

AURIX[™] Development Studio

Guide for HighTec Toolchains

Copyright (C) 2023 HighTec EDV-Systeme GmbH Version 1.1, October 2023:

Table of Contents

INTRODUCTION 1
1. Introduction 2
2. Prerequisites
AURIX™ DEVELOPMENT STUDIO 4
3. AURIX™ Development Studio
3.1. Download 5
3.2. Installation and run
HIGHTEC CONFIGURATIONS IN ADS 6
4. External LLVM for TriCore™ applications
4.1. Import existing project
4.2. Update iLLD
4.3. Manage build configuration
4.4. Activate build configuration 10
4.5. Import linker script
4.6. Build settings
4.7. Build the project
5. External GCC for TriCore™ applications
5.1. Documentation
5.2. Import existing Infineon example
5.3. Manage build configuration 20
5.4. Activate build configuration 22
5.5. Compiler-specific options 23
5.6. Linker script
APPENDIX 26
Appendix A: DMA Example
Appendix B: PPU examples 29
B.1. PPU MetaWare build configuration
B.2. TriCore™ external LLVM build configuration
Appendix C: SCR example
C.1. SCR SDCC build configuration
C.2. TriCore™ external LLVM build configuration
Exclude folders from the build

Document references	. 37
Disclaimer	. 38
Document history	. 39

Introduction

1. Introduction

The following application note demonstrates how to configure and build AURIX[™] Development Studio projects with Hightec toolchains.

2. Prerequisites

Prerequisite	Version
Development tool	AURIX [™] Development Studio Limited 1.9.11-L (Limited version for TC4xx) or later
Infineon Low Level Drivers package	iLLD_TC4xx_2_0_1_2_19_Package
HighTec LLVM toolchain for TriCore™ applications	TriCore™ Development Platform version 8.0.0 or higher

Tab. 1. Prerequisites LLVM Toolchain

Prerequisite	Version
Development tool	AURIX™ Development Studio 1.9.16 or later
HighTec GCC toolchain for TriCore™ applications	TriCore™ Development Platform version 4.9.4.1

Tab. 2. Prerequisites GCC Toolchain

AURIX[™] Development Studio

3. AURIX[™] Development Studio

3.1. Download

The AURIX[™] Development Studio can be downloaded from Infineon's pages using the link below.

https://www.infineon.com/cms/en/product/promopages/aurix-development-studio

It is available for the Windows platform as an executable installation file.



Fig. 1. Download of AURIX[™] Development Studio

3.2. Installation and run

The process of installation is straightforward. When it finishes, the main screen appears.

AURIX-v1.5.4-workspace - AURIX Development	t Studio	_		\times
<u>File Edit Navigate Search Project Debug V</u>	<u>V</u> indow <u>H</u> elp			
🔛 🕼 🔎 🔤 🤌 🕸 🕶 🔗 🕶 📃 🍙 🛞 🔘	⑧ ▣ 월 ▼ 祠 ▼ や ♂ ↓ → → ▼		Q	i 🗈 🖬
C/C++ Projects × here Project Explorer				
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	45 8			
🙆 Quick Links 🗵				
First Steps	*			
Create new AURIX Project				
Import AURIX Project	Console × Properties Problems		•	▼ □

Fig. 2. Initial screen of the AURIX[™] Development Studio

HighTec configurations in ADS

4. External LLVM for TriCore[™] applications

Please ensure that you make all the necessary changes to the project before running it. Running the project without implementing all the changes can result in a corrupt build and may require starting the entire project from scratch. To avoid any issues, thoroughly review and implement all the proposed modifications before proceeding with the execution of the project. Please pay special attention to the **C/C++ Indexer**, see further down for more information.

4.1. Import existing project

To test if the LLVM HighTec compiler works correctly, let's import one of the existing Infineon projects.

Select File→Import→Infineon→AURIX[™] Development Studio Project and click [Next].

Minport		×
Select		Ľ
Select an import wizard:		
type filter text		
 > & General > > C/C++ > Infineon AURIX Development Studio Project > Install > Run/Debug > Team > Other 		
< Back Next > Finish	Ca	ncel

Fig. 3. Import Infineon example

Then select **Infineon TC4xx Code Examples Repository**, select an example of your choice (we will test the "blinky" example for STD Kit TC499) and click **[Finish]**.

Select a Code Examples reposit	orv	Repository root		
Infineon Code Examples Reposi	tory ~			Bro
Search Code Examples				
Blinky TC397				
elect a project to import	Abstract		Boards	Last Upo
Blinky_LED_1_KIT_TC397_TF	An LED is blinking be wait function.	ased on the timing given by a	APPLICATION KIT TC3X7 V2.0,	18.12.20

Fig. 4. Select an example from the list (search is used as a filter)

4.2. Update iLLD

Update project libraries to the latest version and back up the current. Right-click on the project, select **Project updater**->Update iLLD and click [Finish].

When the iLLD is updated, ADS executes the **C/C++ Indexer**. It needs to be finished before starting the build. Otherwise, there is a significant risk of the build failing.



Fig. 5. C/C++ Indexer execution in the lower-right corner

4.3. Manage build configuration

To create a new HighTec LLVM toolchain configuration, right-click on the project and select **Build Configuration > Manage**.

AURIX-v1.5.4-workspace - AURIX Development Stu	dio					-		×
File Edit Navigate Search Project Debug Wind	ow	Help						
🗟 🐚 ≽ 🏂 🔸 🔻 🛷 🕶 🗐 🗿 🕐 🕻	12	▼∛▼∜≎⇒≎▼					Q	i 😭 😼
🚳 C/C++ Projects 🗴 🍋 Project Explorer								- 0
수 수 🎯 📄 🔩 수 국	> @	🖻 🔄 8						
> 😂 Blinky_LED_1_KIT_TC397_TFT [Active - Debug	1	New	>					
> 👺 demo_prj		Go Into						
		Open in New Window						
		Index	>					
		Build Configurations	>	Set Active >				
		Build Targets	>	Manage				
		Build Project Clean Project		Build All Clean All				
		Сору		Build Selected				
Ouick Links ×	B	Paste			-			

Fig. 6. Setting the manage build configuration

The Manage Configurations tab will be opened.

🥵 Blinky_Ll	ED_1_KIT_TC4	99_STD_TRE	3: Manage Cor	nfigurations \times
Configur	Description	Status		
TASKING		Active		
TASKING				
Set Active	e Ne	w	Delete	Rename
			OK	Cancel

Fig. 7. manage configuration tab

Select [New] and give a name to the new configuration. Then select Import predefined, choose TriCore Application → External LLVM → Debug and click [OK].

Name:	HighTec		
Description:			
Copy setting	is from		
◯ Existing c	onfiguration	TASKING TC4xx Debug	,
O Default co	onfiguration		,
O Import fro	om projects	not selected	,
🔾 Import pr	edefined	TriCore Application > External LLVM > Debug	,

Fig. 8. new configuration

4.4. Activate build configuration

To select a new active configuration, right-click on the project and select **Build Configuration** \rightarrow **Set Active** \rightarrow **HighTec (name of the new configuration created)**.

🚳 AURIX-v1.5.4-workspace - AURIX Developm	nent St	tudio				-		×
File Edit Navigate Search Project Debug	Win	idow Help						
🗑 🐚 ≽ 🎘 🥕 🕸 🕶 🔗 🕶 🗐 🗿 🌘): ?	■ 図 ▼ 初 ▼ や ジ や ▼	> •				Q	8
🚳 C/C++ Projects × 🍋 Project Explorer								
	5 ¢- 1	-> @ 🖻 🔄 🕴						
> 😂 demo_prj [Active - External GCC - Debu	ig]	Neu		1				
		New	/					
		Go Into		-				
		Open in New Window		-				
		Index	>	Cat Asting		1 Dahua		
		Build Configurations	>	Manage	,	2 External GCC	Debug	
		Build Project		Puild All	_	3 External GCC	Release	
		Clean Project		Clean All		4 Release		
\blacksquare Quick Links $ imes$	e	Сору		Build Selected				
First Steps	i i i	Paste						
Create new AURIX Project	^	Move						
Manual Import AURIX Project		Rename						
O Flash & Start Project	<u>229</u>	Import						
Product Documentation	4	Export		-			_	
TC3xx User Manual part1 V2.0	<u>ي</u>	Set Active Project Refresh		Problems		B. 🔠 🕬 🖻	⊑ ▼ [9
TC3xx User Manual part2 V2.0		Close Project		leted.				^
TC39x B-step User Manual Appendix V2.0	0	Run As	>					
TC39x BD-step Data Sheet Addendum V1.7	*	Debug As	>					
TC39x BD-step Data Sheet V1.2		Compare With	>					
TC3x7 Application Kit User Manual V2.0		Restore from Local History						
TC39x BD-step Errata Sheet V1.6	68	Edit Project Metadata						
TC3xx Architecture vol1 V1.2.2	*	Run C/C++ Code Analysis	AH 5 1					>
/demo_prj	_	Properties	Alt+Enter	1				

Fig. 9. Setting the active build configuration

4.5. Import linker script

After the iLLD update, the project linker script is replaced by a default one for the iLLD base project. If the linker script is not present after creating a new configuration, it is possible to get it from the latest version of the iLLD package and manually copy it into the project:

Target	Latest iLLD
ТС3хх	iLLD_TC3xx_V1_0_1_17_0 _Package
TC4xx	iLLD_TC4xx_2_0_1_2_19_Package

Infineon Low-Level Drivers are available upon request from Infineon and contain a standardized linker script for LLVM toolchains.

The base TC49x linker script **Lcf_Hightec_Tricore_Tc.IsI**, and specialized linker scripts for the PPU and DMA examples, can also be downloaded via this link: HighTec Linker Scripts.

Blinky_LED_1_KIT_TC499_COM_TRB
> 🎉 Binaries
> 🔊 Includes
> 🔁 Configurations
> 🔁 HighTec
> 📂 Images
> 🔁 Libraries
> 🖻 Blinky_LED.c
> h Blinky_LED.h
> 💼 Cpu0_Main.c
> 尾 Cpu1_Main.c
> 尾 Cpu2_Main.c
> 尾 Cpu3_Main.c
> 尾 Cpu4_Main.c
> 尾 Cpu5_Main.c
🗋 changelog.md
Lcf_Hightec_Tricore_Tc.lsl
README.md
Fig. 10. Imported Linker File

4.6. Build settings

The build settings must be updated for the active build configuration. To access the settings, right-click on the project and select **Properties** \rightarrow **C/C++ Build** \rightarrow **Settings**.

4.6.1. Apply toolchain

Under the **Tool Settings >Settings** tab, change the Prefix and Path fields:

- The Prefix is always "empty" for the LLVM toolchain.
- The Path must be set to the "bin" folder of the LLVM toolchain.

ype miler text	seungs		~·~~
 Resource AURIX Development Studio Builders C/C++ Build Build Variables 	Configuration: HighTec [Active]	v]	Manage Configuratio
Environment Logging Settings Tool Chain Editor C/C++ General	 Tool Settings Puild Steps Puild Ar Settings LUVM C Compiler Dialect 	tifact 📓 Binary Parsers 🧿 Error Parsers TC1.8.0 instruction set	
Project Natures Project References Run/Debug Stitings WikiText	Preprocessor Path Peth Preprocessor Path Optimization Debugging Warnings Miscellaneous UVM C++ Compiler Preprocessor Includes Optimization Debugging Warnings Miscellaneous Symbols Miscellaneous Symbols ULVM Linker General Usings Utive Distance Symbols Miscellaneous Utive Distance UVM Object Copy ULVM Object Copy ULVM Optimize General ULVM Optimize ULVM Optimize Sultive Optimized Litting General ULVM Print Size General Sentral ULVM Print Size General Sentral	C\HIGHTEC\toolchains\llvm\HighTec\toolchains\tricore\RC8.0.0V3\bin	Browse
		Restor	e Defaults Appl

After the change is done, click [Apply and Close].

Fig. 11. Setting the prefix and path to the HighTec LLVM compiler

When the new toolchain is set, ADS executes the **C/C++ Indexer**. It needs to be finished before starting the build. Otherwise, there is a considerable risk of the build failing.



Fig. 12. C/C++ Indexer execution in the lower-right corner

4.6.2. Clang driver options

The C/C++ compiler driver clang is a complete control program for a large part of the toolchain. It can orchestrate the entire build of an executable with a single command-line invocation.

In ADS, the compiling and linking steps are done separately. For this reason, the following options must be used in both: LLVM C Compiler options and LLVM Linker options.

Architecture: -march=<arch>

In compilation, it affects the available instructions to emit. In linking, it affects the selection of target libraries. Allowed values are:

<arch></arch>	Device
tc161	AURIX™ platform 1st generation (TC2xx)
tc162	AURIX™ platform 2nd generation (TC3xx)
tc18	AURIX [™] platform 3rd generation (TC4xx)

For compilation, set the field **-march** in **LLVM Compiler**→**AURIX Settings**, i.e., **tc18**. For the linking, set the field **-march** in **LLVM Linker**→**General**, i.e., **tc18**.

Errata: -merrata=<bug>

Some derivatives contain silicon bugs, also known as errata. In such cases, the workaround has to be applied to avoid triggering them. This option also affects the selection of target libraries during the linking. Allowed values:

<bug></bug>	Description
cpu141	[CPU_TC.141] Instructions not implemented in TC49A derivative [1]

Please refer to the Errata chapter in [2] for more details.

For compilation, extend the Other flags field in **LLVM Compiler→Miscellaneous**, i.e., - **merrata=cpu141**.

For the linking, extend the Command field in LLVM Linker, i.e., clang -merrata=cpu141.

Floating point strategy: mfloat-abi=<float-abi>

Select the floating point handling strategy for code generation: software function calls or hardware FPU instructions. The <float-abi> keyword has the following pattern:

<float-strategy><double-strategy><size-of-double>, i.e., hs64.

The letters represent the handling strategy for **float** and **double** floating-point types. The number is the size of the **double** type in bits. The possible combinations can be selected from the following table according to the **-march=<arch>**

<float-abi></float-abi>	float strategy	double strategy	double size	Supported archs	Default for
ss32	function calls	function calls	32	tc18	
hh32	FPU instructions	FPU instructions	32	tc161, tc162, tc18	
ss64	function calls	function calls	64	tc18	
hs64	FPU instructions	function calls	64	tc161, tc162, tc18	tc161, tc162
hh64	FPU instructions	FPU instructions	64	tc18	tc18

For more details, please refer to the Multilib variants chapter in [2].

For compilation, extend the Other flags field in LLVM Compiler→Miscellaneous, i.e., -mfloat -abi=hh64

For the linking, extend the Command field in LLVM Linker, i.e., clang -mfloat-abi=hh64

Exceptions: -f[no-]exceptions

Enable or disable the support for C++ exception handling. This flag also controls which libraries are linked. By default, exceptions are enabled.

For compilation, extend the Other flags field in **LLVM Compiler Miscellaneous**, i.e., **-fno-exceptions**.

For the linking, extend the Command field in LLVM Linker, i.e., clang -fexceptions

The exhaustive list of the compiler driver options can be found in the Compiler Driver chapter in [2].

4.6.3. Compiler-specific options

• The Command is pre-set to **clang** in **LLVM C Compiler**.

Optimizations: -0<level>

The compiler accepts the following optimization options: -00, -01, -02, -03, -0fast, -0s, -0z, -0g

Choose from the offered options or set the Other optimization flags field in **LLVM Compiler** →**Optimization**. In the second case, there are two optimization options in the build command, and the compiler will use the last one in the sequence, i.e., in **clang -00 -Ofast**, **-Ofast** will be used.

type filter text	Settings		> + ⇔ + 8
> Resource AURIX Development Studio Builders C/C++ Build Puild Meinbles	Configuration: HighTec [Active]	✓ Manage G	onfigurations
Environment	🐌 Tool Settings 🎤 Build Steps 🗧	P Build Artifact 🔛 Binary Parsers 🧕 Error Parsers	
Settings Tool Chain Editor > C/C++ General Project Natures	Settings Co LLVM C Compiler Dialect Preprocessor	Immediate Immediate <t< th=""><th>•</th></t<>	•
Project References Refactoring History > Run/Debug Settings	Includes Optimization Debugging Working		v
WikiText	Miscellaneous Co	mmand s(COMMAND) \$(FLAGS) \$(OUTPUT_FLAG) \$(OUTPUT_PREFIX)\$(OUTPUT) \$(INPUTS) \$(EXTRA_FLAGS)	
	Construction C		
		Restore Defaults	Apply
0		Apply and Close	Cancel



Certain options, like **-fstrict-volatile-bitfields** in **LLVM C Compiler AURIX**, are not supported by the LLVM toolchain. These options must be unchecked for the successful compilation of the project. The exhaustive list of the compiler options can be found in the Compiler chapter of [2].

4.6.4. Linker-specific options

- The Command is pre-set to **clang** in **LLVM Linker**.
- Although clang understands and forwards most of the common linker options, some need to be passed directly to the linker. To pass an option to the linker, you should prefix the option with **-Xlinker** or **-Wl**,.

For example, option --gc-sections should be passed to the linker as -Xlinker --gc-sections or -Wl, --gc-sections.

Linker script: -T<linker-script>

We use the linker script containing the Hightec keyword. The information on how to select and import it is described in chapter Linker script.

Set the field Linker Script in **LLVM Linker** General, i.e., .../Lcf_Hightec_Tricore_Tc.1s1.

Libraries

Internal libraries

When using the clang compiler driver, the library paths are chosen based on the **-march**, **-merrata**, **-mfloat-abi**, and **-f[no-]exceptions** options. The libraries like C standard library, or C++ library, are automatically linked from these paths. Some functions from the C standard library reference the file IO functions. To prevent the linker error: **ld.lld: error: undefined symbol**, the **libhtcos.a** library has to be linked as well.

Add the option **-1htcos** in **LLVM Linker-Miscellaneous-Add**.

External libraries

When linking external libraries, it is necessary to pass the library name with **-1** prefix: **-1**<**lib**> as well as the library path with **-L** prefix: **-L**<**libpath**>. To link, e.g., the "C:\tricore\libs\libbsp.a" library, the following parameters have to be passed to the linker: **-1bsp** and **-LC:\tricore\libs**.

Add options -1<1ib> and -L<1ibpath> in LLVM Linker→Miscellaneous→Add to link an external library.



Fig. 14. LLVM Linker Options

HIGHTEC

Certain options like **-nostartfiles** in **LLVM Linker General**, are not supported by the LLVM toolchain. These options must be unchecked for the successful linking of the project. An exhaustive list of linker options can be found in the Linker chapter of [2].

4.6.5. LLVM Object Copy options

• **llvm-objcopy** is pre-set.

An exhaustive list of Binutils options can be found in the Binutils chapter of [2].

	Settings	Q * Q *
Resource ARRO Development Studie Builders (Crc-s Build Build traibles Environment Logging Settings Tool Chain Editor Crc-s General Project Ibdurenses Relationing History Reur/Debug Settings WikiText	Settings Configuration: HighTe: [Active] Deloct Settings: # build Steps Image: Settings: # build Addict: Image: Settings: Connand:	v v v v v v v v v v v v v v v v v v v
	AURC stemps BAURC stemps BAURC stemps BAURC stemps BAURC stemps Bring Bring Bring BAURC stemps BAURC stemps BAURC stemps BAURC stemps BAURC stemps Compared stemps Co	

Fig. 15. LLVM Object Copy Options

4.6.6. LLVM Create Listing options

• **llvm-objdump** is pre-set.

Durce IX Development Studio ders ++ Build Juild Variables rivitonment	Configuration: HighTec [Active]			
ttX Development Studio ders ++ Build luild Variables invironment	Configuration: HighTec [Active]			
luild Variables			 Manage Co 	infiguration
penten lor Billing, LDD, LDT, LDT, LDT, LDT, LDT, LDT, LDT	🛞 Tool Settings 🎤 Build Steps 🍨 Build A	utifact 📓 Binary Parsers 🥹 Error Parsers		
9	Settings Command:	llvm-objdump		
ditor	W LLVM C Compiler All options: Bialect Preprocessor	-h-5"8linky_LED_1,VIT_TC499_COM_TR8.elf"		4
nces story ttings	Includes Optimization Debugging Warnings Expert setting	2		,
	Miscellaneous Command	SIC COMMAND) SIELAGS, SICHTERITE ELAGI, SICHTERITE		
	Sufficiences Service Services Services			
			Restore Defaults	Apply



An exhaustive list of Binutils options can be found in the Binutils chapter of [2].

4.6.7. LLVM Print Size options

• **llvm-size** is pre-set.

er text	Settions	6+0
	our and a second s	
X Development Studio lers + Build	Configuration: HighTec [Active]	✓ Manage Configuration
uild Variables svironment	🛞 Tool Settings 🎤 Build Steps 😨 Build Artifact 📓 Binary Parsers 🧿 Error Parsers	
ttings	Settings Command: Ilvm-size	
ool Chain Editor ++ General cct Natures cct References	V (b) LUMC Complet All options formet=beneley *Binky_LED_1_VT_TC499_COM_TR8.eff* B Preprocessor Includes	
ctoring History Debug Settings	Optimization Debugging Winners Event cattions	
Debug Settings Text	Command C	
	Vorter - Complete Marconson - Complete Morcellencon	
		Restore Defaults Ap

Fig. 17. LLVM Print Size Options

An exhaustive list of Binutils options can be found in the Binutils chapter of [2].

4.7. Build the project

Now, the project will use the HighTec LLVM compiler to build the application.

The final project content after a successful build is shown in the below figure (the copied linker file and the generated **.elf** file).



Fig. 18. Content of the imported Infineon project after a successful build

A complete project, run on a board.

Ne têt Debug Show News Toels Config Neip	Ne tot Naviget Seach Poject Debug Window Help	
12.26年後日後,11年8月,		Q. (#) 🔂
* ** 14 伊 行 伊 行 川 米 1g Conditatedy wat 一 四 為 巴 Conditative Conditative Conditative Conditative Conditative	C/C++ Popert X & Popert System ** 🖬 📷 Lef (Highter, Trans, T. M. X	
後後日口口(D) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1		
Conditioned + a x CLUBicondition Sec. Max x CLUBiconductures CLUBouther/MCauthers CLUBouthers CLUBouther/MCauthers CLUBouthers CLUBouthers CLUBouther/MCauth	V LUME VOM 1 KIT YEAR COM INI [Astro-Hapter]	
V: // internationact, proving to continuent, river	200 1	1
 Hoderfan (Oran Hoderfan (Oran<th>Society and Sectors Society and Sectors Society and Sectors</th><th></th>	Society and Sectors Society and Sectors Society and Sectors	
Texture (Control (Contro) (Control (Contro) (Contro) (Contro) (Contro) (Contro) (> > > > > > > > > > > > > > > > > > >	
 * Buildow productively hosts to escote Safety liberry tests */ * Index Buildow (See Safety 1) 	Source State State <t< th=""><th></th></t<>	
anapility and particular and an anapility and an	> (Gal)Matc 111 (
termine and constructional (1) // Call main function of CF00 */	> is GreijManc 316)> deema AD prize > is GreijManc 317 Deemannamenta 317	
folder of TFL CFL 200 SETURE FIOR SAIS *) B BATYCORA 072 C	
 The for tage(based) and and tage to an function of ON +/ 	B throughing 000 128(2002)(1,1000) 0 to for New York Table 001 120(2012)(1,1000)	
weath >+ IPA_th_gost_arrowreal_said +-	B. U. (etc.), E.D. (etc.), E	
* Go into infinite loop, normally the code shall not reach here */ INLEWinfinite(our):	Ld_being_Nov_hbd 000 *1.000m_pbg.(*)	
•0	8 MOLECTER 2015	
static voidCStartBasic(void)	347 SECTORS	
• 1 / The fait in the user stack printer */	200 CPULARAE IST DEFINITION	
 Inc_ind_satestreaming(sile); Its_best(0000;); 	Constant a second secon	
/*Initialize the declarat area and the COD. Function Calls Foundation of the Second	The Rest Part of Control (1, 2004)	
Fig.Dev_init(Old(testigned ini *)Old(0), (testigned ini *)Old_D00(01))	Control con ADD Papers 393 1 - edition - epical	
• J	C Import ALBOX Project 2016 (2) /	
	Entry Property in	
Teest vector address	Conceller Decementation (2012)	· · · · · · · · · · · · · · · · · · ·
rif defined(_TRRUD_)	C Tare 610 2013 11 Release Nates	
Transa point a start"	WALC WOLD WALCHMALD PERCH	
/recent operation		
<pre>// provide dealine is definedclass)</pre>		
ANTI ACTAS _ ANT_ A ATTAC _ TATAC _ TATAC		
from solid		
foreign extra concentration of the second se		
<pre>/Point definedphs_1 //popus_texts.itert' //p</pre>		
Jeed 1		
vedd_TTATT(vedt)		
e Ingergenetitigen (de Columbia) (PRSearthfootmare):		
To defined		
Trease police relation to the second se		
A CALL A		
2. Two Task Target Source Ressan		
13 Success 15:15:17 Govet Widewhopfwareer Program with 10 but - code size 49865 by		
4.1 Valuess 15.11.17. Ganed UNDebugheever Program with 12.04.1 - understand 0105.19 4.1 Samma 15.11.17. Ganed UNDEbugheever Program with 10.04.1 - understand 0105.19		
41 Success 15:15:47 Core5 UEEnhopferver Program with 10 0al - code size 43165 by		
041 Ento 15:15:55. Geneo PF1438 Programming with special (enture based) in 44 Revenue 15:15:55. Geneo PF1438 Filoshier events methods science and filos		
✓45 Success 15:15:02. ComeO PTASM Textir so differences found (125 so).	0.0	3 6 M
	-	
MANNUC CPC thest after download fulled in memory report how ARXIVOR1 to ARXIVITY1	Webbie Inset 1110	
		DIS on of the SITPL
🖵 yangan 🧰 📮 🔤 🔤		V V QI 🖉 (11.0020 0

Fig. 19. UART_VCOM_1_ example run on board TC400_COM_TRB

5. External GCC for TriCore[™] applications

5.1. Documentation

The configuration of an external GCC toolchain is described in the documentation of the AURIX Development Studio.

Choose Help→Help Contents, and navigate to AURIX Development Studio User Guide→Tasks→External Toolchains→Configuring and external GCC Toolchain.



Fig. 20. Navigation inside Help Content window

5.2. Import existing Infineon example

To test if the HighTec compiler works correctly, let's import one of the existing Infineon projects.

Select File > Import > Infineon AURIX Development Studio Project and click Next.

de Import		×
Select	Ľ	1
Select an import wizard:		
type filter text		
 > See General > See C/C++ > See Infineon AURIX Development Studio Project > See Install > See Run/Debug > See Team > See Other 		
< Back Next > Finish	Cancel	

Fig. 21. Import Infineon example

Then select an example of your choice (we will test the "blinky" example for Application Kit TC397).

🍓 Import AURIX Development Stu	idio Project			×
Select an AURIX Developmen	t Studio Project to import			
-Select a Code Examples repository	Repository root			
Infineon Code Examples Repositor	y ~		B <u>r</u> o	wse
Search Code Examples				
Blinky TC397				
Select a project to import		1	l of 349 P	roject
Name	Abstract	Boards	Last Upo	lated
Blinky_LED_1_KIT_TC397_TFT	An LED is blinking based on the timing given by a wait function.	APPLICATION KIT TC3X7 V2.0,	18.12.202	20
Description of Blinky_LED_1_KIT_TO A wait function is used to add delay	2397_TFT /s between switching on and switching off an LED on	port pin P13.0.		
	< <u>B</u> ack <u>N</u> ext >	<u> </u>	Cancel	

Fig. 22. Select an example from the list (search is used as a filter)

5.3. Manage build configuration

To create a new HighTec GCC toolchain configuration, right-click on the project and select **Build Configuration**->**Manage**.

🙆 AURIX-v1.9.16-workspace	- AUI	RIX Development Studio					– 🗆 X
File Edit Navigate Search Project Debug Window Help							
🔛 🕼 🔁 🕶 🌌 🕶 🔑 🛛		J 🗿 🌰 🎯 🛄 🤊 😐 🛂	V	🎋 🕶 🛷 🕶 🖗 🔻 🖗	- *:-		ର 🖻 🔂
🛃 C/C++ P 🗙 🌇 Project	E						- 0
	E	3 🔩 8					
> 🎏 Blinky_LED_1_KIT_TC39	07 TE	TIActive					
		New	,				
		Go Into					
		Open in New Window					
		Index	>				
		Build Configurations	>	Set Active	>		
		Build Targets	>	Manage			
		Build Project		Build All			
		Clean Project		Clean All			
🚳 Quick Links 🗙		Сору		Build Selected			
First Steps	ß	Paste					
🚳 Create new AURIX Proje	×	Delete					
🚳 Import AURIX Project		Nove					
🎯 Flash && Start Project		Kendriken					
Product Documentation	2	Import					
🔞 TC3xx User Manual part		Ехроп					
🔞 TC3xx User Manual part		Set Active Project					
🔞 TC39x B-step User Man	de la construcción de la constru	Refresh					
🔞 TC39x BD-step Data She		Close Project					
🔞 TC39x BD-step Data She	0	Run As	>				
TC3x7 Application Kit U	夺	Debug As	>				
TC39x BD-step Errata SF		Ieam Compare With	>				
TC3xx Architecture vol1		Restore from Local History	,				
TC3xx Architecture vol2	aß.	Edit Project Metadata					
/Blinky_LED_1_KIT_TC397_TFT	98	Open documentation					
	-						•

Fig. 23. Setting the manage build configuration

The Manage Configurations tab will be opened.

Blinky_LED_1_KIT_TC397_TFT: Manage Configurati					
Configuration	Description	Status			
Debug		Active			
Release					
Set Active	New Dele	ete Rename			
	ОК	Cancel			

Fig. 24. manage configuration tab

Select [New] and give a name to the new configuration. Then select Import predefined, choose TriCore Application \rightarrow External GCC \rightarrow Debug and click [OK].

C/C++ P × 1 Project E □	- (6 Create New Configurat	ion	×	-
Blinky_LED_1_KIT_TC397_TFT [Active		Note: The configuration n Please ensure that it is vali	ame will be used as a directory name in the file sy d for your platform.	stem.	
		Name: HighTec			
		Description:		_	
		Copy settings from			
		 Existing configuration 	Debug	~	
	E	 Default configuration 		~	🚽 📃 + 📑 + 🗖
	No	 Import from projects 	not selected		
Quick Links X		 Import predefined 	TriCore Application > External GCC > Debug	~	
First Steps Image: Steps Image: Create new AURIX Project Image: Steps Image: Steps <			OK Can	cel	
🐵 Flash && Start Project					
Product Documentation (1) TC3xx User Manual part1					
TC3xx User Manual part2					
🔞 TC39x B-step User Manual Appendi					
🚳 TC39x BD-step Data Sheet Addendι					
TC39x BD-step Data Sheet					
TC3x7 Application Kit User Manual					
🔞 TC39x BD-step Errata Sheet					
TC3xx Architecture vol1					
TC3xx Architecture vol2					

Fig. 25. new configuration

5.4. Activate build configuration

To select a new active configuration, right-click on the project and select **Build Configuration** \rightarrow **Set Active** \rightarrow **HighTec (name of the new configuration created)**.

AURIX-v1.9.16-workspa	ace - A	URIX Development Studio							-		J	×
File Edit Navigate Sea	rch	Project Debug Window Help										
🔛 🕼 🏲 🕶 🔀 🕶 🌽	0	0 🗗 🕥 🔘 🛄 🕐 🕒 📘	8	🏇 🕶 🥜 🕶 🔮 📼		5 -	୬ 🗇 ▾ 🖒 ▾			Q	: 😭	Ec
🛃 C/C++ P 🗙 🍋 Proj	ect E.	8									-	
$\langle \neg \neg \rangle$	Q	🖻 🔄 8										
 Blinky_LED_1_KIT_TC3 M Includes 	907 T	New	>									
> 🔁 Configurations		Go Into										
> Blinky_LED.c		Open in New Window										
> 🔓 Blinky_LED.n > 💽 Cpu0_Main.c		Index	>									
> Cpu1_Main.c		Build Configurations	>	Set Active	>		1 Debug					
> Cpu2_Main.c		Build Targets	>	Manage		~	2 HighTec					
> Cpus_Main.c		Build Project		Build All			3 Release	📄 🖳 🚮 📝 📑	<u> </u>	- 📬	• •	
		Clean Project		Clean All								_
🛃 Quick Links 🗙		Copy		Build Selected								^
	TÊ.	Paste										
First Steps	×	Delete										
Create new AURIX Pr		Move										
Import AURIX Projec		Rename										
🎯 Flash && Start Proje	-	los a set										
Product Documentation	24	Import										
🔞 TC3xx User Manual p	<u> </u>	Export										
TC3xx User Manual p		Set Active Project										
TC39x B-step User M	S.	Refresh										
TC39x BD-step Data		Close Project										
TC39x BD-sten Data	0	Run As	>									
TCD.7 Application K	*	Debug As	>									
TC3x7 Application Ki		Team	>									
TC39x BD-step Errata		Compare With	>									
TC3xx Architecture v	_	Restore from Local History										-
TC3xx Architecture v	68	Edit Project Metadata										Þ
/Blinky_LED_1_KIT_TC397_TF	-	Open documentation										10
		and the second sec										

Fig. 26. Setting the active build configuration

5.5. Compiler-specific options

5.5.1. Path to the compiler

The path and name prefix of the Hightec compiler must be updated for the active build configuration.

Right-click on the project and select **Properties**→**C/C++ Build**→**Settings**. Change the **Prefix** and **Path** fields under the **Tool Settings** tab. The prefix is always "tricore-" for AURIX devices; the path might be different.

Properties for demo_prj	i				_		×
type filter text	Settings						▼ 8
 > Resource > AURIX Development Builders > C/C++ Build 	Configuration: External GCC - De	ebug [Active]		~	Manage Co	nfigurations	.
Environment	🛞 Tool Settings 🎤 Build Steps	ዋ Build Artifa	ct 🗟 Binary Parsers 😣 Error Parser	S			
Logging Settings Tool Chain Editor > C/C++ General Project Natures Project References Refactoring History > Run/Debug Settings WikiText	 Settings Settings AURIX GCC Compiler Dialect Preprocessor Includes Optimization Debugging Warnings Miscellaneous 	Instruction set Prefix Path	TC1.6.2 instruction set tricore- C:\HighTec\toolchains\tricore\v4.9.4	I.1\bin	B	rowse	
< >	AUKIX Settings	1		Apply a	nd Close	Cancel	-

Fig. 27. Setting the prefix and path to the HighTec GCC compiler

Now, the project will use the HighTec GCC compiler to build the application.

5.5.2. Include paths

Right-click on the project and select **Properties** \rightarrow **Aurix Development Studio** \rightarrow **Build** check the box to autodiscover compiler include paths.

🚳 AURIX-v1.9.16-	ALIDIV Davidance C	A collection of the second		
File Edit Naviga	Properties for Blinky_LED_1_F	(IT_TC397_TFT	— 🗆 X	
	type filter text	Build	← → ⇒ %	۹ 🖻 🗖
C/C++ P × Blinky_LED_1 > Configura > Libraries > Blinky_LEE > Blinky_LEE > Blinky_LEE > Blinky_LEE > Cpu0_Mai > Cpu2_Mai > Cpu3_Mai	 > Resource > AURIX Development Studio AURIX Build Booster Build > C/C++ Build > C/C++ Build > C/C++ General Project Natures Project References Refactoring History > Run/Debug Settings WikiText 	Auto-discover compiler include paths		
> 🖻 Cpu4_Mai > 尾 Cpu5 Mai				📑 🗝 🗖
🛃 Quick Links 🗙				
First Steps Create new A Final Magnet AURI Filash && Sta				
Product Docume				
TC3xx User N TC3xx User N	?		Apply and Close Cancel	
🔞 TC39x B-step	User Manual Appendi			
🔞 TC39x BD-ste	ep Data Sheet Addendı			
TC39x BD-ste	ep Data Sheet			
TC3x7 Applic	ation Kit User Manual			
TC39x BD-ste	ep Errata Sheet			
TC3xx Archit	ecture vol1			
M TC3xx Archit	ecture vol2			
/Blinky_LED_1_KIT_TC	397_TFT			1

Fig. 28. Include paths

5.5.3. GCC linker options

Right-click on the project and select **Properties**→**C/C++ Build**→**Settings**, click on AURIX GCC Linker. Under **Command line pattern**, change:

From: \${COMMAND} \${FLAGS} \${OUTPUT_FLAG} \${OUTPUT_PREFIX}\${OUTPUT} \${INPUTS}

To:**\${COMMAND}** -nocrt0 **\${FLAGS} \${OUTPUT_FLAG} \${OUTPUT_PREFIX}\${OUTPUT} \${INPUTS}**

Click on Apply and Close.



Fig. 29. GCC linker options

5.6. Linker script

Before the imported Infineon example can be built with the HighTec compiler, the Lcf_Gnu_Tricore_Tc.lsl linker script file must be copied into the newly imported Infineon project.

You can find the linker script here: HighTec Linker Scripts.

After the linker script is in place, the project is ready to be built.

Appendix

Appendix A: DMA Example

This guide contains instructions for how to successfully compile the **DMA_Mem_to_Mem_1_KIT_TC499_COM_TRB** project using the HighTec LLVM toolchain version 8.0.0. It assumes that the project has been imported into the current workspace.

The following changes are required before the first project build attempt:

- Wait until the indexer finishes.
- Right-click on the project, then **Project updater→Update iLLD→Finish**.
- Right-click on the project and click on **Refresh**.
- Wait until the indexer finishes.
- Add Lcf_Hightec_Tricore_Dma.lsl linker script adapted specifically for the DMA use case.

The following changes are required to build the project:

- - Give any name
 - Select the option Import predefined, then TriCore Application→External→LLVM→Debug and click [OK].
- Set the newly created configuration to active by clicking on [Set Active].
- Wait until the indexer finishes.
- Edit the build settings: right-click on the project Properties→C/C++ Build→Settings
 - Click on **Settings** and Update Path to the LLVM bin directory even though it is not used later on, i.e., "C:\HighTec\toolchains\tricore\v8.0.0\bin" and click on **[Apply]**.
 - Click on LLVM Linker, extend the command string with -merrata=cpu141 to clang -merrata=cpu141 and click on [Apply].
 - Click on LLVM Linker->General and change Linker Script name to .../Lcf_Hightec_Tricore_Dma.lsl and click on [Apply and close].
- Update the file DMA_Mem_to_Mem.c:

```
From:
```

```
/* DMA Source buffer for DMA transfer stored inside DSPR0: 0x70000000 */
uint32 g_dataForDmaTransfer[DATA_ARRAY_LENGTH] __at(0x70000000);
```

```
/* DMA Destination buffer stored inside DLMU RAM (of CPU0): 0xB0000000 */
uint32 g_dmaLmuDestination[DATA_ARRAY_LENGTH] ___at(0xB0000000);
```

To:

#if defined(__TASKING__)
/* DMA Source buffer for DMA transfer stored inside DSPR0: 0x70000000 */
uint32 g_dataForDmaTransfer[DATA_ARRAY_LENGTH] __at(0x70000000);
/* DMA Destination buffer stored inside DLMU RAM (of CPU0): 0xB0000000 */
uint32 g_dmaLmuDestination[DATA_ARRAY_LENGTH] __at(0xB0000000);

#elif defined(__HIGHTEC__) && defined(__clang__)
#pragma clang section bss=".dma.dspr0bss"
/* DMA Source buffer stored inside .dma.dspr0bss section */
uint32 g_dataForDmaTransfer[DATA_ARRAY_LENGTH];
#pragma clang section bss=".dma.dlmu0bss"
/* DMA Destination buffer stored inside .dma.dlmu0bss */
uint32 g_dmaLmuDestination[DATA_ARRAY_LENGTH];
#pragma clang section bss=""

#else
#error "Unknown compiler"
#endif

• Build the project.

Appendix B: PPU examples

This guide contains instructions for successful compilations of the following projects using HighTec LLVM toolchain version 8.0.0:

- iLLD_TC49xA_ADS_PPU-IPC
- iLLD_TC49xA_ADS_PPU-IRQ
- iLLD_TC49xA_ADS_PPU-SIMD
- iLLD_TC49xA_ADS_PPU-STU

It assumes that one of the projects has been imported into the current workspace.

The following changes are required before the first project build attempt:

- Wait until the indexer finishes.
- Right-click on the project, then **Project updater Update iLLD Finish**.
- Right-click on the project and click on Refresh.
- Wait until the indexer finishes.
- Add the Lcf_Hightec_Tricore_Ppu.lsl linker script specifically adapted for the PPU use case.

B.1. PPU MetaWare build configuration

The following changes are required to build the PPU project for the existing PPU Debug (Ext. MetaWare) build configuration:

- Right-click on the project, then Build Configurations -> Set Active -> PPU Debug (Ext. MetaWare).
- Right-click on the project, then **Properties**→**C/C++ Build**→**Settings**→**Settings**. Update MetaWare Installation Path to, i.e., "C:\HighTec\toolchains\mwaurix\v0.2.0" and click on [Apply and close].
- Click on the project, then Properties -> C/C++ Build -> Behavior. Disable parallel build by unchecking the "enable parallel build" checkbox, then click on [Apply and close]. Optionally, configure the option "use optimal jobs" to "use parallel jobs" and set the value to be lower or equal to the physical cores of the used laptop. This is not a fail-proof option, and one might encounter a compiler crash during the build.

One by one, right-click on the following directories/files, then Resource Configuration→Exclude from build→Select your build configuration and click [OK] (some may already have been disabled).
 /Libraries/iLLD/TC49A/CpuGeneric/Asclin
 /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/Tricore
 /Libraries/Infra/Ssw/TC49A/Csrm, /Libraries/Infra/Ssw/TC49A/Tricore
 AllowAccess.c, PpuInterface.c
 Cpu0_Main.c, Cpu1_Main.c, Cpu2_Main.c, Cpu3_Main.c, Cpu4_Main.c, Cpu5_Main.c

• Build the PPU project.

B.2. TriCore[™] external LLVM build configuration

The following changes are required to build the Tricore project part:

- - Give any name
 - Select the option **Import predefined**, then **TriCore Application**→**External**→**LLVM**→**Debug** and click [OK].
- Set the newly created configuration to active by clicking on [Set Active].
- Wait until the indexer finishes.
- Edit the build settings: right-click on the project **Properties** >C/C++ Build > Settings
 - Click on **Settings** and Update Path to the LLVM bin directory even though it is not used later on, i.e., "C:\HighTec\toolchains\tricore\v8.0.0\bin" and click on [Apply].
 - Click on LLVM Linker, extend the command string with -merrata=cpu141 to clang -merrata=cpu141 and click on [Apply].
 - Click on LLVM Linker > Miscellaneous > Add 1htcos. Then click on [Ok] and [Apply].
 - Click on LLVM Linker-General and change Linker Script name to .../Lcf_Hightec_Tricore_Ppu.lsl and click on [Apply and close].
- Right-click on the project, then Properties > AURIX Development Studio > AURIX Build Booster
 > Libraries paths > Delete /Libraries/iLLD if present and click on [Apply and close].
- Right-click on the project, then Properties AURIX Development Studio AURIX Build Booster
 Ignore paths Add the following folders and click on [Apply and close].
 /Libraries/iLLD/TC49A/ArcEV, /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/ArcEV
 /Libraries/Infra/Ssw/TC49A/Csrm
 /PPU
- Right-click on the project, then Properties C/C++ General Paths and Symbols Includes GNU C, and ensure that the include paths do not contain any of the following directories and their subdirectories, click on [Apply and close].
 /Libraries/iLLD/TC49A/ArcEV, /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/ArcEV
 /Libraries/Infra/Ssw/TC49A/Csrm
 /PPU
- One by one, Right-click on the following directories, then Resource Configuration→Exclude from build →Select your build configuration and click [OK] (some may already have been disabled).
 /Libraries/iLLD/TC49A/ArcEV, /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/ArcEV
 /Libraries/Infra/Ssw/TC49A/Csrm
 /PPU
- Change the definition of uncached data/bss sections in Cpu0_Main.c due to the warning: unknown pragma ignored [-Wunknown-pragmas]

 Change uncached.lmubss section definition (applies to iLLD_TC49xA_ADS_PPU-IPC, iLLD_TC49xA_ADS_PPU-IRQ, and iLLD_TC49xA_ADS_PPU-STU)
 From:

```
#pragma section all "uncached.lmubss"
/* Declaration of global uninitialized variables */
#pragma section all
```

To:

```
#if defined(__HIGHTEC__) && defined(__clang__)
#pragma clang section bss=".uncached.lmubss"
#else #pragma section all "uncached.lmubss"
#endif
/* Declaration of global uninitialized variables */
#if defined(__HIGHTEC__) && defined(__clang__)
#pragma clang section bss=""
#else
#pragma section all "uncached.lmubss"
#endif
```

 Change uncached.Imudata section definition (applies to iLLD_TC49xA_ADS_PPU-STU): From:

```
#pragma section all "uncached.lmudata"
/* Definition of global initialized variables */
#pragma section all "uncached.lmudata"
```

To:

```
#if defined(__HIGHTEC__) && defined(__clang__)
#pragma clang section data=".uncached.lmudata"
#else
#pragma section all "uncached.lmudata"
#endif
/* Definition of global initialized variables */
#if defined(__HIGHTEC__) && defined(__clang__)
#pragma clang section data=""
#else
#pragma section all "uncached.lmudata"
#endif
```

 Change the following code in AllowAccess.c due to the warning: non-portable path to file; specified path differs in case from file name on disk [-Wnonportable-include-path]: From:

```
#include "IfxCpu_Cfg.h"
#include "IfxGtm_Cfg.h"
#include "IfxEgtm_Cfg.h"
#include "IfxQspi_Cfg.h"
#include "IfxCan_Cfg.h"
#include "IfxAsclin_Cfg.h"
#include "IfxSent_Cfg.h"
#include "IfxGpt12_Cfg.h"
#include "IfxI2c_Cfg.h"
#include "IfxGeth_Cfg.h"
```

HIGHTEC

#include "IfxCpu_cfg.h"
#include "IfxGtm_cfg.h"
#include "IfxEgtm_cfg.h"
#include "IfxQspi_cfg.h"
#include "IfxCan_cfg.h"
#include "IfxAsclin_cfg.h"
#include "IfxSent_cfg.h"
#include "IfxSpt12_cfg.h"
#include "IfxI2c_cfg.h"
#include "IfxGeth_cfg.h"

• Build TriCore™ project.

To:

Copyright © 2023 HighTec EDV-Systeme GmbH

Appendix C: SCR example

This guide contains instructions for how to successfully compile the **iLLD_TC49x_ADS_SCR_Blinky_LED_1** project using the HighTec LLVM toolchain version 8.0.0. It assumes that the project has been imported into the current workspace.

The following changes are required before the first project build attempt:

- Wait until the indexer finishes.
- Right-click on the project, then **Project updater Update iLLD Finish**.
- Right-click on the project and click on **Refresh**.
- Wait until the indexer finishes.

C.1. SCR SDCC build configuration

The following changes are required to build the SCR project with the SDCC toolchain:

- Right-click on the project, then **Build Configurations→Manage→New**
 - Give any name.
 - Select the option Import predefined, then SCR Application→External→SDCC→Debug and click [OK].
- One by one, Right-click on the following directories, then Resource Configuration→Exclude from build →Select your build configuration and click [OK] (some may already have been disabled).
 Configurations, Libraries
 CPU0_Main.c, CPU1_Main.c, CPU2_Main.c, CPU3_Main.c, CPU4_Main.c, CPU5_Main.c
 SCR_AURIX_TC4x.c
- Edit the build settings: right-click on the project Properties >C/C++ Build > Settings
 - SDCC Compiler
 - Update Command with the full compiler path to:
 - "C:\HighTec\toolchains\scr\v1.0.0/bin/scr-cc" -c -mtc4xx-scr --model-large -MMD
 -o "SCR/main.rel"
 - SDCC Linker
 - Update Command with the full compiler path to:
 "C:\HighTec\toolchains\scr\v1.0.0/bin/scr-cc" -mtc4xx-scr --model-large --xram
 - -size 0x400 -o 'aurix_scr.ihx' ./SCR/main.rel
 - Update Command Line pattern to: **\${COMMAND}**
 - Update SDCC Object copy CArray command with the full compiler path to: "C:\HighTec\toolchains\scr\v1.0.0/bin/scr-objcopy" -I ihex -0 binary 'aurix_scr.ihx' 'aurix_scr.bin'
 - Update Command Line pattern to: **\${COMMAND}**
- Build the SCR project.

C.2. TriCore[™] external LLVM build configuration

The following changes are required to Build the Tricore project part:

- Right-click on the project, then **Build Configurations→Manage→New**
 - Give any name
 - Select the option Import predefined, then TriCore Application→External→LLVM→Debug and click [OK].
- Set the newly created configuration to active by clicking on [Set Active].
- Wait until the indexer finishes.
- Edit the build settings: right-click on the project Properties >C/C++ Build > Settings
 - Click on **Settings** and Update Path to the LLVM bin directory even though it is not used later on, i.e., "C:\HighTec\toolchains\tricore\v8.0.0\bin" and click on [App1y].
 - Click on LLVM Linker, extend the command string with -merrata=cpu141 to clang -merrata=cpu141 and click on [Apply].
 - Click on LLVM Linker > Miscellaneous > Add 1htcos. Then click on [Ok] and [Apply].
- Right-click on the project, then Properties > AURIX Development Studio > AURIX Build Booster
 > Libraries paths > Delete /Libraries/iLLD if present and click on [Apply and close].
- Right-click on the project , then Properties AURIX Development Studio AURIX Build Booster
 Ignore paths, add the following folders, and click on [Apply and close].
 /Libraries/iLLD/TC49A/ArcEV
 /Libraries/iLLD/TC49A/Csrm
 /Libraries/iLLD/TC49A/Scr
- Right-click on the project, then Properties >C/C++ General >Paths and Symbols >Includes >GNU C, ensure that the Include paths do not contain /Libraries/iLLD/TC49A/Scr or its subdirectories, and click on [Apply and close].
- One by one, Right-click on the following directories, then Resource Configuration→Exclude from build
 →Select your build configuration and click [OK] (some may already have been disabled).
 /Libraries/iLLD/TC49A/ArcEV
 /Libraries/iLLD/TC49A/Csrm
 /Libraries/iLLD/TC49A/Scr
 /SCR
- Change the following code in Cpu0_Main.c: From:

```
if (PMS_SCR_CON0.B.SCREN == 0)
{
    IfxPmsPm_MemoryConfig memCfg = {&scr_xram, PMS_XRAM, SIZE_scr_xram};
    IfxPmsPm_initScr(IfxPmsPm_ScrBootMode_userMode, FALSE, &memCfg);
}
```

To:

HIGHTEC

```
#if defined(__HIGHTEC__) && defined(__clang__)
   /*
           _____
-----
    * Include the binary file of the SCR program into the TriCore build
*_____
*/
    extern const unsigned char _bin_file_start[];
    extern const unsigned char _bin_file_end[];
    __asm__(
     ".section \".rodata\", \"a\", @progbits\n"
    "_bin_file_start:\n"
    ".incbin \"../../iLLD_TC49x_ADS_SCR_Blinky_LED_1/Debug_SCR/aurix_scr.bin\"\n"
     " bin file end:\n"
    ".previous\n"
    );
#endif
    if (PMS_SCR_CON0.B.SCREN == 0)
    {
#if defined(__TASKING__)
        IfxPmsPm_MemoryConfig memCfg = {&scr_xram, PMS_XRAM, SIZE_scr_xram};
#elif defined(__HIGHTEC__) && defined(__clang__)
        IfxPmsPm_MemoryConfig memCfg = {(void*)&_bin_file_start, PMS_XRAM, (unsigned long
int)((unsigned long int)_bin_file_end-(unsigned long int)_bin_file_start)};
#endif
        IfxPmsPm_initScr(IfxPmsPm_ScrBootMode_userMode, FALSE, &memCfg);
    }
```

• Build TriCore[™] project.

Exclude folders from the build

Some examples contain superfluous folders that can interrupt or contaminate a build.

Building such examples results in compiler errors like: error: unknown type name 'sint64', error: unknown type name 'uint64' or error: use of undeclared identifier 'uint64'.

The following changes are required to build these projects for LLVM Build configuration:

- Right-click on the project, then Properties > AURIX Development Studio > AURIX Build Booster
 > Libraries paths, delete /Libraries/iLLD if it is present, and click on [Apply and close].
- Right-click on the project , then Properties AURIX Development Studio AURIX Build Booster
 Ignore paths, add the following folders, and click on [Apply and close].
 /Libraries/iLLD/TC49A/ArcEV, /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/ArcEV
 /Libraries/Infra/Ssw/TC49A/Csrm
- Right-click on the project, then Properties→C/C++ General→Paths and Symbols→Includes→GNU C, ensure that the Include paths do not contain any of the following directories and their subdirectories, click on [Apply and close].
 /Libraries/iLLD/TC49A/ArcEV, /Libraries/iLLD/TC49A/Csrm, /Libraries/iLLD/TC49A/Scr
 /Libraries/Infra/Platform/ArcEV
 /Libraries/Infra/Ssw/TC49A/Csrm
- One by one, Right-click on the following directories, then Resource Configuration→Exclude from build, select your build configuration, and click [OK] (some may already have been disabled).
 /Libraries/iLLD/TC49A/ArcEV
 /Libraries/iLLD/TC49A/Csrm
 /Libraries/iLLD/TC49A/Scr

Manually excluding these folders doesn't always prevent them from being included again upon rebuilding.

Document references

[1] "AURIX™ TC49x errata sheet, Infineon Technologies AG, v1.16, 2023-08-07"

[2] "TriCore™ Development Platform: User Guide, HighTec EDV-Systeme GmbH, Version 8.0.0, October 2023"

Disclaimer

Please Read Carefully:

This document contains descriptions for copyrighted products that are not explicitly indicated as such. The absence of the TM symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this document.

The information in this document has been carefully checked and is believed to be entirely reliable. However, HighTec EDV-Systeme GmbH assumes no responsibility for any inaccuracies. HighTec EDV-Systeme GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this document or its associated product. HighTec EDV-Systeme GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages that might result.

HighTec EDV-Systeme 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.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may occur without the express written consent from HighTec EDV-Systeme GmbH.

Copyright © 2023 HighTec EDV-Systeme GmbH, D-66113 Saarbrucken.

Document history

Version	Date	Changes to the previous version
1.0	June 2023	Initial version
1.1	October 2023	Changes to accommodate newer version of ADS
1.2	November 2023	Added section: External toolchain GCC



HighTec EDV-Systeme GmbH Europaallee 19, D-66113 Saarbrücken info@hightec-rt.com +49-681-92613-16 www.hightec-rt.com