

August 2020

SnS CAN API Driver

User Guide

Rev 1.4

August 2020



August 2020

This document refers to Xilinx Vivado and SDK Release Version 2018.2.

Copyright (C) 2020 by Sital Technology Ltd.

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Sital Technology Ltd.



August 2020

Table of Contents

1	Intr	oduction4
	1.1	Scope
	1.2	Audience
	1.3	Support4
	1.4	About this User Guide
2	Arcl	hitecture and Software Layers5
	2.1	Components
	2.2	Delivery Package
3	Env	ironment and Settings7
	3.1	Required Settings and Files by Application and Library projects7
4	Driv	ver Layer – Layer2 & Layer111
	4.1	CAN API Driver
5	Viva	ado Project12
	5.1	Vivado project Top modules
	5.2	Vivado project Output/Input Ports
	5.3	Vivado project Top modules Addresses
6	Test	ting Schem and Environment15
	6.1	Network Schem
	6.2	Board, Transceivers, Network15
	6.3	Terminal Program
7	Test	t Application17
	7.1	Program Description
	7.2	Test Results19



August 2020

1 Introduction

1.1 Scope

This document provides a walkthrough for CAN API StandAlone 2018.2 AXI UltraScale+ driver development.

1.2 Audience

This document assumes basic familiarity with Vivado and SDK 2018.2 provided by Xilinx. The data and procedures described in this document cover <u>UltraScale+, Xilinx SDK Release Version 2018.2</u>.

1.3 Support

If you have any question or require further assistance, use any of the following methods to contact Sital customer support:

- By Email: support@sitaltech.com
- By Phone: +972-9-7633300
- By Fax: +972-9-7663394

1.4 About this User Guide

This document is the user's manual for the CAN API software for Sital's AXI ARINC-825-4 IP Core.

This core incorporates an AXI Slave core coupled with one ARINC-825-4 IP core. This manual is intended to serve as a user's manual for aspects of the AXI ARINC-825-4 core's operation that are not covered by the ARINC825 IPs manual. Please refer to the ARINC-825-4IP core user's manual for a detailed description of the ARINC-825-4 IP.



August 2020

2 Architecture and Software Layers

2.1 Components

The software architecture is composed of the following components / blocks:



Table 1 below provides descriptions of the various software layers.

		Table 1. Software Layers
Layer	Name	Purpose & Functionality
User Application	Code example	- Entry point for the application
		- Get Number of CAN devices in AXI / PCI
		- Initialize all devices
		 Create frame, messages, receive and transmit messages
SnS API layer		 Apply SnS logic by user application calls
		 Write and Read using driver layer to specific SnS device
		 Support DataBase arrays of SnS layer
Driver layer	Layer2 – SitalSDK	 Negotiate with layer1 to verify devices communication
		- Translate RW from Application layer to AXI address RW
	Layer1 – SitalAPI	- RW via AXI to and from IP core

 Layer1 – SitalAPI
 RW via AXI to and from IP core

The software image deployed to the target platform is based on target-specific StandAlone BSP. The flow of creation is:

- Export Vivado hardware project AXI Master, AXI Slave ARINC825 (and PMOD socket Configuration for transceiver connection), targeting for the platform you are working on

For the software components; User application and driver layers

- Create Xilinx SDK StandAlone Application and Library project <u>Note:</u> User application uses Driver Layer library.
- NOTE for settings details of the various projects described, see chapter 3 of this document.



August 2020

2.2 Delivery Package

The delivery Package contains the following projects' settings and the complete source code

- Layer2 (Driver Layer) Library project
- Layer2 Driver documentation
- Vivado project for Avnet UltraZed-EG SOM with UltraZed_EG IO Carrier Card
- Test User application



August 2020

3 Environment and Settings

Notes: -

- 1. Current StandAlone AXI version was developed and tested on a Xilinx ZYNQ evaluation board consisting of an Avnet UltraZed-EG SOM System on Module and an UltraZed_EG IO Carrier Card. This used StandAlone Xilinx SDK Release Version 2018.2.
- 2. All projects source code and environments are delivered as a reference; See section 2.1.

3.1 Required Settings and Files by Application and Library projects

General Note: For all Test Application and Library projects, add the Symbol

- STANDA_AXI - for AXI ARINC825 Driver

To do this, go to project properties: Project \rightarrow Properties -> C/C++ Build -> Settings -> Symbols -> "STANDA_AXI" (-D is added automatically by SDK).

SDK Properties for CodeSample			— 🗆 X
type filter text	Settings		<> ▼ <> ▼ ▼
 type filter text Resource Builders C/C++ Build Build Variables Environment Logging Settings Tool Chain Editor C/C++ General Project References Run/Debug Settings 	Settings Configuration: Debug [Active] Tool Settings Devices Perform General Symbols Symbols Symbols Warnings Optimization Debugging Profiling Doirectories Miscellaneous Software Platform Processor Options Software Platform Charleneous ARM v8 g++ linker General Libraries Miscellaneous Software Platform Directories Software Platform Software Platform	Build Steps Puild Artifact R E	Configurations Sinary Parsers Sinary Parsers<



August 2020

To use BSP .h files in a library project, one needs to go to project properties: Project \rightarrow Properties -> C/C++ Build -> Settings -> Directories and to add the following path –

../../CodeSam31_bsp/psu_cortexa53_0/include/

Where CodeSam31_bsp - BSP name (CodeSam207_bsp - in latest delivery).

SDK Properties for StlCan_API	
type filter text	Settings $(\neg \neg \neg \neg \neg$
 Resource Builders C/C++ Build Build Variables Environment Logging 	Configuration: Debug [Active] • Manage Configurations • Tool Settings • Devices • Build Steps • Build Artifact • Binary Parsers • Error Parsers • Erro
Tool Chain Editor ▷ C/C++ General Project References Run/Debug Settings	 MRM v8 gcc assembler Beneral S ARM v8 g++ compiler Bymbols Bymbols Warnings Doptimization Debugging Binfered Options Binfered Options Binfered Options Borcessor Options S ARM v8 archiver Beneral ARM v8 archiver Beneral
	Restore Defaults Apply
?	OK Cancel



August 2020

3.1.1 Driver – Layer2 and Layer1 files

For the driver layer project, import the following files:

can4linux.h CommonTypes.h ReturnCodes.h SitalAPI.h SitalCan_API.cpp SitalCan_API.h SitalCan_SDK.cpp

3.1.2 SnS API Layer files

For the SnS layer project, import the following files:

can4linux.h CommonTypes.h ReturnCodes.h SitalCan_API.h SitalSnS_API.cpp



August 2020

3.1.3 User Application – test

- Make Linker to use Driver Layer Library Project and SnS Layer Library project

_

^{SDK} Properties for CodeSam67			— 🗆 X
type filter text	Settings		<> ▼ <> ▼ ▼
type filter text > Resource Builders ~ C/C++ Build Build Variables Environment Logging Settings Tool Chain Editor > C/C++ General Project References Run/Debug Settings	Settings Configuration: Debug [Active] Tool Settings Devices Configuration: Debug [Active] Configuration: Debug General Symbols Warnings Optimization Debugging Profiling Directories Miscellaneous Software Platform Processor Options Chiker Script Sinker Script Software Platform Directories Inferred Options Directories Directories Software Platform Directories Software Platform Directories Directories Directories Directories Directories Directories Directories Directories Directories Directories Directories Directories Directories Directories Software Platform Directories Directories Software Platform Directories ARM v8 g++ Linker Software Platform Directories Software Platform Directories Dire	Build Steps Build Artifact Build Steps Cother options (-XLinker [option]) Cother Objects Cother Objects Dother	Manage Configurations Manage Configurations ary Parsers S
	< >		Restore Defaults Apply
?			OK Cancel



August 2020

4 Driver Layer – Layer2 & Layer1

4.1 CAN API Driver

Xilinx SDK 2018.2 StandAlone AXI driver is used as low level driver in accordance with Vivado Hardware project.

A handle is created and any call to the driver layer is translated to this handle internally.

A general function 'sitalDevice_AccessMemory(..)' is being used widely by the CAN API to access the driver layer's read and write to registers and memory sections.



August 2020

5 Vivado Project

5.1 Vivado project Top modules

1. SnS CAN Module – SnS Sital Transceiver is used in Testing System as SnS Receiver



2. Three CAN Modules CAN/CAN FD Sital Transceivers are used in Testing System as CAN/CAN FD Transmitters (Sequencers are configured)





August 2020

5.2 Vivado project Output/Input Ports

Diagram × Address Editor × my_constr.xdc ×
E:/prj_EG1720/project_1.srcs/constrs_1/new/my_constr.xdc
Q, 🔐 ←, → Ϫ 🖻 🛍 🗙 // 🎟 ♀
<pre>1 set_property PACKAGE_PIN AC9 [get_ports "TxCan0_out"] ;#[get_ports {JX1_HP_DP_03_P}]</pre>
2; set_property_IOSTANDARD_LVCMOS18 [get_ports "TxCan0_out"];
/ set property DACKAGE DIN Y7 [get ports "D1Can0 in"] .#[get ports (JV1 HD DD 01 D)]
5 set property IOSTANDARD LVCMOS18 [get ports "R1Can0 in"]; #[get_ports [bai_ir_br_oi_r]]
6 'set property PACKAGE PIN AA7 [get ports "R2Can0 in"] :#[get ports (JX1 HP DP 01 N)]
7 set property IOSTANDARD LVCMOS18 [get ports "R2Can0 in"];
8 set property PACKAGE_PIN AC8 [get ports "R3Can0 in"] ;#[get ports {JX1 HP DP 03 N}]
9 set_property_IOSTANDARD_LVCMOS18 [get_ports "R3Can0_in"] ;
10 set_property PACKAGE_PIN AB7 [get_ports "RhCan0_in"] ;#[get_ports {JX1_HP_DP_00_N}]
<pre>11 set_property IOSTANDARD LVCMOS18 [get_ports "RhCan0_in"] ;</pre>
<pre>12 set_property PACKAGE_PIN AB8 [get_ports "RlCan0_in"] ;#[get_ports {JX1_HP_DP_00_P}]</pre>
<pre>13 set_property IOSTANDARD LVCMOS18 [get_ports "RlCan0_in"] ;</pre>
14
15 set_property PACKAGE_PIN AB2 [get_ports "TxCan1_out"] ;#[get_ports {JX1_HP_DP_11_P}]
16; set_property IOSIANDARD LVCMOSI8 [get_ports "TxCan1_out"];
1/ '
19, set property IOSTANDARD LVCMOS18 [get ports "P1(an1 in"] .
20 ·
21 set property PACKAGE PIN H4 [get ports "TxCan2 out"] :#[get ports (JX2 HP DP 03 P)]
22 set property IOSTANDARD LVCMOS18 [get ports "TxCan2 out"];
23
24 set property PACKAGE_PIN A3 [get ports "R1Can2_in"] ;#[get ports {JX2_HP_DP_01_P}]
<pre>25 set_property IOSTANDARD LVCMOS18 [get_ports "R1Can2_in"] ;</pre>
26
27 set_property PACKAGE_PIN_E6 [get_ports "TxCan3_out"] ;#[get_ports {JX2_HP_DP_11_GC_P}]
<pre>28 set_property IOSTANDARD LVCMOS18 [get_ports "TxCan3_out"];</pre>
29
30; set_property PACKAGE_PIN C6 [get_ports "R1Can3_in"];#[get_ports {JX2_HP_DP_09_P}]
31 set_property IOSIANDARD LVCMOSI8 [get_ports "RICan3_in"];
32
33 1



August 2020

5.3 Vivado project Top modules Addresses

Diagram × Address Editor ×	my_constr.xdc	×			
Q ★ ♦ 📾					
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
∨ ‡ zynq_ultra_ps_e_0					
✓ III Data (40 address bits : 0x0080)	000000 [512M])				
stICAN_0/axi_bram_ctrl_0	S_AXI	Mem0	0x00_8000_0000	64K 🔹	0x00_8000_FFFF
stICAN_1/axi_bram_ctrl_0	S_AXI	Mem0	0x00_8001_0000	64K 🔹	0x00_8001_FFFF
stICAN_2/axi_bram_ctrl_0	S_AXI	Mem0	0x00_8002_0000	64K 🔹	0x00_8002_FFFF
stICAN_3/axi_bram_ctrl_0	S_AXI	Mem0	0x00_8003_0000	64K 🔹	0x00_8003_FFFF



August 2020

6 Testing Schem and Environment



6.2 Board, Transceivers, Network

Testing system consists of

- Evaluation board Avnet UltraZed-EG SOM with UltraZed_EG IO Carrier Card
- One Sital PMOD SnS CAN/CAN FD Transceiver (used as receiver only)
- Three Sital PMOD CAN/CAN FD Transceivers (used as transmitters only)
- CAN/CAN FD Network (changeable configuration)
- Relays switch for disconnections simulation





August 2020

Please, use the following picture for right placement of SnS Transceiver – on the left bottom.



6.3 Terminal Program

Tera Term is used as a Terminal program for Xilinx SDK environment. Terminal ID is VT100 and Terminal size must be more than 80x24.

Tera Term: Terminal setup		×
Terminal size	New-line	OK
100 X 38	Receive: CR 🗸	
✓ Term size = win size Auto window resize	Transmit: CR ~	Cancel
Terminal ID: VT100 ~ Answerback:	□ Local echo □ Auto switch (VT<->T	Help EK)



August 2020

7 Test Application

7.1 **Program Description**

First, sitalCan_DeviceInit() function should be called for every device (CAN IP Core) to be used with baseAddress, Low - baseRate and High – dataRate parameters.

Then, sitalCan_OpenDevices() function should be called only once for all devices

The following fragment sets Sequencer for CAN FD Message from deviceId1 -

sitalCan_EnableSequencers() function should be called for certain device to Enable(Disable) all the sequencers configured for this device.

One of the initialized CAN devices can be determine as SnS Receiver device using th following function – sitalSnS_Initialize(deviceId0).

The following fragment (simplified) means the core is continuously receiving LEARNING_MSG_NUMBER messages – for the Learning Phase and TRACKING_MSG_NUMBER messages – for the Tracking Phase. Following every 1,000 messages, a dot '.' is printed.



```
Sital Technology
                                                                                  August 2020
for (int count=0; count< LEARNING_MSG_NUMBER; count++)
{
        while(1)
        {
                 swResult = sitalCan_GetNextMsg(deviceId0, &msgReceived, &RecMsg);
                 if (sitalReturnCode_SUCCESS != swResult)
                 {
                         xil_printf("\r\n get swResult = %d \r\n", swResult);
                         return swResult;
                 if (msgReceived) break;
        }
        if (count%1000 == 0)
                 xil_printf(".");
        sitalSnS_MsgLearning (deviceId0, &RecMsg, &msgIdIndex,
                                                                    &calculatedDeviation,
                                  &bParticipation);
}
xil printf(" \n\r Learning phase is completed");
xil_printf(" \n\r Please, run script of disconnects simulation. Sleep for %d seconds", sleepSeconds);
sleep(sleepSeconds);
for (int count=0; count< TRACKING_MSG_NUMBER; count++)
{
        while(1)
        {
                 swResult = sitalCan_GetNextMsg(deviceId0, &msgReceived, &RecMsg);
                 if (sitalReturnCode_SUCCESS != swResult)
                 {
                         xil_printf("\r\n get swResult = %d \r\n", swResult);
                         return swResult;
                 }
                 if (msgReceived) break;
        }
        if (count%1000 == 0)
                 xil_printf(".");
        sitalSnS_MsgTracking (deviceId0, &RecMsg, &msgIdIndex,
                                  &apprSnSEntryIndex, &calculatedDeviation);
}
xil printf(" \n\r Tracking phase is completed");
sitalSnS FaultIsolation (deviceId0, &msgNonProblemNumber, MsgNonProblemArray,
```

&msgProblemNumber, MsgProblemArray);



7.2 Test Results

The following results are also copied to TerminalPrint.txt file wich is included in the Delivery package.

1. Filter functionality check –

133 293275 8 12 23 34 45 56 67 78 89 11111444 303276 1 cc 133 313277 8 12 23 34 45 56 67 78 89 No Message 11111444

2. Wiring Fault functionality check -

After Learning phase -

MessageID	Period	Fault NonP	ts Tail	H/L	BusE	Lea Count	rned Par	am	Occurr Count	Par	am	Recycl Count	er Param
11111111	10000	0	0	0	Ø	25	f1 14	0 0	0	0 0	0 0	0	0 0
12222222	10000	0	0	Ø	0	25	f3 20	0 0	0	0 0	0 0	Ø	0 0
333	10000	0	0	0	Ø	25	f3 1d	0 0	0	0 0	0 0	Ø	0 0
444	10000	0	0	Ø	Ø	25	f3 1c	0 0	0	0 0	0 0	Ø	0 0
Learning	phase :	is com	nplete	ed									





After Tracking phase -

lessageID	Period	Faul† NonP	ts Tail	H/L	BusE	Learned Count Para	m	Occuri Count	- Par	am	Recyc] Count	er Param
11111111	10000	1	250	Ø	0	25 f1 14	0 0	250	1f 49	0 0	Ø	0 0
122222222	10000	1	250	Ø	0	25 f3 20	0 0	250	f9 2c	0 0	Ø	0 0
333	10000	1	250	Ø	0	25 f3 1d	0 0	250	f9 28	0 0	Ø	0 0
444	10000	1	250	Ø	0	25 f3 1c	0 0	250	f9 29	0 0	Ø	0 0
Tracking	phase i	is com	nplete	ed								
Wiring pr Message I 11111111 Message I no more	oblems Ds - p 122 Ds - r Message	 proble 222222 no pro e ID	ems ap 2 oblems	opear	ed 333	444						
Please, r Please, t	un scri o stop	ipt of scrip	f file ot of	e sav disc	ing. \$1 onnects	eep for 15 : simulation	secon Slee	ds p for	15	secon	ds	



3. Authentication fault functionality check

The Learning phase has exactly the same result. After Tracking phase -

MessageID	Period	Fault NonP	s Tail	H/L	BusE	Lear Count	ned Par	am	Occurr Count	- Par	am	Recyc] Count	.er Param
11111111	10000	Ø	0	0	Ø	247	f1 15	0 0	Ø	0 0	0 0	0	0 0
12222222	10000	223	35	0	Ø	324	f4 21	0 15	35	f0 15	0 0	0	0 0
333	10000	0	0	0	Ø	247	f3 1c	0 0	0	0 0	0 0	0	0 0
444	10000	Ø	0	Ø	Ø	247	f4 1d	0 0	Ø	0 0	0 0	0	0 0
Tracking	phase	is com	nplete	ed									
Authentic Message 12222222 Message 11111111	cation [Ds - 2 [Ds - L	proble proble no pro 333	ems ems ap oblems }	opear	ed 444	no more	e Me	ssag	ie ID				
Please, r	un scr	ipt of	file	e sav	ing. Sl	leep for	15	sec	onds				

Some Message of the origin, where Message ID 0x11111111 was sent from, is sending now by the origin of the Message with ID 0x12222222 –

MessageID	Period	Fault NonP	ts Tail	H/L	BusE	Learned Count Param	Occurr Count Param	Recycler Count Param
11111111	10000	Ø	Ø	Ø	0	247 f1 0 15 0	000	0 0 0
12222222	10000	223	35	0	0	J24 f4 0 21 15	35 f0 0 15 0	0 0 0



August 2020



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.