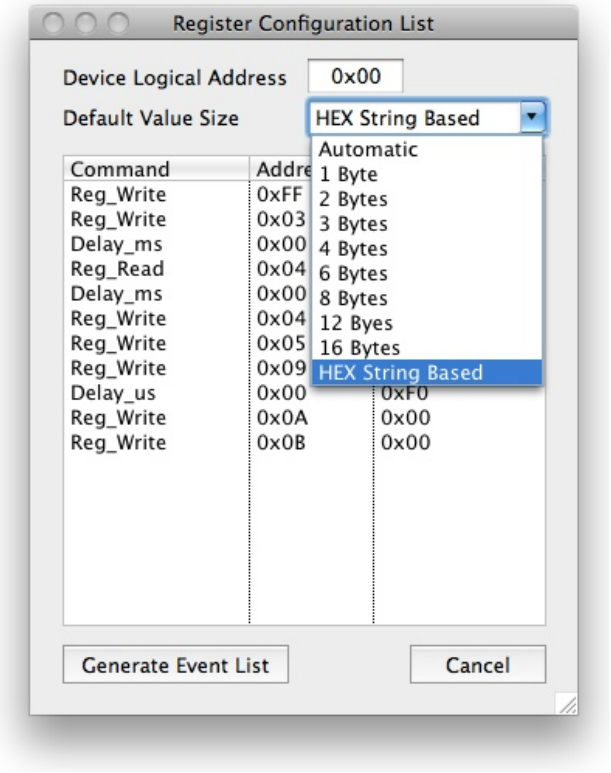
Go to menu "**File/Import Register Configuration**". Select a CSV (ASCII comma separated value) file that has the following format.
- The comment lines are indicated by a semicolon at the start of the line.
- The parameters are sequenced "Function, Address, Data, Comment".
- The functions are "**Reg_Write**", "**Reg_Read**", "**Delay_ms**" and "**Delay_us**".

"**Address**" is the register address to be accessed.
"**Data**" is the value to be written in the register or the delay to be applied.

The delay function do not use the "Address" field. Set to 0.
Values can be given both in HEX format (0x...) or in decimal format.
The import tool is not case sensitive.

File format example:

```
; Mode = Sample,,,
; Function,Address,Data,Core
Reg_Write,0xFF,0x08,SLIMbus
Reg_Write,0x03,0x22,SLIMbus
Delay_ms,0x00,30000,SLIMbus
Reg_Read,0x04,0x00,SLIMbus
Delay_ms,0x00,0x03,SLIMbus
Reg_Write,0x04,0x8E,SLIMbus
Reg_Write,0x05,0xA0,SLIMbus
Reg_Write,0x09,0x00,SLIMbus
Delay_us,0x00,0xF0,SLIMbus
Reg_Write,0x0A,0x00,SLIMbus
Reg_Write,0x0B,0x00,SLIMbus
```



Be sure to enter the proper Logical Address of the device that needs register configuration. The Value Size can also be specified. When set to "Automatic", the Value Size is computed based on the actual given value. If this behavior is not desired, the Value Size can be forced to any of the allowed sizes. The "**HEX String Based**" size option will follow the size of the hexadecimal string. 0x0000 will get a Value Size of 2 bytes. 0x000000 will get a Value Size of 3 bytes. The Value Size is not based on the actual value anymore.

If the imported csv file is OK, just click on the "**Generate Event List**" button and all the new messages will get appended to the existing event list.

The delays are either generated by message gaps or empty superframes. ScriptBuilder is adjusting delays to the closest possible value.

In the following example, the 30000 ms delay (30s) is turned into a combination of loops and superframe repeat. Indeed, at 24MHz, the duration of 1 superframe is equal to 256us. To reach 30s, we would need 117187,5 superframes. This value is bigger than the loop repeat limit (32767), therfore ScripBuilder mixes Loop repeat and superframe repeat to reach the desired 117187.5 superframes.

The 3 ms delay is simply turned into 12 empty superframes.

The 240us delay is turned into a message gap that will effectively insert a delay of 240us between the 2 messages.

0	Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
1	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0xFF,SS...
2	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x03,SS...
3	Loop Begin	29298
4	Superframe Begin	4,SM=19,CG=10,RF=6,12,Auto
5	Loop End	
6	Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
7	REQUEST_VALUE	Src=0xFF,Dst=0x00,TID=3,AM=1,BA=0...
8	Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
9	Superframe Begin	12,SM=19,CG=10,RF=6,12,Auto
10	Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
11	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x04,SS...
12	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x05,SS...
13	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x09,SS...
14	Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
15	Message Channel Gap	68
16	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x0A,SS...
17	CHANGE_VALUE	Src=0xFF,Dst=0x00,AM=1,BA=0x0B,SS...

By default after a REQUEST_VALUE message, an empty superframe is added to give some time to the device to answer. If one superframe is not enough, just add a Delay_x command with the desired duration.

As the event list is appended to the existing script, make sure that there is at least one superframe event in the list before importing the register configuration messages.

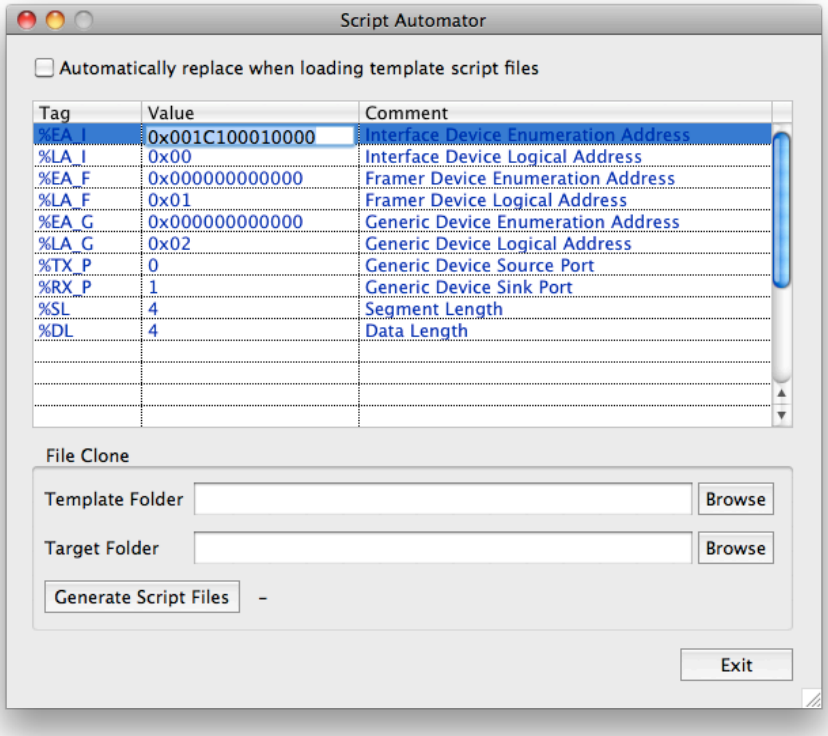
Important note:
When importing large register configuration files (a thousand or more registers to configure), the generation time can be significant: many tens of seconds to many minutes during which the software will not be responding. This is due to the way the ScriptBuilder simulator operates to make sure the script is error free. The conversion to XML of many thousands of message is also a time consuming task. ScriptBuilder is not optimized for this kind of use case. This task is much more efficiently handled by our SLIMbus Audio Bridge and its software.

3.2. Script Automator

It's possible to write script templates with parameters having literal expressions instead of numerical values. The literal expression are starting by the character "%".

0		Boot Sequence	3072
1		Superframe Begin	2, SM=19, CG=9, RF=1, 12, Auto, 0
2		Comment	Empty Superframes for the REPORT_PRESENT messages
3		Superframe Begin	4, SM=19, CG=9, RF=1, 12, Auto, 0
4		Comment	Logical Address Assignment
5		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto, 0
6		ASSIGN_LOGICAL_ADDRESS	Dst=%EA_I, LA=%LA_I
7		ASSIGN_LOGICAL_ADDRESS	Dst=%EA_G, LA=%LA_G
8		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto, 0
9		CONNECT_SINK	Src=0xFF, Dst=%LA_G, CN=0, PN=%RX_P
10		Loop Begin	200
11		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto, 0
12		Loop End	
13		Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto, 0
14		BEGIN_RECONFIGURATION	
15		- NEXT_DEFINE_CHANNEL	CN=0, SD=3140, TP=0, SL=4
16		- NEXT_DEFINE_CONTENT	CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4
17		- NEXT_ACTIVATE_CHANNEL	CN=0
18		RECONFIGURE_NOW	
19	R	Superframe Begin	1, SM=19, CG=9, RF=1, 12, Auto, 0
20		Loop Begin	50000
21		Superframe Begin	2, SM=19, CG=9, RF=1, 12, Auto, 0
22		Loop End	

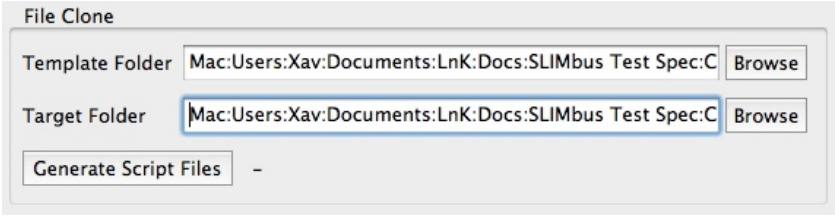
In this example, %EA_I is used to represent the Interface Device Enumeration Address. %LA_I represents the Interface Device Logical Address.
There are 10 predefined strings. As such, a script will not be accepted by the Traffic Generator. Go the the menu **Tools / Script Automator** to open the control window.



The string can get assigned a given value. The user can create up to 10 new strings on top of the existing ones (colored in dark blue).

When the check box is ticked, ScriptBuilder will automatically replace the strings by the define values when opening the script.

It is also possible to use the “File Clone” function and create Component specific scripts out of a template script folder.



Once the template folder and the target folder are identified, click on the “Generate Script Files” button and wait for the file copy to be over. The target directory will contain the same files than the template directory, but with the string replaced by the values given in the table.

Note that the file clone tool will use any text files present in the template directory: .lst (native ScriptBuilder format), .xml (Traffic Generator input file), .slb (Audio Bridge script library), ...

3.3. SLIMbus Protocol Checker

ScriptBuilder offers many levels of verification.
Message parameter checks
- Framing Information consistency check
- Message protocol check (Reconfiguration Sequence)
- Message channel bandwidth overflow

The main purpose of the tool is to allow the user to generate complex scripts that are fully compliant to the SLIMbus specification. Non compliant behavior can be manually inserted. Any deviation to compliancy will be flagged with warnings or errors.

3.3.1. Message parameter check

When an erroneous value is given in a message parameter, an error flag appears in the second column under the form of a red capital "**E**".
When a Message gets specified a "NACK" message response, ScriptBuilder will either mark it with an exclamation point "**!**" for the messages not involved in bus reconfiguration and with a capital "**N**" for the bus reconfiguration messages.
Use the "Message Fields" window to modify these parameters. The parameter window is automatically refreshed when new messages are added.

3.3.2. Framing Information consistency check

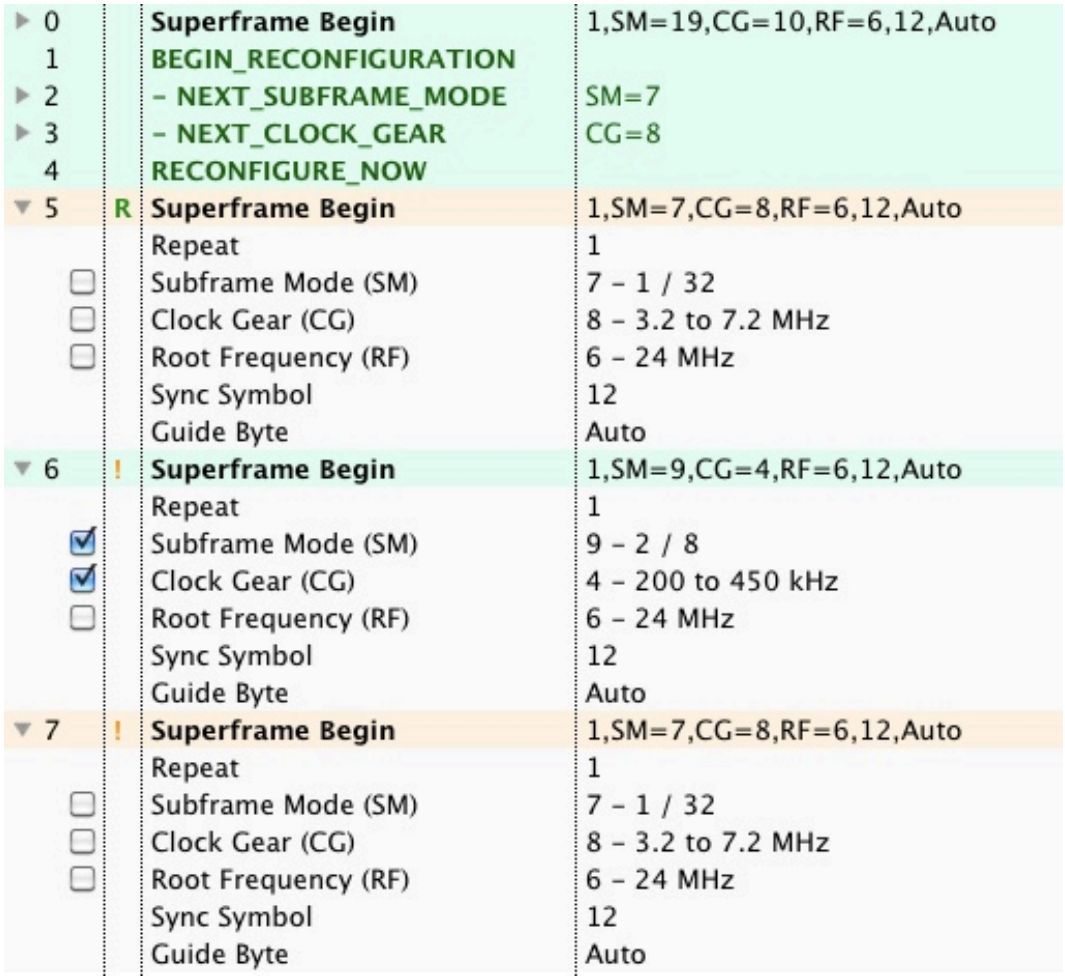
When inserted in the event list, a "Superframe Begin" event always gets the right framing information values. ScriptBuilder will track all the reconfiguration messages and will always make sure that the framing information is valid.

A green capital "**R**" in front the "Superframe Begin" event will indicate a Reconfiguration Boundary.

The following example changes the Subframe Mode and the Clock Gear. Have a look to the SM and CG values of the superframes following the Reconfiguration Boundary.

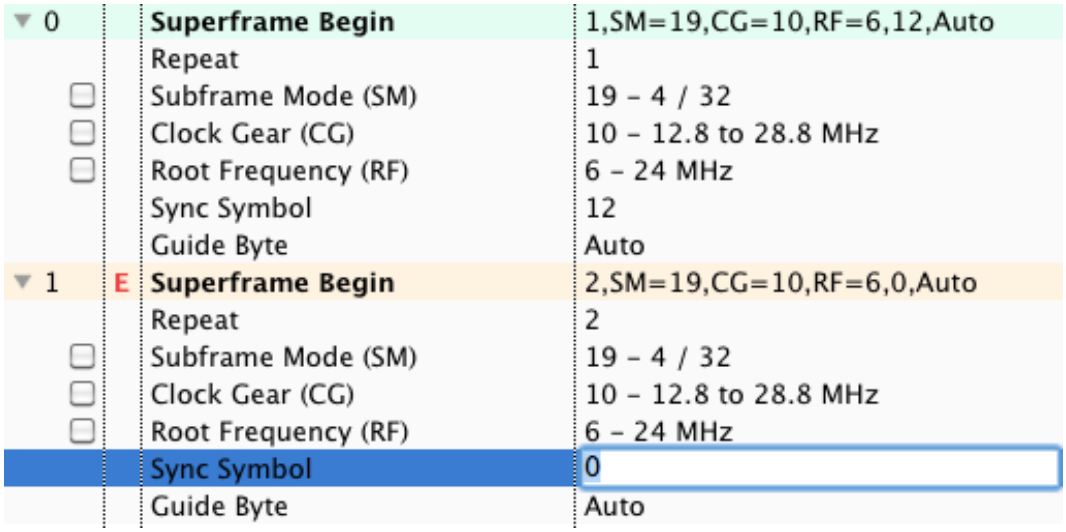
2		Superframe Begin	1,SM=19,CG=10,RF=6,12,Auto
3		BEGIN_RECONFIGURATION	
4		- NEXT_SUBFRAME_MODE	SM=11
5		- NEXT_CLOCK_GEAR	CG=6
6		RECONFIGURE_NOW	
7	R	Superframe Begin	1,SM=11,CG=6,RF=6,12,Auto
8		Superframe Begin	1,SM=11,CG=6,RF=6,12,Auto

To deviate from the normal behavior, the user can click on the check box in front of the Framing parameters (subframe mode, clock gear, root frequency) and enter its own value. When the check box is set, the automatic value update is deactivated for that particular superframe.



0		**Superframe Begin**	1,SM=19,CG=10,RF=6,12,Auto
1		**BEGIN_RECONFIGURATION**	
2		**- NEXT_SUBFRAME_MODE**	SM=7
3		**- NEXT_CLOCK_GEAR**	CG=8
4		**RECONFIGURE_NOW**	
5	R	**Superframe Begin**	1,SM=7,CG=8,RF=6,12,Auto
		Repeat	1
☐		Subframe Mode (SM)	7 - 1 / 32
☐		Clock Gear (CG)	8 - 3.2 to 7.2 MHz
☐		Root Frequency (RF)	6 - 24 MHz
		Sync Symbol	12
		Guide Byte	Auto
6	!	**Superframe Begin**	1,SM=9,CG=4,RF=6,12,Auto
		Repeat	1
☑		Subframe Mode (SM)	9 - 2 / 8
☑		Clock Gear (CG)	4 - 200 to 450 kHz
☐		Root Frequency (RF)	6 - 24 MHz
		Sync Symbol	12
		Guide Byte	Auto
7	!	**Superframe Begin**	1,SM=7,CG=8,RF=6,12,Auto
		Repeat	1
☐		Subframe Mode (SM)	7 - 1 / 32
☐		Clock Gear (CG)	8 - 3.2 to 7.2 MHz
☐		Root Frequency (RF)	6 - 24 MHz
		Sync Symbol	12
		Guide Byte	Auto

The superframe sync symbol can also be corrupted., leading to a warning message (if there is only one superframe corrupted) or an error message (if two or more consecutive superframes are corrupted).



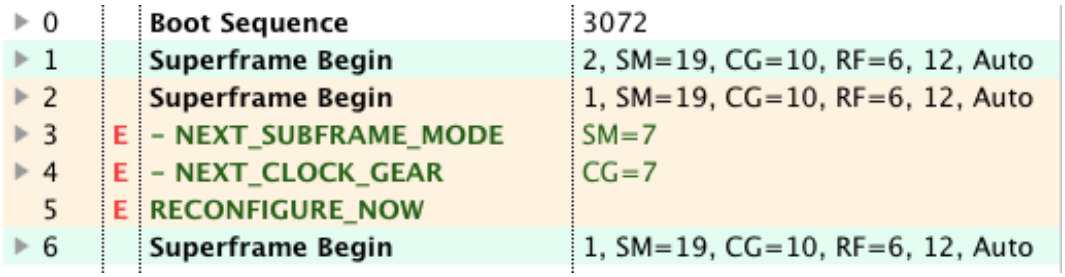
0		**Superframe Begin**	1,SM=19,CG=10,RF=6,12,Auto
		Repeat	1
☐		Subframe Mode (SM)	19 - 4 / 32
☐		Clock Gear (CG)	10 - 12.8 to 28.8 MHz
☐		Root Frequency (RF)	6 - 24 MHz
		Sync Symbol	12
		Guide Byte	Auto
1	E	**Superframe Begin**	2,SM=19,CG=10,RF=6,0,Auto
		Repeat	2
☐		Subframe Mode (SM)	19 - 4 / 32
☐		Clock Gear (CG)	10 - 12.8 to 28.8 MHz
☐		Root Frequency (RF)	6 - 24 MHz
		Sync Symbol	0
		Guide Byte	Auto

3.3.3. Message protocol check

The Reconfiguration Sequences are all tracked and applied when possible. When a Reconfiguration Boundary happens, a capital green "**R**" is displayed in front of a "Superframe Begin" event, telling the user that changes are going to take effect in that particular superframe.

When a Reconfiguration sequence is illegal or not properly built, the faulty event is marked with a capital red "**E**".

In the following example, there is a missing BEGIN_RECONFIGURATION message.



0		**Boot Sequence**	3072
1		**Superframe Begin**	2, SM=19, CG=10, RF=6, 12, Auto
2		**Superframe Begin**	1, SM=19, CG=10, RF=6, 12, Auto
3	E	**- NEXT_SUBFRAME_MODE**	SM=7
4	E	**- NEXT_CLOCK_GEAR**	CG=7
5	E	**RECONFIGURE_NOW**	
6		**Superframe Begin**	1, SM=19, CG=10, RF=6, 12, Auto

Additionally, if a reconfiguration sequence is not valid, the Reconfiguration Boundary will not happen (no "**R**" displayed). The announced changes are discarded.

When scripts are very long, the protocol check takes time as the complete script needs to be screened every time a change is made. It is therefore possible to disable the protocol automatic check. Go to the menu "**Tools/Disable Protocol Auto Check**". When working in that mode, it is very easy to make mistakes. It is highly advisable to run a protocol check from time to time. Go the the menu "**Tools/Run Protocol Auto Check**". Automatic checking can be re-enabled by going to the menu "**Tools/Enable Protocol Auto Check**".

3.3.4. Message channel capacity overflow

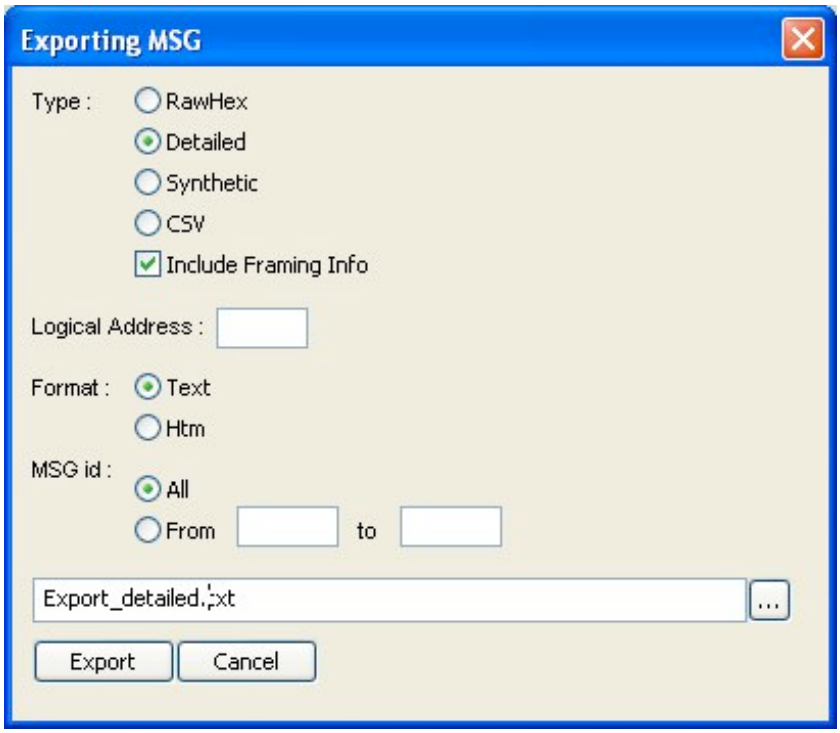
Given a subframe mode, the Message Channel has only a limited (though well defined) capacity to carry messages in one superframe. ScriptBuilder will automatically insert a "Superframe Begin" event when a message is crossing the superframe boundary. Therefore, the user will always have the insurance that its Message Channel traffic will never exceed the capacity of the Message Channel in a superframe.

The overflow check is also performed when the subframe mode parameter of a reconfiguration sequence is modified. New superframes will be inserted wherever necessary. However, Script builder will not delete superfluous superframes when there is an excess of capacity. It will be up to the user to fine tune his scenario.

3.4. Import Message Capture

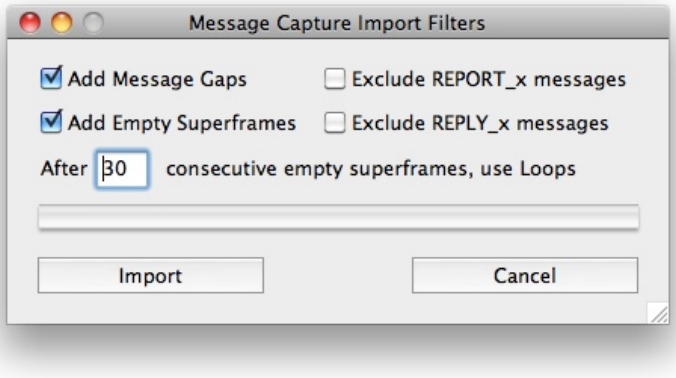
ScriptBuilder has an import tool to convert a captured trace into a script file. This is very convenient to replay and eventually modify a sequence of messages that has been captured by the SLIMbus Protocol Analyzer.

ScriptBuilder accepts the text file generated by the Protocol Analyzer. To generate the right text file, go to the Protocol Analyzer menu "**File/Export Msg**".



Make sure to select "Type: Detailed", "Format: Text" and to include the superframe information. If the file does not have the right format nor all the necessary information, nothing is going to happen...

In ScriptBuilder, got to the menu "File/ImportMessage Capture". Select the desired file. A small control window will appear.



The default setting will attempt to generate a script that will reproduce the message content and absolute timings.

The check box “Add Message gap” will decide if the messages belonging to a given superframe will be concatenated with no gaps between them (Unchecked) or if the absolute timing of the messages in the superframe will be reproduced (Checked).

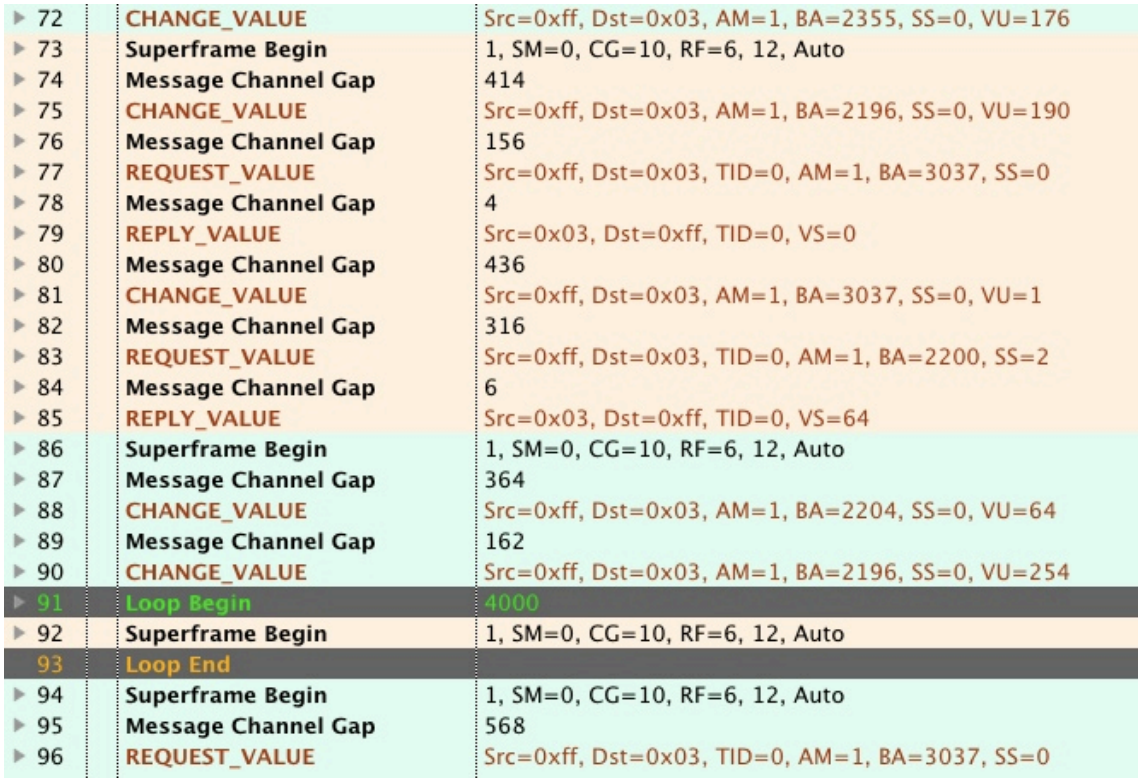
The check box “Add Empty Superframes” will decide if the empty superframes must be added to the script (Checked) or ignored (Unchecked). A comment is inserted to indicate the number of skipped superframes, when this option is selected. Depending of the number of consecutive empty superframes, the tool will use either the “Repeat” parameter of the superframe or the “Repeat” parameter of the loop or eventually both together. The threshold is modifiable by the user. The target of the tool is to use as few superframes as possible in the script in order to ease the job of the Traffic Generator.

It is possible to filter out some of the Core messages involved in interactive transactions:
- REPLY_VALUE and REPLY_INFORMATION
- REPORT_INFORMATION, REPORT_PRESENT and REPORT_ABSENT

The filtered out messages are replaced by a comment. When the message gaps are used, the skipped message is replace by a gap of the corresponding message size to preserve the absolute timings.

The progress bar below the check boxes will indicate the completion of the file analysis.

ScriptBuilder will use a maximum of 15 loops and 1100 superframes. This will dictate the maximum length of the message capture import capabilities.



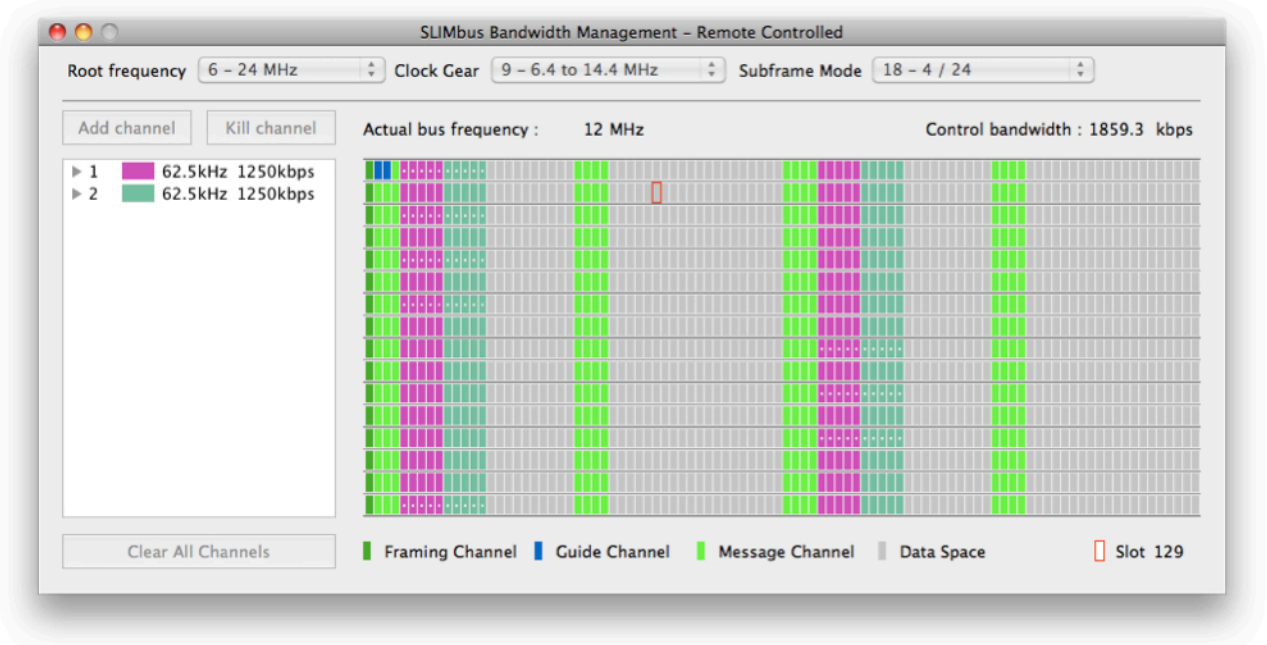
3.5. SLIMbus Bandwidth Management Tool Box

ScriptBuilder can connect to SLIMbusMan (Rev 1.3.0 and later releases) and can use it as a remote display for the superframe organization. Use the Tools menu item “Connect to SLIMbusMan” to create the link. The connection can be verified by looking at the window title bar of SLIMbusMan: “Remote Controlled” is added to the title. To disconnect ScriptBuilder from SLIMbusMan, go to the Tools menu and select “Disconnect from SLIMbusMan”.

SLIMbusMan will display the following information:
- The subframe mode, the clock gear and the root frequency
- The split between control space and data space
- The location of a given message in the Message Channel
- The location a the segments of a data channel.
- The location of the Clock and Data line errors.

The slot layout is organized in 16 half frame, showing a complete superframe at a time. On the left side, all the dark green slots belong to the Framing Channel (Frame sync and Framing Information bits)

When data channels are activated (and displayed), it is possible to modify the channel offset by dragging the first segment of the data channel. The new segment distribution value is automatically fed back into the NEXT_DEFINE_CHANNEL message that is responsible of the status of the segment under change.



In ScriptBuilder, click on a given event and SLIMbusMan will reflect the state of the bus for that given event. Use up and down arrow to easily navigate through the list of events.

4. Script Example

Purpose of the exercise: Build a script to stream two sine waves to the device under test (DUT). The Traffic Generator will play the role of the active Framer.

The first thing to know is the Enumeration address of the DUT. If that information is not known, it is easy to figure out by booting the bus and looking at the REPORT_PRESENT messages.

For the present exercise, we will use the LnK SLIMbus Audio Bridge. This component has 3 logical devices:
- Interface device: EA=0x01C100010000
- Framer Device: EA=0x01C100010100
- Generic Device: EA=0x01C100010200

The Generic Device has 8 bidirectional data ports, numbered from 0 to 7. They all support the ISOC, PUSHED and PULLED protocol.

We will use a bus clock of 24.576 MHz and use Clock Gear 9. The Presence Rate of the audio stream will be 44.1 kHz. That means that we must use the PUSHED or PULLED protocol. However, the Traffic Generator only supports efficiently the PUSHED protocol.

4.1. Script Initialization

The HW configuration is left unchanged. The Framing Information is adapted to our needs. Set the clock Gear to 9 and Root Frequency to 1 (24.576 MHz). HW Guide Byte and Framer inputs are left unchanged.

Add an Automatic Message Response entry to “mute” the Manager response to its own messages.

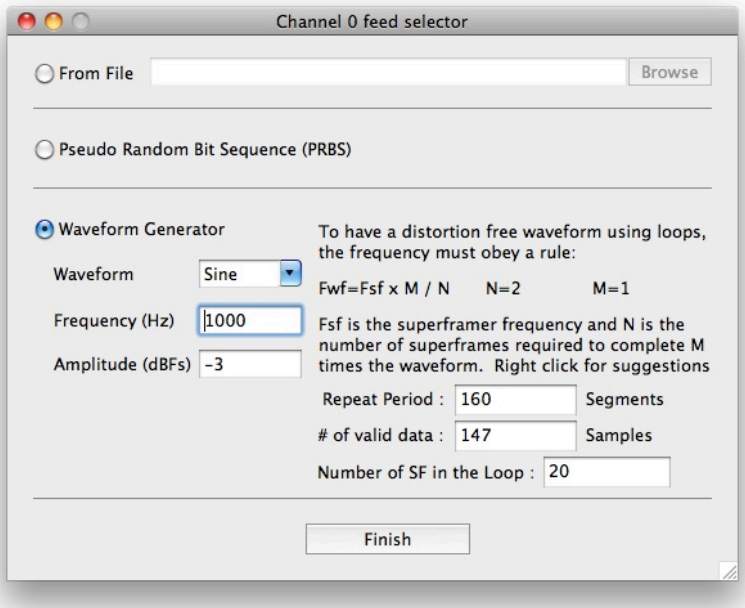
▼ 0	**HW configuration**	
	SLIMbus Level	1V8
	TRIG Level	3V3
	Bus Hold	1 – Enabled
	Clock Source	Internal
▼ 1	**Framing Information**	
	Subframe Mode	19 – 4 / 32
	Clock Gear	9 – 6.4 to 14.4 MHz
	Root Frequency	1 – 24.576 MHz
▼ 2	**HW Guide Byte**	
	Mode	Dynamic
▼ 3	**Framer**	
	Status	Enabled
▼ 4	**Automatic MSG response**	
	Device Address (EA or LA)	0xFF
	Message Type	0xFF – ALL Types
	Message Code	0xFF – ALL MESSAGES
	Response	NORE
	Counter	0

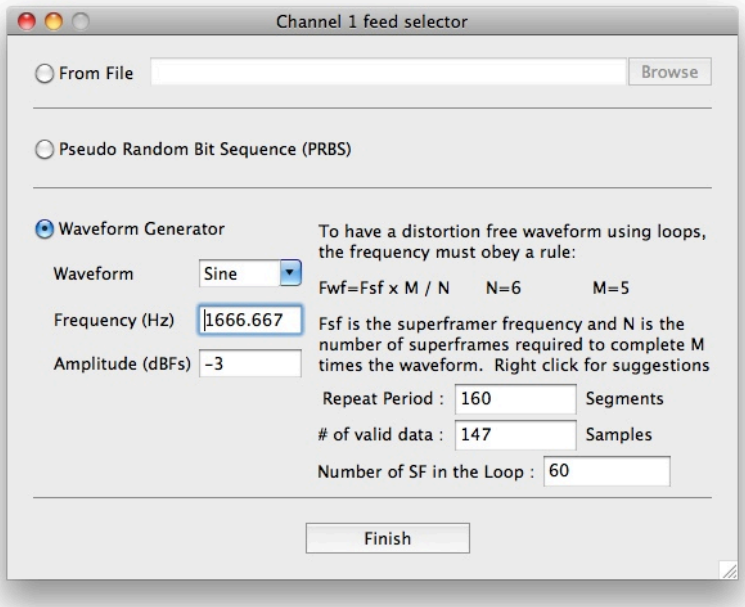
Now, we need to define the audio stream content. We have two channels, each with a different sine wave frequency. So we need two Channel Feed entries.

The sine wave frequencies must be equal to $F_{Sine}=F_{Superframe}$ x M / N, where M is the number of times the sine wave repeats over N superframes.

Fsuperframe = 12.288 MHz/6144 = 2 kHz. So we can chose for 1 kHz (M=1, N=2) and 1.66666666 kHz (M=5, N=6).

Note: In this particular case, we have N1=2 and N2=6. Obviously the loop cannot both repeat at the same time 2 and 6 superframes. The solution is to have the Least Common Multiple of N1 and N2 = 6 superframes in the infinite loop. But this is not going to be sufficient. The Pushed protocol is used so the Presence bit pattern repetition must also be respected.





When taken onto account, the Repeat Period imposes a repetition of 20 superframes for the channel 0 and 60 superframes for the channel 1. To have a clean streaming on both channel, we must use the Least Common Multiple of 20 and 60, which is obviously 60.

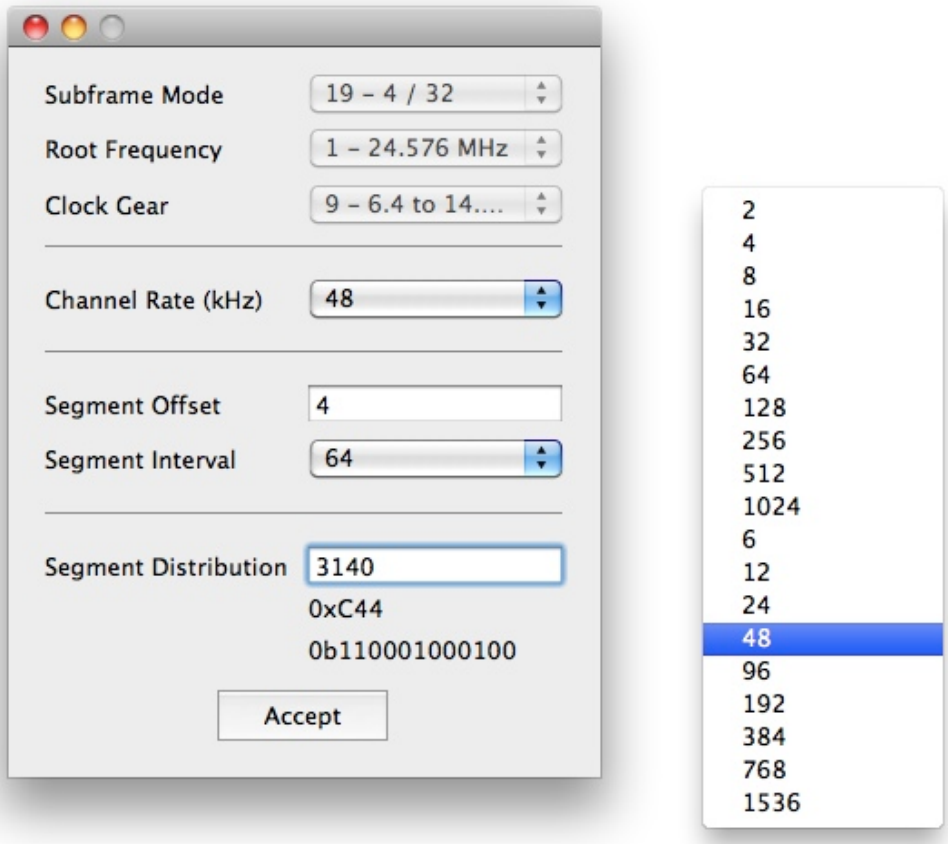
▸ 0	**HW configuration**	
▸ 1	**Framing Information**	
▸ 2	**HW Guide Byte**	
▸ 3	**Framer**	
▸ 4	**Automatic MSG response**	
▾ 5	**Channel Definition**	
	Channel Number	0
	Signal Feed	Sine: 1kHz; -3dBFs
	Segment Distribution	3140
	Segment Length	7
	Transport Protocol	1
	Frequency Locked	1
	Presence Rate	11
	AUX Format	0
	Data Type	1
	Channel Link	0
	DataLength	6
	Sink	TG
▾ 6	**Channel Definition**	
	Channel Number	1
	Signal Feed	Sine: 1,666667kHz; -3dBFs
	Segment Distribution	3147
	Segment Length	7
	Transport Protocol	1
	Frequency Locked	1
	Presence Rate	11
	AUX Format	0
	Data Type	1
	Channel Link	0
	DataLength	6
	Sink	TG

4.2. Script Event List

Now, we need to add all the necessary messages to enumerate and configure the DUT for the audio streaming. As the Traffic Generator is the active framer, a boot sequence must be first added.
Then, give some time for the DUT to transmit its REPORT_PRESENT messages. Finish the enumeration sequence by adding three ASSIGN_LOGICAL_ADDRESS messages.

▸ 0	**Boot Sequence**	3072
▸ 1	**Superframe Begin**	2, SM=19, CG=9, RF=1, 12, Auto
▸ 2	**Superframe Begin**	10, SM=19, CG=9, RF=1, 12, Auto
▸ 3	**Superframe Begin**	1, SM=19, CG=9, RF=1, 12, Auto
▸ 4	**ASSIGN_LOGICAL_ADDRESS**	Dst=0x01C100010000, LA=0
▸ 5	**ASSIGN_LOGICAL_ADDRESS**	Dst=0x01C100010100, LA=1
▸ 6	**ASSIGN_LOGICAL_ADDRESS**	Dst=0x01C100010200, LA=2

The next step is to create the data channels. We need to carry 48 kHz audio samples. Therefore the channel rate must be superior or equal to 48 kHz. The closest possible value is 62.5kHz. To easily compute the value of the segment distribution, right click on the value and start the “Segment Distribution Calculator”.



Pick the desired Channel Rate and enter the Segment Offset. The tool will automatically compute the Segment Distribution value. Click on the “Accept” button to finish.

7	**Superframe Begin**	1, SM=19, CG=9, RF=1, 12, Auto
8	BEGIN_RECONFIGURATION	
9	- NEXT_DEFINE_CHANNEL	CN=0, SD=3140, TP=1, SL=7
	Channel Number (CN)	0
	Segment Distribution (SD)	3140
	Transport Protocol (TP)	1 – Pushed Protocol
	Segment Length (SL)	7
10	- NEXT_DEFINE_CONTENT	CN=0, FL=1, PR=11, AF=0, DT=1, CL=0, DL=6
	Channel Number (CN)	0
	Frequency Locked (FL)	1 – Frequency locked
	Presence Rate (PR)	11 – 44.1kHz
	AUX format (AF)	0 – Not applicable
	Data Type (DT)	1 – LPCM audio
	Channel Link (CL)	0 – Channel not linked
	Data Length (DL)	6
11	- NEXT_ACTIVATE_CHANNEL	CN=0
12	- NEXT_DEFINE_CHANNEL	CN=1, SD=3147, TP=1, SL=7
13	- NEXT_DEFINE_CONTENT	CN=1, FL=1, PR=11, AF=0, DT=1, CL=0, DL=6
14	- NEXT_ACTIVATE_CHANNEL	CN=1
15	RECONFIGURE_NOW	

We need to finish the script by adding the never ending lops at the end to allow for infinite duration data streaming. As we saw already, the loop will need to repeat 60 superframes

to allow for both sine waves to repeat without truncation or sample drop. The loop repeat parameter must be bigger than 32767 to trigger the never ending loop.

▼ 16		**Loop Begin**	65000
		Repeat	65000
▼ 17	R	**Superframe Begin**	60, SM=19, CG=9, RF=1, 12, Auto
		Repeat	60
☐		Subframe Mode (SM)	19 - 4 / 32
☐		Clock Gear (CG)	9 - 6.4 to 14.4 MHz
☐		Root Frequency (RF)	1 - 24.576 MHz
		Sync Symbol	12
		Guide Byte	Auto
▶ 18		**Loop End**	

The final step is to generate the XML script.
It can be saved in a file or directly pushed into the traffic generator software.

5. Remote Control and DLL

ScriptBuilder comes with a DLL for remote control. The DLL is suitable for integration in C ++, Visual Basic and many other Windows programming languages.
The provided files are:
- LnKSLIMbusScriptBuilder_IF.lib
- LnKSLIMbusScriptBuilder_IF.dll
- LnKSLIMbusScriptBuilder_IF.h

Use the .lib and .dll file in your own application.
The header file (.h) is meant to be used in a C++ application and is giving the necessary function prototypes. Read carefully the header file to have a detailed description of the function parameters and the returned values.

The DLL provides all the necessary calls to build automation around ScriptBuilder. the basic functions are file open, file save, file append, register file import, XML generation and export and event list clear all.