SENT Simulator
User's Guide

Author: Laurent Beaurenaut
LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information
For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (www.infineon.com).

Warnings
Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.
We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com
<table>
<thead>
<tr>
<th>Table of Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Scope</td>
<td>5</td>
</tr>
<tr>
<td>2 Needed material</td>
<td>6</td>
</tr>
<tr>
<td>3 Installation steps</td>
<td>7</td>
</tr>
<tr>
<td>4 User’s Interface</td>
<td>8</td>
</tr>
<tr>
<td>5 Simulator’s configuration</td>
<td>10</td>
</tr>
<tr>
<td>6 Code revisions history</td>
<td>11</td>
</tr>
</tbody>
</table>
1 Scope

Dear Reader,

Thanks for using this SENT simulator! This tool will help you to emulate a SENT sensor, in order to help you to develop a SENT driver on your target microcontroller. This document will give you step by step instruction in order to install and operate the simulator.

The aim of this simulator is to generate a SENT compatible signal. For this a TC1796 microcontroller TriBoard is used.

The following features are supported by the simulator:

- Generation of SENT compliant frames.
- Up to 8 data nibbles per frame.
- Easy to use user's interface.
- Serial data generation.
- Automatic and manual mode supported, so that data are read from a predefined table or modified “on the fly”.
- Programmable clock rate.
- Error generation: signal loss, synchronization loss, clock drift, invalid CRC, invalid data nibble, serial data error.
- Full source code available.

Have fun with Infineon's SENT simulator!

Note: Single Edge Nibble Transmission (SENT) refers to the SAE standard J2716. For more information, please visit www.sae.org.

Note: The simulator generates logical SENT frames. Electrical characteristics as defined by the standard are not covered by the simulator.

Note: The code delivered with this application note is aimed at development and demonstration purpose only. Neither is its quality nor its robustness guaranteed.
2 Needed material

The following HW and SW material is mandatory to operate the tool:

HW:

- A PC with local administration rights.
- A functional TC1796 TriBoard (incl. cables, power supplies and adapter board).
- A debugger supporting TC1796.
- A standard oscilloscope.

SW:

- The Simulator source code and executable (included in this package).
- MTTTY, or any terminal window program.
- Optional: MemTool (available on www.infineon.com).
- Optional: Tasking toolchain for TriCore, v2.5.

It is strongly recommended to have the Tasking toolchain installed and running on the PC where the simulator is running, so that the default parameters of the simulator can be changed (recompilation needed). For more information about the toolchain, please visit www.tasking.com.

The Simulator software delivered with this application note is made of:

- C and Header file for the simulator.
- Two option files for Tasking toolchain v2.5 (one for locating the code in the SPRAM, one for locating it in the Flash).
3 Installation steps

It is assumed that the user knows how to operate the different tools mentioned above (Compiler, Debugger, TriBoard, etc.). The following steps may be followed to install operate the simulator:

- Install all the needed SW tools (MemTool, MTTY, Tasking toolchain).
- Connect a TriBoard to the debugger. Connect the PC serial port to the serial port of the TriBoard.
- Connect a Scope to Pin 2.8 (SENT signal) and 2.15 (Trigger).
- In the Compiler toolchain, create a new project and add all the C and H files provided. Load the wanted options (code in RAM or code in Flash). Build the project.
- In case the code is located in the RAM, a debugger session can be started.
- In case the code is to be located in the Flash, it needs to be flashed first (e.g. by using MemTool).
- Open MTTY (default communication settings) and select “connect”.
- Run the program: a SENT signal should be displayed on the scope.
- A message should be displayed on the terminal. The simulator is ready to use!

Note: In case the message does not display, reset the TriBoard and restart the program.
The following keys of the keyboard are used to send commands to the simulator.

```
Error Generation  Data nibble 1..8  Serial data nibble:
value + / - 1     value + / - 1

Used keys
“q”: Increase the first data nibble by 1 in manual mode.
“a”: Decrease the first data nibble by 1 in manual mode.
“w”: Increase the second data nibble by 1 in manual mode.
“s”: Decrease the second data nibble by 1 in manual mode.
“e”: Increase the third data nibble by 1 in manual mode.
“d”: Decrease the third data nibble by 1 in manual mode.
“r”: Increase the fourth data nibble by 1 in manual mode.
“f”: Decrease the fourth data nibble by 1 in manual mode.
“t”: Increase the fifth data nibble by 1 in manual mode.
“g”: Decrease the fifth data nibble by 1 in manual mode.
“z”: Increase the sixth data nibble by 1 in manual mode.
“h”: Decrease the sixth data nibble by 1 in manual mode.
“u”: Increase the seventh data nibble by 1 in manual mode.
“j”: Decrease the seventh data nibble by 1 in manual mode.
“i”: Increase the eightth data nibble by 1 in manual mode.
“k”: Decrease the eightth data nibble by 1 in manual mode.
```

Figure 2  User’s interface.
Simulator’s configuration

“o”: Increase serial data by 1.
“i”: Decrease serial data by 1.
“.”: Increase the clock rate by 1%.
“-“: Increase the clock rate by 1%.
“y”: Signal ON.
“x”: Signal OFF.
“n”: Manual Mode selected (the value of the data nibbles can be modified on the fly).
“m”: Automatic mode selected (the value of the data nibbles is read from a table).

Error keys

“1”: The transmit clock will drift of +2%.
“2”: The transmit clock will drift of -2%.
“3”: Synchronization loss: calibration pulse missing.
“4”: Synchronization loss: calibration pulse too early.
“5”: Frame CRC error generated.
“6”: The last data nibble will be too short (data < 0).
“7”: The last data nibble will be too long (data > 15).
“8”: Serial Data start bit missing.
“9”: Serial Data start bit too early.
“0”: Serial data CRC error generated.

Note: The control keys have been selected to fit with a German keyboard. By modifying the simulator’s source code, it is possible to modify the control keys. This can be done in the function ASC0_viRx() in the file ASC0.c.
5 Simulator’s configuration

The settings that can be modified by the user are “defines” located in the file “Parameters.c”. After a parameter has been modified, the program should be recompiled.

// NEEDS TO BE CONFIGURED BY THE USER
#define NUM_DATA_NIBBLE 6   // Valid range: 1 to 8.
#define SDATA_CRC 0          // 0: the serial data nibble is not part of the frame CRC.
#define SDATA_USED 0          // 0: Serial Data nibble always 0.
#define TCLOCK 5000          // Transmit clock period in ms. Valid range: 3000 to 10000.
#define INIT_CLK_DEV 100      // Initial Clock deviation from nominal frequency.
#define INIT_DATA 0x000000000 // Initial DATA value (Manual Mode).
#define INIT_SDATA 0x00000000 // Initial Serial DATA value (4 bit ID + 1 byte data).
#define USER_DATA_SIZE 3      // Number of elements in the data table (automatic mode).

Figure 3 Default simulator’s settings.

- **NUM_DATA_NIBBLE**: This field defines the number of data nibbles within a frame. A value between 1 and 8 should be set.
- **SDATA_CRC**: This field sets if the serial data nibble should be taken into account in the frame CRC calculation. 0 means that it is not taken into account, 1 that it is taken into account.
- **SDATA_USED**: This field sets if a valid serial data nibble should be generated in the frame. If set to 0, the serial data nibble is always set to 0. If set to 1, the nibble is computed from the user’s data.
- **TCLOCK**: This field defines the bit rate of the SENT communication (in ns). Values between 3000 and 10000 are valid (as per SENT specification).
- **INIT_CLK_DEV**: This field defines the deviation (in %) of the bit rate with respect to the nominal one defined by TCLOCK. Deviation lower than 80% or higher than 120% are to be detected by the DUT.
- **INIT_DATA**: This field defines the initial value of the data in manual mode.
- **INIT_SDATA**: This field defines the initial value of the serial data (ID + payload).
- **USER_DATA_SIZE**: This field defines the number of elements in the user’s defined data table (automatic mode).

In Automatic mode, the SENT frames are generated from the table defined in file “Users_data.c”.

# Code revisions history

<table>
<thead>
<tr>
<th>Code revision history table</th>
</tr>
</thead>
<tbody>
<tr>
<td>v 1.0</td>
</tr>
<tr>
<td>v1.1 Correct bug in crc calculation function crc4()</td>
</tr>
<tr>
<td>v1.2 Correct bug in clock ratio calculation in function CPU_viSRN0()</td>
</tr>
</tbody>
</table>

