SnS CAN API

Library

User's Guide

Rev 2.10.1

August 2020

1 Terminology and Environment

1. Sital's SnS CAN Defender is a fully functional CAN bus/CAN-FD/ARINC-825-4 node with additional functionality. The added functionality is the capability to capture Safe and Secure parameters.
2. SnS is Sital's acronym for Safe and Secure.
3. Sital's SnS CAN Defender can be used by Sital's SnS software as a CAN bus/ARINC-825-4 node defender, network defender, network attacker or network monitor.
4. A CAN bus/ARINC-825-4 network consists of several nodes with Sital's SnS CAN Defenders used as a CAN network defender.
5. The algorithm described herein provides wire fault detection as the "safety" part of an SnS CAN Defender.
6. The "security" part of Sital's CAN Defender will be described in another document.

2 Algorithm Description

Sital's CAN bus SnS wire fault sensor detection algorithm consists of three phases of operation:

1. Learning Phase
2. Tracking Wire Fault Conditions Phase
3. Fault Isolation Phase

2.1 Terms Definitions

Table 1. CAN Messages SnS Information Array

Message ID	Message Period	Fault_Occurence	In-Range SnS Entry 0	Out-of-Range SnS Entry 1	Out-of-Range SnS Entry 2	Out-of-Range SnS Entry 3	…	Recycler SnS Entry N
Message_ID_1	Period for Nessage_ID_1	Fault_Occurence_ Count	*SnS_Entry_0_ Count*	*SnS_Entry_1_ Count*	*SnS_Entry_1_ Count*	*SnS_Entry_1_ Count*	…	*SnS_Entry_N_ Count*
			SnSParamAv0	*SnSParamAv1*	*SnSParamAv2*	*SnSParamAv3*	…	*SnSParamAvN*

1. As shown in Table 1, the CAN Messages SnS Information Array is an array of CAN bus Message SnS information structures, where the CAN bus Message_ID is a Key field for each such structure.
2. Each CAN Message Information structure consists of one In-Range SnS Entry (Index 0) and zero, one or multiple Out-of-Range SnS Entries (Indices 1, 2, 3,…).
3. Each SnS Entry's structure consists of the vector called SnS Parameters Average values. This consists of the array of "SnSParamsAvX" values in the table above, where "X" is the index value, and the counter value of the messages for the respective index (SnS_Entry_X_Count). These counter values, which indicate the number of messages affecting the respective entry's statistics for the particular Message_ID, are used for computing the Parameter Average values.
4. The Baseline for every CAN bus/ARINC-825-4 message is the SnS Parameters Average values of the In-Range SnS Entry (index 0). This SnS Entry is initially computed during the Learning Phase, and is also slowly updated during the Tracking Wire Fault Conditions Phase.
5. During the Tracking and Wire Fault Condition Phase, each received CAN bus/ARINC-825-4 message is sorted into the appropriate entry or "bin", where each entry is identified by its index value and has associated SnSParamsAvX and SnS_Entry_X_Count values. This process involves determining whether the parameter value for newly received messages is within the pre-determined tolerance (designated as "tolerance") of the bin's nominal (center) value. The default value for the tolerance is 15 units, and the width of each bin = 2●tolerance. If desired, users may increase or decrease the value of tolerance by means of changes to Sital's software library source code.
6. If there is no appropriate pre-existing SnS Entry for a particular message, the software will create a new Out-of-Range SnS Entry column, using the next available index value. The center (mid-point) value for the new Out-of-Range SnS Entry bin (column) must be an integral multiplier of 2●(tolerance) from the center (average) of the In-Range SnS Entry (index 0) bin (column). The range of the parameter values for messages going into the new bin will be = 2●(tolerance). The new measured parameter value must fall within ± tolerance of the bin's center value.

7. Fault Occurrences is a count of the total number of Out-of-Range parameter values measured for the respective Message_ID. This encompasses messages affecting the statistics for *all* of the Out-of-Range message entries.
8. The column labelled "Recycler SnS Entry N**"** is to accommodate a situation where a new message is received with a parameter value that is out-of-tolerance for both the "In-Range" bin and all of the "Out of Range bins" and there are no additional "Out of Range bins" available. In this case, the new message is stored in the Recycler bin.

2.2 Learning Phase

The Learning Phase is the first step of Sital's CAN Defender's overall algorithm. The Sital SnS software enters this phase as the result of a user request. Typically, this is done following power turn-on, but may also be done periodically.

In order to capture all SnS statistics during the Learning Phase, and also to maximize the probability of detecting intermittent faults during the Tracking and Wire Fault Conditions Phase, it is recommended to *disable* the SnS CAN Defender's IP hardware Message_ID filter. This will enable the SnS data collection process to measure parameters for *all* CAN messages and pass these statistics to the Sital and user software. Message_ID filtering may still be performed by application software.

During this phase, CAN message data and SnS parameter data are stored in the IP core's shared memory. This FIFO/circular buffer memory structure can store message data and SnS data for up to the 15 most recently received messages. As a result, in order to avoid data loss, it is necessary for software to frequently read this memory and copy it to host memory.

During this phase, multiple CAN Messages from all Network Nodes are processed by the Sital CAN Defender. The involves the Sital SnS CAN Defender's hardware and software proceeding with its "learning" process. The purpose of this process is to develop the "fingerprint" or "signature" for the signal characteristics of messages received from all Message_IDs on the bus. This process involves the IP hardware measuring various pulse widths and other timing parameters by means of a high-frequency sampling clock and performing averaging calculations over the course of each received message.

Software computes the averages for each of the individual parameters for *all* received messages associated with each individual Message_ID. The result of this approximately 4-second process is that the SnS sensor software computes an array of parameters to characterize the messages received from all Message_IDs on the bus. By the completion of the learning process, for each Message_ID, software will have written the values of SnSParamAv0 and SnS_Entry_0_Count for the "In-Range bin" denoted by index value 0 to the CAN Messages SnS Information Array table, as shown in Table 1.

At this time, for each Message_ID, the value SnSParamAv0 will define the center point of the In-Range (SnS Entry 0) bin. The minimum value for future values of the massage's parameter affecting this bin's statistics will be = SnSParamAv0 – tolerance, and the maximum value for future values of the massage's parameter affecting this bin's statistics will be = SnSParamAv0 + tolerance.

This process establishes the baseline for all CAN bus messages to be received over the network. Once the learning process has been completed, the SnS CAN Defender is now prepared to enter the Tracking Wire Fault Conditions Phase and begin monitoring all message transmissions for the purpose of detecting wire faults.

2.3 Tracking Wire Fault Conditions Phase

During the Tracking Wire Fault Conditions Phase, the Sital SnS hardware and software will continuously measure and compute the signals' parameter values for all received messages. In the same way as during the learning phase, this process involves hardware measuring various pulse widths and other timing parameters by means of a high-frequency sampling clock and performing averaging calculations over the course of each received message. For each message, the values of the various measured and averaged parameters are compared with the average value (SnSParamAv0) initially determined during the Learning Phase. The value of SnSParamAv0 is updated very slowly during the Tracking Wire Fault Conditions Phase by means of a low-pass digital filter applied to the new incoming parameter values.

The Tracking and Wire Fault Conditions Phase involves sorting received message parameters into specific "bins" (entries) within the CAN Messages SnS Information Array (Table 1). This process involves determining whether the parameter values for newly received messages are within the pre-determined tolerance of the average value for each Message_ID. The default value for this tolerance is 15 units. If desired, users can change this value by means of changes to Sital's software source code.

During this phase, for each received message's parameter, software determines whether the parameter's value for the current message is In-Range or Out-of-Range. To determine that the measured parameter value for a message is In-Range, its value must be within ± tolerance of the parameter's maintained In-Range average value (SnSParamAv0) prior to the receipt of the current message. If all of a message parameter values are determined to be In-range, software will determine that this message did *not* indicate a wire fault.

However, if the absolute value difference between the measured value(s) of a parameter for a received message and the previously computed In-Range average for that parameter (SnParamAv0) is greater than the tolerance value, then Sital's SnS software will characterize that message as indicating a wiring fault and will set the value of the Fault_Occurence Boolean to '1'. The Sital API will pass the value of the Fault_Occurence Boolean to application software in real time. This will result in software the incrementing the value of the Message_ID's Fault_Occurrence_Count by one.

For each parameter received for each message, the respective entry in the CAN Messages SnS Information Array will be updated. During the Learning Phase (and also possibly during the Tracking and Wire Fault Conditions Phase), a new row is created in the CAN Messages SnS Information Array for each new Message_ID that's received. When a new message is received containing a specific Message_ID, the values in the respective row of the CAN Messages SnS Information Array table will be updated. If the parameter value has been determined to be In-Range, then SnS_Entry_0_Count will be incremented by one and the SnSParamAv for In-Range SnS Entry 0 will be updated based on the output of a low-pass software digital filter. In this case, the value of the Fault_Occurrence_Count for the specific Message_ID will *not* be incremented.

If the parameter value has been determined to be Out-of-Range, then the Sital software will determine which entry (column) of the row of the table for the specific Message_ID value will be updated. The software will search the columns to determine whether current measured parameter value is within ± tolerance of the value of SnSParamAv1 for Out-of-Range SnS Entry 1. If not, then the software will perform the same test for the parameter value relative to for Out-of-Range SnS Entry 2, Out-of-Range SnS Entry 3, etc.

If the current measured parameter value is within ± tolerance of the SnSParamAvX average for an existing Out-of-Range SnS Entry, then that column of the CAN Messages SnS Information Array table will be updated for the specific Message_ID. That is, the values of SnS_Entry_X_Count *and* the Message_ID's Fault_Occurrence_Counter will be incremented by one and the value of SnSParamAvX will be re-computed as the updated arithmetic average of this parameter's values for all entries falling within the range for the Out-of-Range Entry X.

However, if the parameter value is *not* within ± tolerance of the SnSParamAvX for *any* of the existing Out-of-Range SnS Entries, then it will be necessary for Sital's software to create a new Out-of-Range SnS Entry bin

(column). In this case, the center point for the new Out-of-Range SnS Entry bin (column) must be a multiple of 2●tolerance away from the center point of the In-Range entry (Entry 0) and the parameter value for the latest received message must fall within ± tolerance of the center point of this newly created entry. At this time, SnS_Entry_X_Count will be initialized to a value of 1 and the value of will SnSParamAvX will be assigned the value of the parameter for the most recently received message.

Wiring faults may be intermittent. In that case, they may occur for a very short time period, and may or may not reappear later. Storing the average of several Out-of-Range SnS entries helps to accurately locate the bus fault for intermittent open circuit and short circuit faults. All occurrences of the fault are averaged and contribute to the capability to locate the fault.

2.4 Fault Isolation Phase

During this phase, Sital's SnS CAN Defender's API/library software and user software need to determine the nature and approximate location of intermittent and continuous wire faults. These determinations will be made based on statistics stored in the CAN Messages SnS Information Array Table. Further, it is recommended that they be based on a series of rules:

- There's a number of different types of wire faults that can be detected:
  - Open circuit fault for CAN_H only.
  - Open circuit fault for CAN_L only.
  - Open circuit fault for both CAN_H and CAN_L (which are assumed to be in close proximity with each other).
  - Missing bus termination.
  - Short circuit fault between CAN_H and CAN_L.
  - Short circuit fault between CAN_H and V_{CC}.
  - Short circuit fault between CAN_H and GROUND.
  - Short circuit fault between CAN_L and V_{CC}.
  - Short circuit fault between CAN_H and GROUND.
- It is suggested to accumulate statistics in the CAN Messages SnS Information Array for 5-minute periods. Following that, software can perform analyses based on the rules below to determine and locate wire fault conditions. Following that, the IP hardware and software will be able to continue gathering additional data. Optionally, software may re-initiate the Learning Phase prior to re-starting the Tracking Wire Fault Conditions Phase.
- In order to be able to determine the approximate locations of open circuit faults, it's required for user software to be cognizant of the mapping between Message_ID values, physical nodes and nodes locations within the CAN bus's topology.
- In general, the difference in parameter averages between the In-Range entries (SnSParamAv0) and the various Out-of-Range parameters' averages (SnSParamAvX) will be indicative of the distance between the SnS CAN Defender and the location of wiring faults. However, there are multiple additional factors that need to be taken into account. These include bus network topology, variations in cable propagation constant and impedance, connector impedances, stub locations, stub lengths and ECU impedances.
- For a fault consisting of open wires on CAN_H ***and/or*** CAN_L, nodes on opposite sides of an open circuit fault cannot communicate with each other.
- For a fault consisting of an open wire for CAN_H ***or*** CAN_L but ***not*** both, concurrent bus traffic on both sides of the fault will disrupt communication on the opposite side of the fault. If an intermittent single-wire open circuit fault occurs during a message's arbitration process for a node on one side of the fault while a message is being transmitted on the opposite side of the fault, this is very likely to prevent successful completion of the CAN bus arbitration process and successful reception of the message on the opposite side of the wire break. Similarly, if an intermittent single-wire open circuit fault occurs while messages are being transmitted on both sides of the fault, it is highly likely there will be CRC failures at receiving nodes for the messages on both sides of the fault.
- An intermittent single-wire or two-wire open circuit fault will result in Out-of-Range parameter values, but only from messages received from nodes on the ***same*** side of the wire fault as the SnS CAN Defender. It's

not possible to receive valid, complete CAN messages from the opposite side of the open circuit wire fault during times when the fault is present. As a result, messages transmitted by nodes located on the remote side of a fault will *not* affect the statistics in the data base: either the In-Range *or* Out-of-Range statistics. Because of that, software will be able to determine and aggregate the node identities for all of the Message_IDs from messages with Out-of-Range parameter values in the CAN Messages SnS Information Array. Based on those statistics, along with knowledge of the CAN bus's topology (locations of nodes), the software will be able to determine that the open circuit fault is located between two adjacent nodes on the bus for which one node populated Out-of-Range statistics into the CAN Messages SnS Information Array table while the other node did *not* populate Out-of-Range statistics into the table.

- For open circuit faults, it *is* possible to distinguish between single-wire and two-wire faults.
- An open circuit fault in a bus termination is very likely to result in Out-of-Range values parameter from messages from most or all Message_IDs and therefore all nodes on the CAN bus.
- If the SnS CAN Defender is on the same side of a short circuit fault as the transmitting node, short circuit faults *can* be detected. However, their approximate location can only be determined for messages in which the SnS CAN Defender is the transmitting node.
- In general, if either CAN_H or CAN_L are shorted to ground, no CAN bus communication can take place. However, when operating at low data rates such as 125 kbps, sometimes messages can be received correctly.

Following notifications of wiring faults, the user's application may execute its security playbook software and take the appropriate system-level actions. These can include shutting down a CAN bus and switching traffic to a redundant bus, shutting down certain processes at the local system level and/or communicating about the anomalies' occurrence over the CAN bus or some other interface. Such actions may be necessary to mitigate against the risk of flight or mission failure.

3 CAN Messages SnS Information Array

3.1 CAN Message SnS Info

```
Struct MsgSnSInfo

{
 canid_t                 MsgId
 U32BIT                  FaultOccurrences
 SnSParamsStruct         SnSParamsAv[SnS_entry_index]
 U32BIT                  msgCounter[SnS_entry_index]
}
 MsgSnSInfoStruct
```

Elements

MsgId	CAN message ID
FaultOccurrences	Post flight Fault Isolation result for every CAN Message
SnSParamsAv	SnS Parameters Average for every SnS entry
msgCounter	CAN message counter for every SnS entry

Description

For every CAN bus Message_ID, there is one In-Range Entry and possibly one or more Out-of-Range SnS Entries, where ***SnS_entry_index*** for Learned SnS Parameters Entry is Zero (0). The initial statistics for the In-Range entry come out of the Learning phase. All SnS entries have a constant tolerance value (= “tolerance”). An SnS Parameters Average SnSParamAvX value is computed for each SnS entry. The value of the entry’s counter SnS_Entry_X_ Count is used for updating the value of the Parameter’s Average, SnSParamAvX.

For every Out-of-Range parameter, the value of the Message_ID’s Fault_Occurence_ Count will be incremented by one.

DataBase: The CAN Messages SnS Information Array table is used in all three stages of the algorithm to provide a complete picture of the SnS Parameters’ average values for all CAN Message_IDs.

Table 2. Main DataBase CAN Messages SnS Information Array -- Example

Message_ID	**Message Period (ms)**	**Fault_Occurence**	**In-Range SnS Entry 0**	**Out-of-Range SnS Entry 1**	**Out-of-Range SnS Entry 2**	**Out-of-Range SnS Entry 3**	…	**Recycler SnS Entry N**
Message_ID_1	Period for Message_ID_1	Fault_Occurrence _Count_1	*SnS_Entry_0_ Count*	*SnS_Entry_ 1_ Count*	*SnS_Entry_ 2_ Count*	*SnS_Entry_3_ Count*	…	*SnS_Entry_N_ Count*
			SnSParams Av0	*SnSPara msAv1*	*SnSPara msAv2*	*SnSParams Av3*	…	*SnSParams AvN*
0x232	15	25	10000	15	06	03	…	01
			<SnS values>	<SnS values>	<SnS values>	<SnS values>	…	<SnS values>
0x456	20	23	15287	12	05	04	…	02
			<SnS values>	<SnS values>	<SnS values>	<SnS values>	…	<SnS values>

Table 2 is an example showing the various counter values of CAN messages SnS information array for two CAN Message_IDs. This shows an example that could occur during the Tracking Wire Fault Condition Phase of the SnS algorithm.

4 SnS API Reference

4.1 Sital SnS API Library Design

All Sital API functions are written in 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>
                                   ……..                   ……..

                                 )
```

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**.

4.2 sitalSnS_Initialize

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

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)

Description

Initialization: The Main DataBase Array is cleared. This function may be called after power turn-on or after calling the **sitalSnS_Close()** function.

4.3 sitalSnS_Close

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

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)

Description

This function may be called during any phase and at any time. This function stops all activity and prepares all the parameters and memory for calling the **sitalSnS_Initialize()** function.

4.4 sitalSnS_MsgLearning

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

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
pRxMsg	Pointer to Received CAN message
Outputs	
msgIdIndex	CAN Message_ID information Index in current DataBase
calculatedDeviation	This CAN message SnS Parameters Deviation from the average of the In-Range SnS Parameters Entry (SnSParamAv0) in the CAN messages SnS Information array.
bParticipation	Participate or not in forming the In-Range SnS parameters' average values.

Description

This function creates the In-Range Parameters Entry (index 0) in the CAN message Main DataBase Information array. This is the first stage of SnS processing.

All hardware filters must be disabled – **sitalCan_EnableFilters(FALSE).**

This function searches the received CAN Message (Frame) ID in the current DataBase.
If it is not in the list, output function parameter ***msgIdIndex*** has the next available value. If the Message_ID is in the list, ***msgIdIndex*** indicates that the software has found the entry bin (column) for this value in the table. Newly received CAN message SnS Parameters are compared to the SnS Parameters Average (SnSParamAv0) of this Entry and the deviation value is calculated. The absolute value of the deviation is compared to the tolerance value. The processing for the In-Range entry – Entry number 0 – is the same as for any other Entry (Out-of-Range SnS Entries) and is described in paragraph 2.2.

If the absolute value of the calculated deviation value is less than the tolerance value, then the newly received CAN message SnS Parameters are participating in computing the value of the parameter average, SnSParamAv0.

If the absolute value of the calculated deviation value is greater than the tolerance value, the parameter value for the message is not included in re-computing the the parameter average, SnSParamAv0.

This Calculated Aggregative deviation value is the ***calculatedDeviation*** output function parameter and boolean ***bParticipation*** output function parameter means

True indicates participation. **False** indicates not participation in the average calculation.

4.5 sitalSnS_MsgTracking

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

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
pRxMsg	A Pointer to the Received CAN message
Outputs	
msgIdIndex	The CAN message ID information Index in the current DataBase
apprSnSEntryIndex	Appropriate SnS Entry index for these CAN message SnS Parameters.
calculatedDeviation	The CAN message SnS Parameters Deviation from the average (SnSParamAvX) of the appropriate Entry (“X”) in the CAN Messages SnS Information Array.

Description

This function uses the CAN Messages SnS Information Array for tracking every CAN message with the purpose of accumulating the statistics required for the next stage of the SnS Algorithm. This second stage of SnS processing occurs immediately following the completion of the first phase, which is the Learning Phase.
For best performance of the SnS algorithm, the hardware Message_ID filter should be disabled – **sitalCan_EnableFilters(FALSE).**

This API function performs the following steps:
1. Search the Message (Frame) ID in the current CAN Messages SnS Information Array (after Learning stage).
   a. If the Message_ID is not in the list, output function parameter ***msgIdIndex*** returns -1 value. The user should then consider calling the learning function to add it to the list.
   b. If the Message_ID it is in the list, its index is returned by ***msgIdIndex.***

2. Starting with the In-Range entry (Entry 0), this function compares the SnS parameters for the new received message with those in all entries in the CAN Messages SnS Information Array.

3. If the parameter for the new message is within ± tolerance from the value of SnS parameter average (SnSParamAv0) for the In-Range entry (Entry 0), then the counter value for the In-Range SnS Entry (SnS_Entry_0_Count) is incremented by 1, and the In-Range SnS average (SnSParamAv0) is re-computed using a digital low-pass filter to accommodate for slow environmental changes.

4. If the parameter value for the new message is outside the ± tolerance from the value of the SnS parameter average (SnSParamAv0) for the In-Range entry (Entry 0), then the new message needs to be added to one of the Out-of-Range entries.

   a. A search is made through the various Out-of-Range entries. Each of these entries represents prior findings, and if a match is found, that entry's counter value (SnS_Entry_X_Count) is incremented by 1 and the entry's average value (SnsParamAvX) is recomputed based on the vales of the previous average, the new parameter entry and the Count value.

   b. If there's no Out-of-Range entry for which the new message's parameter is within ± tolerance from the value of its SnS parameter average, then the function creates a new entry based on the value of the SnS parameter for the new message. In this case, the center point for the new Out-of-Range SnS Entry bin (column) must be a multiple of 2●tolerance away from the center point of the In-Range entry (Entry 0) and the parameter value for the latest received message must fall within ± tolerance of the center point of this newly created entry. At this time, SnS_Entry_X_Count will be initialized to a value of 1 and the value of will SnSParamAvX will be assigned the value of the parameter for the most recently received message.

apprSnSEntryIndex is an output function parameter that indicates the value of the SnS entry index within the current CAN Messages SnS Information Array.

calculatedDeviation is an output function parameter that indicates the calculated deviation from the average value (SnSParamAvX) for this SnS entry.

4.6 sitalSnS_FaultIsolation

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

Parameters

deviceId	Unique Device ID 0 - (sitalMaximum_DEVICES - 1)
Outputs	
msgNonProblemNumber	Number of messages (elements) in ***msgNonProblemArray***
msgNonProblemArray	Array of Message IDs where no problem was evident
msgProblemNumber	Number of messages (elements) in ***msgProblemArray***
msgProblemArray	Array of Message IDs where a problem was evident

Description

Following completion of the first two phases, the DataBase CAN message SnS Info Array is ready for processing. Using the Fault Occurrences field, it's possible to determine two types of messages: (1) messages for which no problem was found; and (2) messages for which a problem was detected. These two groups of frames are then written to a pair of output arrays. These two arrays may be used by the user for further fault processing.

Refer to paragraph 2.4 for a description of the Fault Isolation Phase.

4.7 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

The DataBase information from the CAN Messages SnS Information Array is sent as ASCII symbols to a host computer. The host computer may then store this information to its hard drive.

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.