OneEye_UART_Shell_1 for KIT_AURIX_TC297_TFT Shell over UART using OneEye

AURIX[™] TC2xx Microcontroller Training V1.0.0



Please read the Important Notice and Warnings at the end of this document



Demonstrate how to implement the OneEye shell over the UART (USB) interface. A Shell is used to parse a command line and call the corresponding command execution.

After configuring the OneEye UART interface, a OneEye shell is used to interpret and manage commands like "info" or "help".



Introduction

- OneEye is a GUI that enables the creation of interactive Graphical User Interface. Graphical elements can be drag from a toolbox and drop onto the GUI. The behavior of the created GUI can be customized. Different communication interfaces like UART, Ethernet, CAN, DAS can be used to interact with the embedded system
- SyncProtocol / ProtocolBB is a synchronous protocol that enables data streaming between the target microcontroller and OneEye. It enables to open multiple communication channels, provide packet acknowledge and packet checksum. Data are transported within a message with a message ID and a message payload. See the OneEye help for more information.

Single frame							
Offset	0	0 1		3			
0	Start Byte	Sender	Receiver	Payload length			
4	Flags (frame	:Type=data)	Checksum payload	Checksum header			
8	Messa	age ID	(Reserved)				
12		Messag	e length				
16	Message payload						
	(Message payload)						

Note: It is recommended to go through some of the basic tutorials listed in the help embedded in OneEye (Menu: Help -> OneEye help). This enables a quicker ramp-up in the OneEye concept and ensures a nice journey with OneEye



Hardware setup

This code example has been developed for the board KIT_AURIX_TC297_TFT_BC-Step.

The board should be connected to the PC through the USB port 1





Configuration overview

In this configuration a shell running on the microcontroller is connected to the COM port. In OneEye, two signals **bb.in** and **bb.out** are used to connect the COM port data stream to the BB protocol. The BB protocol is configured to open a channel reserved for the shell. This channel connects to the lineEdit and textEdit with the **console.in** and **console.out** signals.





Implementation - AURIX

Enabling the OneEye library

The OneEye library must be enabled by adding the following line to *lfx_Cfg.h*: *#define IFX_OE_AL_USE_AURIX_ILLD*

Configuring the UART communication

The UART communication is initialized with the function *initUart()*, which also initializes the BB protocol.

In the infinite while loop, the function *processUart()* executes the SyncProtocol.

Configuring the OneEye shell

A OneEye shell (*Ifx_Oe_Shell*) is an object that enables command line parsing and command execution.

The OneEye shell communication interface (*Ifx_Oe_ShellBb*) enables streaming of data using the BB protocol (*Ifx_Oe_SyncProtocol*). The OneEye shell is initialized with *initShell() / Ifx_Oe_Shell_init()*.

The *ifx_oe_shell.h* file can be found in the Libraries\OneEye directory.

Running the shell

The shell is executed in the background loop by calling processShell() / Ifx_Oe_Shell_process().



Run and Test

- After code compilation, flash the device using the Flash button 1 to ensure that the program is running on the device
- For this training, the OneEye application is required for visualizing the values. OneEye can be opened inside the AURIX[™] Development Studio using the following icon:





In this training, the OneEye configuration is provided inside the Libraries folder. The following steps are needed to configure the oscilloscope from a brand-new configuration.

Setup OneEye for editing

Select the OneEye menu "**Options -> Edit mode**" (if not already checked) to enable the edit mode. Select the OneEye menu "**View -> Browser box**", "**View -> Property box**", "**View -> Tool box**" (if not already checked) to display the browser, property box, and tool box. Note that the box can be moved around.

OneEye			-	
File Options View Help				
Tool box	8×		Property box	
Bar Gauge	^			
Clock Gauge				
Combo Box				
Graph				
Graph Channel	× >			
All	~ ~			
Path Object Titl signals layout				
> interface interface. configuration debugBox				
	<u>></u>			



Removing the default DAS interface

When the OneEye configuration is created by ADS, it is already setup with a DAS interface. Select the interface in the Browser box 1 and delete it with "right click and remove" as it is not required in this example.





Configuring the UART interface: Signal creation

The first step is to create two signals to connect the received and transmit data over the UART.

Create a signal group and set its **name** property to **bb**.

OneEye	- 0	X
File Options View Help		
Browser box & X	Property box &	×
All ~		^
	Signal: group	
Path Object Inte	name bb	
bb eroup	type group	
layout	offset 0x0	
configuration	access read-write V	
	model-update-value-enabled	
	title	
	user-data	
	send-get-request	
	on-updated	
	include-path	
	include-as-read-only	
< >	Enum values	
Tool box Browser box	Match events	~
OneEye Version 2.48.0 937, 465		



Add two signals of type **char** into the **bb** group, name them **in** and **out**, and set their **title** property to respectively **BB in** and **BB out**.

OneEye			— C) X
File Options View Help				
Browser box	ē ×	Property box		₽×
All	~			^
		Signal:	char	
Path Object Title		name	in	
signals		type	char	
in <u>char</u> BB in		offset	0x0	
out char BBout		access	read-write	~
layout		model-undate-value-enabled		
		title	BBin	
		une dete		_
		user-data		
		send-get-request		
		on-updated		
		Enum values		
		Enum values		
		Match events		
		Connections		
Tool box Browser box				¥
OneEye Version 2.48.0 937, 465				



Configuring the UART interface: COM port

Right click in an empty area of the Browser box, and select **Add child -> Interface**. Then right click on the created interface and select **Add child -> com**. Select the **com** item and set its **device** property to the COM port connected to the AURIX board. Set the **baudrate** property to **115200** and click **connect**.

The COM port is now opened and ready for communication.

OneEye			- 0	\times
File Options View Help)			
Browser box		ē ×	Property box	8 ×
All Path Signals bb layout	Object	Title	Plugin Com Port: com plugin PCom_Core::com device COM17 baudrate 115200	~
Tool box Browser box	COM.COM17 COM.COM17	>	connect	



Configuring the UART interface: Transmit stream

Right click on the **interface** in the Browser box, and select **Add child -> dataMessageHandler**. Then right click on the created **dataMessageHandler** and select **Add child -> message** to create a message item. Configure the **message** with the **id=0xFE**, **interval=0.001**, **send-on-new-data** checked, **dir=tx**, stream checked.

OneEye				- 🗆 X
File Options View Help				
Browser box	8 ×	F	Property box	5 ×
All	~			
			Message:	message
Path Obje	ect Title		id	0xFE
> signals			interval	0.001
layout	LCOM17		send-on-new-data	
✓ dataMessageHandler			dir	tx v
✓ message 0xFE				
field bb.o	out		type	extended ~
✓ message 0xFF			length	0
field bb.ir	n		at man	
com COM	1.COM17		stream	
configuration				
<				
Tool box Browser box				
Offecye version 2.46.0 957,465				



Right click on the **message**, and select **Add child -> field**. Configure the field with **name=bb.out**, **bit-pos=0**, **buffer=512**.

Now, data will be transmitted over the UART each time the **bb.out** signal is written with some data.

OneEye						_		\times
File Options View Help								
Browser box		8 ×	Pro	perty box				8 ×
All All Path > signals layout ' interface ' dataMessageHandler ' message field com configuration	Object Title COM.COM17 0xFE bb.out 0xFF bb.in COM.COM17			Message field: name bit-pos buffer buffer-as-pakcet	field bb.out 512	+	-	
Tool box Browser box								
OneEye Version 2.48.0 937, 465		L	1					



Configuring the UART interface: Receive stream

Right click on the **dataMessageHandler** and select **Add child -> message** to create a second message item. Configure the message with the **id=0xFF**, **interval=-1**, **dir=rx**, stream checked.

OneEye						\times
File Options View Help						
Browser box	5 >	Pro	operty box			8×
All	~	Γ				
			Message:	message		
Path C	Object Title		id	0xFF		
> signals			interval	-1		
layout ✓ interface C	COM.COM17		send-on-new-data			
✓ dataMessageHandler			dir			
✓ message 0:)xFE					
field b	b.out		type	extended	`	~
✓ message	b in		length	0		
com C	COM.COM17		stream			
configuration						-
<	>					
Tool box Browser box						
OneEye Version 2.48.0 937, 465						



Right click on the **message**, and select **Add child -> field**. Configure the field with **name=bb.in**, **bit-pos=0**.

Now each time data are received over the UART, the **bb.in** signal will be updated.

OneEye	- 🗆 X
File Options View Help	
Browser box 🗗 🗙	Property box 🗗 🗙
All	Message field: field
> signals	bit-pos 0
layout V interface COM.COM17 V dataMascaraHandlar	buffer 0
 ✓ dataMessageHandler ✓ message ØxFE field bb.in com COM.COM17 configuration 	buffer-as-pakcet
Tool box Browser box	
OneLye version 2.40.0 551, 405	



Configuring the UART interface: Push button

Drag and drop a **pushButton** widget from the toolbox onto the layout, configure it with **title=Setup Serial** Interface, on-click={show.connection.ui}.

OneEve		plugin PCom_Core::com
File Options View Help		device COM17
Browser box	5 ×	Property box connect
Path Ob > signals > layout	oject Title	id data-in + - data-out + - auto-connect + - title Setup Serial Interface user-data enabled defined-by-connection v visible yes v update-method on-new-data v style-sheet

Clic



Configuring the BB protocol

Right click in an empty area of the Browser box, and select **Add child -> protocolEngine**. Then right click on the created **protocolEngine** and select **Add child -> protocol-core-bb**. Connect the BB protocol stream to the **bb.in** and **bb.out** signals by setting respectively the **data-in** and **data-out** properties. Set the **name** property to **BB-core**. And set the **timeout** to **2000** ms so that frames are dropped after 2 seconds in case the microcontroller is not answering.

OneEye					- 0	×
File Options View Help						
Browser box		₽ ×		Property box		₽×
All		~				
				Plugin protocol co	ore: protocol-core-bb	
Path	Object	Title		name	BB-core	
> signals				plugin	PProtocol Core::bb	
> layout	CON CON17			data ta		
✓ protocolEngine	.BB-core			data-in	DD.IN	_
protocol-core-bb]	data-out	bb.out	
configuration				timeout	2000	
			Setup Serial Interface			
			· · · · · · · · · · · · · · · · · · ·			
<		>				
Tool box Browser box		L				
OneEye Version 2.48.0 937, 472						



Configuring the Shell: signals creation

Create a signal group and set its **name** property to **console**.

OneEye		_	
File Options View Help			
Browser box 🗗 🕽		Property box	ē ×
All		Signal: group	^
Path Object Title		name console	
✓ signals console group		type group	
> bb group		offset 0x0	
> layout > interface COM.COM17		access read-write	~
> protocolEnBB-core		model-update-value-enabled	
configuration	Setup Serial Interface	title	
		user-data	
		send-get-request	
		on-updated	
		include-path	
		include-as-read-only	
		Enum values	
< >>			
	_	Match events	
Tool box Browser box			
OneEye Version 2.48.0 1134, 484			



Add two signals of type **char** into the **console** group, name them **in** and **out**, and set their **title** property to respectively **Console Rx** and **Console Tx**. Set the **access** property of the **in** signal to **read-only** and the **access** property of the **out** signal to **write-only**.





Create the Shell widgets

Drag and drop a **textEdit** widget from the toolbox onto the layout, set the **textEdit** properties **auto-connect** to **console.in**. Set the **update-method** to **all-on-new-data**.

OneEye					_	×
File Options View Help						
Browser box 🗗 🗙	Setup Se	rial Interface	Property box			8×
All Path Object signals layout nushButton Setun SerialInte. textEdit Console Rx interface COM.COM17 protocolEngine .BB-core configuration Configuration	Console Rx		Plugin widget: id data-in data-out auto-connect title user-data enabled visible update-method style-sheet	textEdit) * * *	
Tool box Browser box						
OneEye Version 2.48.0 1030, 465						



Drag and drop a **lineEdit** widget from the toolbox onto the layout, set the **lineEdit** properties **auto-connect** to **console.out**. **Check** the **capture-key** property to enable each key stroke to be send.

OneEye	- 🗆 X
File Options View Help	
Browser box & Setup Setup	Interface Property box & X
All Path Object Title > signals * layout pushBu Setup Serial Inte Setup Serial IntetextEdit Console Rx lineEdit Console Tx > interface COM.COM17 > protocolEn .BB-core configuration Console Rx	Plugin widget: lineEdit id data-in data-out auto-connect title user-data enabled defined-by-connection
	visible yes 🗸
	capture-key 🗹
	format default 🗸
	format-string %1
	format-field-width 0
	format-base 10
Tool box Browser box Console Tx	format-fill-char V
OneEye Version 2.48.0 1134, 484	



Connect the lineEdit and textEdit widget to the BB protocol

Right click on the **protocol-core-bb** and select **Add child -> target**. Select the **target** item and set **local-port** and **remote-port** to **2** to match the AURIX settings, **set signal-in=console.out**, **signal-out=console.in**.

OneEye		-	
File Options View Help			
Browser box & X	Setup Serial Interface	Property box	8×
All		Tarpet: tarpet	
Path Object Title		local-port 2	
> signals > layout		remote-port 2	
> interface COM.COM17		signal-in console.out	
protocolingine .BB-core protocol-core-bb target Signals to neole configuration	Console Rx	signal-out console.in forward send-parent-value	
Tool box Browser box	Console Tx		
OneEye Version 2.48.0 1134, 484		-	



Test the shell interface

Restart the AURIX software. The shell textbox should display the "Hello World !" text **1**.



Enter "info" in the **Console Tx** lineEdit field **2** and press ENTER, the microcontroller executes the *printShellInfo()* function and should answer as below to acknowledge the command.

rowser box	8 X		Setup Serial Interface	Pr	operty box		8	
All Path Signals Idvalue protocolEngine Protocol-core-bb target configuration		Console Rx	Shell>Hello world! info The shell command was called ! Shell>]	Plugin widget: id data-in data-out auto-connect title user-data enabled visible capture-key format	lineEdit + - onsole.out + - onsole.out + - defined-by-connection yes default		
					format-field-width	0	j	
<					format-base	10		

Save your configuration with CTRL+S.

Exit the edit mode with the OneEye menu "Options -> Edit mode" to only see the GUI 3.

OneEy	e			_	\times
File Op	tions	View	Help		
			Setup Serial Interface	•	
	Shel:	l>Hello	world!		
Console Rx	info				
	The s	shell o	command was calle	d!	
	Shel:	1>			
Console Tx					
OneEye Vers	ion 2.4	8.0			

References









- → AURIX[™] Development Studio is available online:
- https://www.infineon.com/aurixdevelopmentstudio
- > Use the *"Import…"* function to get access to more code examples.
- > More code examples can be found on the GIT repository:
- https://github.com/Infineon/AURIX code examples
- > For additional trainings, visit our webpage:
- https://www.infineon.com/aurix-expert-training
- → For questions and support, use the AURIX[™] Forum:
- https://www.infineonforums.com/forums/13-Aurix-Forum

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.



Edition 2022-06 Published by Infineon Technologies AG 81726 Munich, Germany

© 2021 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document? Email: <u>erratum@infineon.com</u>

Document reference OneEye_UART_Shell_1 _KIT_TC297_TFT

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application. For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

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

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.