SnS CAN API

Library

User's Guide

Rev 2.12

August 2020

1 Terminology and Environment

1. Sital SnS Transceiver is fully functional CAN/CAN FD Transmitter and CAN/CAN FD Receiver with additional functionality – getting Safe and Secure parameters.
2. SnS means Safe and Security.
3. User application can use Sital SnS API SoftWare and Sital SnS Transceiver as CAN Node defender or CAN Network defender or CAN Network attacker or CAN Network sniffer.
4. A CAN /CAN FD Network consists of several Nodes and one of them is Sital's SnS Node. The following algorithm and API library describe using of Sital's SnS Node as CAN Network defender.
5. The Safe and Secure parts can be named as Wires Fault detection and CAN Nodes identification accordingly.
6. The following algorithm and API library are detailed to the level of CAN Messages, so it's for the User Application to use Can Messages for binding with appropriate CAN Nodes and their locations.
7. On the level of Can Messages the Wires Fault detection can be explained as
   a. Wires Fault discover – Yes or No
   b. If Yes - Where its location - between what CAN Messages' origins
8. Only if there is no Wires Fault discovered (Wires Fault discover - No) the CAN Nodes Identification can be fulfilled.
9. On the level of Can Messages the CAN Nodes Identification can be named as CAN Message Authentication. If CAN Message Authentication is failed – this CAN Message occurrence can be named as Message Authentication fault.

2 Algorithm Description

There are three stages in working with SnS CAN API:

1. Calibration and learning phase
2. Tracking Wires or Authentication Faults condition phase
3. Post flight fault isolation

2.1 DataBase Terms' Definitions

<table>
<tr><td rowspan="3">Message ID</td><td rowspan="3">Period</td><td colspan="4">Fault Occurrences summarized</td><td rowspan="2">Learned SnS Entry 0</td><td rowspan="2">Occur SnS Entry 1</td><td rowspan="2">Occur SnS Entry 2</td><td rowspan="2">…</td><td rowspan="2">Recycler SnS Entry n</td></tr>
<tr><td colspan="2">Message Parameters</td><td colspan="2">During from Previous Msg</td></tr>
<tr><td rowspan="2">Non period</td><td rowspan="2">Tail</td><td rowspan="2">High Low</td><td rowspan="2">Bus errors</td><td>Counter</td><td>Counter</td><td>Counter</td><td>…</td><td>Counter</td></tr>
<tr><td>SnSParamsAv</td><td>SnSParamsAv</td><td>SnSParamsAv</td><td>…</td><td>SnSParamsAv</td></tr>
</table>

1. Main DataBase array is array of all CAN Messages Information structures, where CAN **Message ID** is a Key field for every such structure. This array is used on all three phases of the algorithm.
2. Every CAN Message Information structure consists of the following fields
   a. **Period** – the Period that learned during Learning Phase - for Periodical CAN Messages only
   b. The field **Fault Occurences summarized** consists of this CAN Message faults and during previous good CAN Message faults. All those faults are faults counters of the certain type.
   c. One **Learned SnS Entry** (index 0), several **Occurring SnS Entries** (index 1,2,3,…) and one **Recycler SnS Entry** (index – the last one). Everyone of them consists of the vector named - SnS Parameters Average values (SnSParamsAv on the diagram above) and Counter of the messages, which were participated in this Average calculation
3. BaseLine for every CAN Message is SnS Parameters Average values of the Learned SnS Entry (index 0). This SnS Entry is filled during Learning Phase, and used also on Tracking Phase.
4. On Tracking Phase for newly received CAN Message SnS Parameters Constant Threshold Vector Deviation value is used for selecting suitable SnS Entry of this CAN Message Information structure. This Threshold helps to choose such an SnS Entry (Learned or Occurring), where the distance to SnS Parameters Average values is less than this Threshold.
5. If there is no such an SnS Entry – this CAN Message SnS Parameters are going to Recycler SnS Entry.
6. Fault Occurrences are counted during Tracking Phase for every CAN Message and what happened during from previous received CAN Message.

2.2 Calibration – Learning Phase

1. This is the first phase of Algorithm. During this phase multiple CAN Messages from all Network Nodes are processed by Sital CAN Defender. Result of this processing is a reference BaseLine for every frame on the Network – Learned SnS Parameters Entry in DataBase.
2. This phase should be conducted following the user request, where place, time and conditions are chosen by the user. Typically the learning phase is done after power on.

2.3 Tracking Wires or Authentication Faults condition phase

As it was described in 2.1 –

1. For newly received CAN Message SnS Parameters Constant Threshold Vector Deviation value is used for selecting suitable SnS Entry of this CAN Message Information structure. This Threshold helps to choose such an SnS Entry (Learned or Occurring), where the distance to SnS Parameters Average values is less than this Threshold.
2. If there is no such an SnS Entry – this CAN Message SnS Parameters are going to Recycler SnS Entry.
3. All types of Fault Occurrences are counted during Tracking Phase for every CAN Message and what happened during from previous received CAN Message.
    a. Non Period – period of this Message is not suitable – threshold used
    b. Tail – this Message occurrence is not in the BaseLine
    c. High/Low – only half CAN signal occurrence
    d. Bus Errors – several types of CAN Bus errors from IP Core Error Register

2.4 Post flight Fault Isolation

1. If there are seen some problems of CAN what type of the problem is it – Wires fault(s) or Authentication fault(s)?
2. Wires Fault(s) are checked first. The combination of quantity non BaseLine CAN Message Occurrences and certain types and quantity of CAN Messages Faults and of between CAN Message Faults occurrences could exactly point out to Wire Fault(s) type of problems.
3. Please note that a wiring fault may be intermittent. In that case, it may be very short in time, and may reappear later. Constant Threshold Vector Deviation permissive value helps accurately locate the bus fault for intermittent issues. Also occurrences of the faults contribute to the ability to pin point the failure.
4. The wires fault is most likely located on the wires between two groups - BaseLine group and the others.
5. As it clarified earlier - given a topology architecture of the bus, and given a mapping of each frame to each node on the bus, it is thus possible to locate the fault on the specific network.
6. Authentication faults may be checked if Wires Fault(s) are not found.
7. Authentication fault(s) can be cought in a very simple way as without Wire Fault(s) CAN Network must work very clear having no side effected errors.

3 SnS API Reference

3.1 Sital SnS API Library Design

All Sital API functions are written on C/C++ and may be represented by the following diagram -

```
<return_value_data_type>

sitalSnS_<name_of_function>      (
                                   <arg1_data_type >      <argument1>
                                   <arg2_data_type >      <argument2>
                                   <arg3_data_type >      <argument3>
                                   ……..                   ……..

                                 )
```

Input and Output parameters are described separately, no Input/Output description means Input parameters.

All data types are defined in file **CommonTypes.h**. Those types are the same for several Operating Systems (OS).

File **ReturnCodes.h** defines return value constants, such as **sitalReturnCode_SUCCESS** or **sitalReturnCode_INVALID_PARAMETER**.

3.2 sitalSnS_Initialize

```
S16BIT sitalSnS_Initialize (
  U8BIT deviceId
)
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)

Description

Initializing. DataBase Array is cleared. May be called after power on or after **sitalSnS_Close()** function calling.

3.3 sitalSnS_Close

```
S16BIT sitalSnS_Close (
  U8BIT deviceId
)
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)

Description

May be called on any stage and at any moment. This function stops all activity and prepares all the parameters and memory for calling **sitalSnS_Initialize()** function.

3.4 sitalSnS_MsgLearning

```
S16BIT sitalSnS_MsgLearning (
                              U8BIT          deviceId
                              RecMsgInfo *   pRxMsg
                              U16BIT *       msgIdIndex
                              SnSParams *    calcDeviation
                              BOOL *         bParticipation
                             )
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
pRxMsg	A Pointer to Received CAN message
Outputs	
msgIdIndex	This CAN message ID information Index in current DataBase
calcDeviation	This CAN message SnS Parameters Deviation vector that contains deviations from Temporary Average of the Learned SnS Parameters Entry in CAN message SnS Info array.
bParticipation	Participate or not in forming Learned SnS Parameters Average values.

Description

This function forms Learned SnS Parameters Entry in CAN message SnS Info array.
This is the first stage of SnS processing. All hardware filters must be disabled – **sitalCan_EnableFilters(FALSE).**

Search the received CAN Message (Frame) ID in the current DataBase.
If it is not in the list, output function parameter ***msgIdIndex*** has next available value. If it is in the list, ***msgIdIndex*** has this found value.
Newly received CAN message SnS Parameters are comparing to temporary SnS Parameters Average vector of this CAN message Learned Entry (not for first time this CAN Message arrived).

If Calculated deviation vector is *close enough* to Average vector - newly received CAN message SnS Parameters are participating in temporary average SnS Parameters calculation for Learned SnS Parameters Entry. ()

If not – there is no participation in average calculation. (*close enough* criteria will be described in another document).

This Calculated deviation vector is the ***calcDeviation*** output function parameter and boolean ***bParticipation*** output function parameter means **True** – participation, **False** – not participation in average calculation.

3.5 sitalSnS_MsgTracking

```
S16BIT sitalSnS_MsgTracking (
                              U8BIT          deviceId
                              RecMsgInfo *   pRxMsg
                              U16BIT *       msgIdIndex
                              U8BIT *        apprSnSEntryIndex
                              BOOL *         faultOccurrence
                             )
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
pRxMsg	A Pointer to Received CAN message
Outputs	
msgIdIndex	This CAN message ID information Index in current DataBase
apprSnSEntryIndex	Appropiate SnS Entry index for this CAN message SnS Parameters.
faultOccurrence	FALSE – no Fault Occurrences , TRUE - Fault Occurrence is added - one or more of them. It may happen during the period from previous CAN Message or/and it's about current CAN message.

Description

Using CAN message SnS Info array for tracking every CAN message with purpose to accumulate all the information needed for the next stage of the Algorithm.This is the second stage of SnS processing, after the first one - learning stage is completed (see also 2.3 Tracking Wires or Authentication Faults conditions).
All hardware filters must be disabled – **sitalCan_EnableFilters(FALSE).**

This API function performs the following steps:
1. Search the Message (Frame) ID in the current DataBase (after learning stage).
   a. If it is not in the list, output function parameter ***msgIdIndex*** returns (-1) value. User should consider calling the learning function **sitalSnS_MsgLearning** to add it to the list.
   b. If it is in the list, its index is returned by ***msgIdIndex.***
2. Among this CAN message ID information in the DataBase (by ***msgIdIndex***) compare the received SnS parameters with the every Entry Average SnS parameters, and first of all with the SnS entry that '' reflects'' BaseLine – Learned SnS parameters.
3. If it is within the tolerated distance, the count of that SnS entry is incremented by 1, and the SnS average of that entry is updated to accommodate for slow environmental changes.
4. If it violates the tolerated distance, then it is added to one of the entries that were kept for the violating distances - Occurring SnS Entries. Also Tail Fault Occurrence of this Message is increased.
   a. A search is made in all the entry bins, each bin represents a prior findings, and if a match is found, that entry bin counter is incremented by 1.
   b. If no entry match, and it is not the last entry, a new fresh entry is updated with the new SnS parameters.

c. If no entry match, at all entries are full, this message and its SnS parameters are added to an additional entry that is kept for such cases - Recycler SnS Entry. These SnS parameters would be overwritten every time a new message fails to match any of the tolerated entries.

apprSnSEntryIndex is output function parameter that means appropriative SnS entry index within current message ID DataBase information.

faultOccurrence is output function parameter TRUE means that one or more of Fault Occurrences are happened, FALSE – no Fault Occurrences are happened.

3.6 sitalSnS_FaultIsolation

```
S16BIT sitalSnS_FaultIsolation (
                                U8BIT       deviceId
                                BOOL *      bWireFault
                                U16BIT*     msgNonProblemNumber
                                canid_t*    msgNonProblemArray
                                U16BIT*     msgProblemNumber
                                canid_t*    msgProblemArray
                               )
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
Outputs	
bWireFault	Wire Fault was detected or not
msgNonProblemNumber	Number of messages (elements) in ***msgNonProblemArray***
msgNonProblemArray	Array of Message IDs where no problem were appeared
msgProblemNumber	Number of messages (elements) in ***msgProblemArray***
msgProblemArray	Array of Message IDs where problem were appeared

Description

After two previous stages the DataBase CAN message SnS Info Array is ready for processing (please, see 2.4 Post Flight Fault Isolation).
Output Parameter ***bWireFault*** is TRUE means that Wire Fault(s) was detected, and FALSE means that Authenticantion problem can appeared. Following description from 2.4 - it's possible to determine two types of frames – no problem was found and a problem is detected (please, see 2.4 Post Flight Fault Isolation). Those two groups of frames are put to two output arrays.
Those arrays can be used by User for further Fault processing.

3.7 sitalSnS_GetDataBase

```
S16BIT sitalSnS_GetDataBase (
    U8BIT    deviceId
    U8BIT*   nameOfFile
)
```

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
nameOfFile	Name of File that contains DataBase information

Description

DataBase information is sent as ASCII symbols to Terminal. The Terminal can be configured to send simulteniously all the information to the hard disk.

17 Atir Yeda St., Kfar-Saba, ISRAEL 44643

Email: info@sitaltech.com

Website: http://www.sitaltech.com

The information provided in this User's Guide is believed to be accurate; however, no responsibility is assumed by Sital Technology for its use, and no license or rights are granted by implication or otherwise in connection therewith. Specifications are subject to change without notice.

Please visit our Web site at http://www.sitaltech.com for the latest information.

© All rights reserved. No part of this User's Guide may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission by Sital Technology.