

# CIC61508 Applets Examples

How to use the CIC61508 Secure SPI Mode Applets Concept

PRELIMINARY

# **Application Note**

CIC61508, V0.9, 2011-06

# FreeTextDocumentTopic



Edition 2011-06

Published by: Hitex (U.K.) Limited. University Of Warwick Science Park, Coventry, CV4 7EZ, UK © 2011 Hitex (U.K.) Limited. All Rights Reserved.

#### Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the product, Hitex (UK) Ltd. 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.

#### Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Hitex Office (<u>www.hitex.com</u>).



	Document Change History				
Date	Version	Changed By	Change Description		
2010-11-21	0.2	M Beach	Initial Keil Version		
2011-03-18	0.9	M Beach	Dave Bench SDCC added		

#### We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your comments (including a reference to this document) to: <u>comments@hitex.co.uk</u>

 $\succ$ 



# **Table of Contents**

1	Introduction	7
1.1	Advanced Hardware Test Functions	7
1.1.1	CIC61508 Applet Runtime Environment	7
2	CIC61508 Applet Examples	8
2.1	Pre-Requisites For Using Examples	8
2.2	CIC61508 Applet Memory Map	9
2.3	Principle Of Operation	11
3	Applets Examples Installation	
3.1	Running The Applets Examples Installer	
3.2	Configuring The Tasking VX TC1782 Toolchain	
3.2.1	Checking the Project Configuration	
4	Using The Dave-Bench SDCC Toolchain	19
4.1	Adapting Dave Bench For CIC61508 Applet Creation	
4.1.1	Setting The CIC61508 Applet Memory Map	21
4.2	Running The Hex To C Converter	
4.2.1	Naming The Hex File Output	23
5	Special Settings Required In Keil VISION	24
6	Using Batchfiles For CIC61508 Applets in SDCC	27
7	Applet Examples In Detail	
7.1	Including CIC61508 Applets In Tricore Applications	
7.2	Example Applets Tricore Program	
7.3	Applet Example 1	
7.3.1	C51 Applet	
1.3.Z	I FICORE DRIVER	
7.3.3	Annlet Example 2	
7.5	Example 2 Tricore Driver	
7.6	Applet Example 3	
7.6.1	Example 3, Part2 Tricore Driver	
8	CIC61508 Secure SPI Mode Driver Library	



# List of Figures

Figure 1	CIC61508 Applet Memory Map	
Figure 2	CIC61508 Applet Development Cycle	
Figure 3	DAVE Bench SDCC Memory Map Page	21
Figure 4	DAVE Bench SDCC Build Steps Page	
Figure 5	DAVE Bench SDCC Output File Naming Page	23
Figure 6	uVISION Target Tab	24
Figure 7	uVISION Output Tab	
Figure 8	Interfacing the Hex 2 C-array converter	
Figure 9	Specifying the linker control file	
i iguie a		



# List of Tables

Table 1	Applets Memory Areas	9
---------	----------------------	---



### 1 Introduction

The CIC61508's secure SPI mode is an alternative SPI protocol for programming the DFLASH calibration data area and advanced diagnostics. It is available in NOTREADY and DISABLED mode and is entered after writing 0x94 to the MODE SFR and using a password. It provides the following facilities:

- 1. DFLASH read, write, erase, programming
- 2. Read and write of IDATA and XDATA RAM
- 3. Read of PFLASH
- 4. Jump to address
- 5. Reset CIC61508

It is intended primarily for production-line use and can only be left via device reset.

### 1.1 Advanced Hardware Test Functions

For software development and production line programming of the DFLASH, these facilities are sufficient. However for low-level board testing during development and in the early stages of production test, additional functions are sometimes required. Due to limited ROM space in the CIC61508, a special RAM area has been reserved into which users can download simple C51 programs to control CIC61508 IO during manufacture and test. These programs can be used to implement common hardware test functions such as reading all the IO pins to check for short circuits or driving the SYSDIS pins to specified patterns to allow connected hardware to be tested.

### 1.1.1 CIC61508 Applet Runtime Environment

- Uses Keil C51 evaluation or Dave-Bench SDCC toolchains
- Eclipse IDE and batchfile compile and link for SDCC tools
- Max code size 256 bytes
- Max data size 128 bytes
- Applets inherit stack and SPI settings from CIC61508
- All IO pins (except SPI) accessible
- Applets can use existing SPI driver
- No interrupts allowed
- IO and memory configuration from CIC61508 normal operation mode is inherited.
- Applets can return to secure SPI mode, if required.

The remainder of this document describes the basic applet examples for the TC1782 evaluation board using the Keil, Dave-Bench and batchfile-based SDCC toolchains. The example applications used in this application note can be downloaded from:

http://www.hitexuk.co.uk/download/cic61508/CIC61508 Applets.zip

The unzip password is "hitexhitex".



# 2 CIC61508 Applet Examples

### 2.1 Pre-Requisites For Using Examples

To run the examples, you will need:

- 1. Tasking Tricore VX v3.3r1 Eclipse compiler
- 2. Keil PK51 evaluation version (free) or Dave Bench v2.01 SDCC tools
- 3. Hex2Carry.exe custom utility to convert C51 HEX386 files to a compilable C-array (supplied).
- 4. TC1782 SafeTkit board
- 5. PLS UDE UAD2 debugger
- 6. Hitex CIC61508 secure SPI mode driver library "CIC\_SecSPI\_Drv.a" for the TC1782 or example driver.
- 7. CIC61508 v2.6 or greater firmware.
- 8. Oscilloscope (not essential)

The supplied installer "CIC61508\_Applets.exe" contains all of these elements except the toolchains. DAVE Bench and Keil PK51 Eval are available for download from www.infineon.com. The Tasking VX Eclipse tools are available for download from www.tasking.com.

The installation of the examples is covered later in this document.



0x0000

### 2.2 CIC61508 Applet Memory Map

The CIC61508 memory available to applets is shown below.



### Table 1Applets Memory Areas

Area	Range
CODE Executable and constar	nts: 0xF100 – 0xF1FF
XDATA variables:	0xF080 – 0xF0FF

This is described to the Keil PK51 compiler via a custom linker control file:



For SDCC, these settings are held in the project settings memory map page or in the case of a batchfile-based system, in the linker command line:

C:/DAVE-Bench-201\SDCC\_XC800\bin\sdcc main.rel SecSPIDefs.rel IO\_ReadTest.rel crtstart.rel crtxinit.rel crtxstack.rel startupxc866.rel --no-xinit-opt --main-return --debug -I"C:\c51dev\CICApps\App2\_SDCC\Common\inc" -I"C:/DAVE-Bench-201\SDCC\_XC800\include" -I"C:/DAVE-Bench-201\SDCC\_XC800\include\xc800" -I"C:/DAVE-Bench-201\SDCC\_XC800\include\xc800" -I"C:/DAVE-Bench-201\SDCC\_XC800\include\asm\xc800" -mXC800 -pXC866\_4FR --model-small --iram-size 0x100 -Wl bBSEG=0x20 --xram-loc 0xF080 --xram-size 0x80 --code-loc 0xF100 --code-size 0x100 --dataloc 0x00 --idata-loc 0x80 --stack-loc 0x80 -Wl -bPSEG=0xF000 -o "App2\_SDCC.ihx"

Provided these setups are used in any applet, the memory map requirements will be met.



### 2.3 Principle Of Operation

The overall applet sequence of operations is set out in the diagram below.



The applet development cycle is:

- 1. Create applet in PK51 or DAVE-Bench/SDCC, using the special linker file or memory map settings.
- 2. Create an Intel 386 format Hexfile.
- 3. Convert the hexfile to a compilable C array using the custom Hex2Carry.exe utility (supplied), called from the uVISION "User" tab or from the DAVE Bench Build-Steps-Pre-Build steps command page.
- 4. Copy the created C51CodeAppX.C and C51CodeAppX.H to the Tricore application's CIC61508\_Applet directory.
- 5. Write a suitable interface to the applet (i.e. driver), including the provided applet loader from the CIC\_SecSPI\_Drv.a library.
- 6. Compile the Tricore application
- 7. Load application into Tricore and run.

The examples provided here use this method but have pre-written applets and Tricore TC1782 drivers.



# 3 Applets Examples Installation

The applet examples programs that execute on the CIC61508 and the TC1782 example driver are supplied in a single installation utility "CIC61508\_Applets.exe".

### 3.1 Running The Applets Examples Installer

Click on the installer .EXE file and follow the on-screen instructions. While you will be given the opportunity to select an alternative installation directory, correct operation can only guaranteed by taking the default "C:\ CIC61508AppletExamples".

The installer will finish by running the Microsoft Visual C redistribution utility. If you already have this present on your PC, the installer will ask if you want to "Remove" or "Repair" the existing version. Please choose "Repair".

The result will be a directory structure as shown below:





This consists of a directory tree for the CIC61508 applet programs and another for the TC1782 driver:



No special installation and DAVE Bench

steps are required for the Keil PK51 SDCC toolchains. However the

Tasking Tricore VX toolchain requires a setup stage, covered in the next section.

### 3.2 Configuring The Tasking VX TC1782 Toolchain

The TC1782 driver application must be imported into the Tasking Eclipse IDE from the directory "C:\CIC61508AppletExamples\TC1782Driver". Start the Eclipse IDE and when prompted for a Workspace, enter this directory name:





Remove this by clicking on the cross to the right of "Welcome" to reveal an empty workspace.

TASKING C/C++ - TASKING VX-toolset for Trid	Core						ارتصار	• ×
<u>File Edit Navigate Search Project Run</u>	Window Help							
	• 🞯 • 🛛 🔞 • 🖉 •	• • • • •	• 5 • 🔅 • 0 •	@• 🛷•	(ii) (ii)	Ē	TASKIN	IG C/
C/C++ Projec 🛛 😓 Navigator " 🗆					- 0	E Outli	ne 🛙	- 8
0 - 0 - 0						An outlin	e is not availa	able.
	Problems 😒 🖬	Console) 🖂 Propert	ies					0
	0 items							
	Description		Resource	Path	Loci	stion ID		Туре
0*			-III					



The TC1782 driver application is supplied as a complete existing project so it can be imported directly into the Workspace by using the File-Import function and

	(T) T/	ASKING	C/C++ -	TASKING	VX-toolse	t for Tr	riCore	
	File	Edit	Navigate	Search	Project	Run	Winde	
		New Open f	File		A	lt+Shift	t+N ▶	
		Close	All		Ch	Ctrl	I+W	
	(127)	Save	-sut		cu	Ch	tl+S	
		Save A	.s			0.0		
	6	Save A	đI		CI	rl+Shif	ft+S	
		Revert						
		Move Renam	ne				F2	
	8	Refrest	h				F5	
		Conve	rt Line Del	limiters T	0		•	
	6	Print				Ctr	rl+P	
		Switch Restart	Workspac	ce			*	
	2	Import	t					
nen selecting General-Existing	4	Export.						Projects Into Workspace:
		Proper	ties			Alt+E	nter	
(T) Import		Exit						
Create new	project	s from an	n archive file	or director	v.			PKS
() Import					Con al		x	
Impart Projects						$\sim$		
Select a directory to search for existing Eclip	ose pro	jects.			T	r	2	
Select root directory: C:\CIC61508Appl	etExar	nples\1	TC1782Dr	river	Bro	wse		
Select archive file			0		Real Pro			
Projects:					<u>Di</u> c	WSEM		
CIC ProdProg TC1782 (C:\CIC61508	Apple	tExam	ples\TC1	782Drive	e Sel	ect All		
CIC_SecSPI_Drv (C:\CIC61508Applet	Examp	oles\TC	1782Driv	ver\CIC_	S	de et é		
					Dese	eect A		
					Re	fresh		ancel
								Browse to the directory and
								click OK. The names of the two Tasking TC1782 projects required will
٠ III.								appear in the <u>P</u> rojects box.
Copy projects into workspace								
Working sets								
Add project to working sets								
Working sets:				-	Seler	+	4	
					Seco		-	
(?) C Back	Vext >	-	Fin	ish		ancel		v0.9, 2011-06
	29-04 ×						9	



Make sure they are selected before clicking Finish.



Two projects will now be present in the workspace.



project "CIC\_ProdProg\_TC1782" is the main project that contains loading and execution of the CIC61508 applets. The project CIC\_SecSPI\_Drv is a sub project that provides the CIC61508 secure mode SPI driver. It is only necessary to build the former project as it will automatically build and link in the sub-project.



#### 3.2.1 **Checking the Project Configuration**

It is necessary to check that the CIC\_SecSPI\_Drv project is configured for the TC1782. This is done via the Build-Configurations menu. Click on the project

name in the C/C++ Projects panel and then right-click to reveal the Build Configurations menu.

TASKING C/C++ -	TAS	GNG VX-toolset for TriCore		the second second	-	-	
<u> </u>	: Se	arch Project Run Window Help	_				
[] - 모 쇼  6 성 - 진 - 박		i i i i i i i i i i i i i i i i i i i	gs .	• 🖻 • 🛍 • 👰 • 🛞 •  🕸	•	0 - (	@ •   🛷 •   🔲 🛙
Ec C/C++ Projec ≥ ▲ S <sup>C</sup> CIC_ProdPro ▷ S <sup>C</sup> CIC_SecS	ф 9_ТС	Sc. Navigator	,	1			
Includes Applets		Go Into					
👂 🔂 Dave Debug		Open in New Window					
<ul> <li>Debugge</li> <li>Docs</li> <li>TARDISS</li> </ul>		Exclude from build Index	•				
⊳ € cstart.c		Build Configurations	•	Set Active		1 Del	bug
▶ 🔓 cstart.h 🗟 CIC_Prod		Make Targets Build Project	•	Build Polete resource cfgs		2 Rel 3 TC	ease 1767
DConfig		Clean Project		Manage		4 TC	1387
MConfig		Clone Project Copy Paste Delete Move Rename				5 TC	1782
	2	Import					
	2	Export		honsole Properties	25)		
	3	Refresh Close Project		red identifier "CIC_SSC0_RSRC" CIC	ourc	e ureMo	Path /CIC_SecSPI_Dry/So
Select	0	Build Documentation Run As Debug As	+	red identifier "CIC_SSC0_RSRC_ CIC red identifier "CIC_SSC0_RB" CIC red identifier "CIC_SSC0_RB" CIC	_Sec _Sec _Sec	ureMo ureMo ureMo	<ul> <li>/CIC_SecSPI_Drv/So</li> <li>/CIC_SecSPI_Drv/So</li> <li>/CIC_SecSPI_Drv/So</li> </ul>
active Build		Profile As	•		ш		
Finally, to		Convert To					
project		Team	+	anto ar	100	-	CONTRACTOR OF
successful,		Compare With Restore from Local History	•				
Project.		Properties Alt+En	ter				

"TC1782" as the

import has been

select Project-Build

Configuration. check that the



## 4 Using The Dave-Bench SDCC Toolchain

The Eclipse-based Dave Bench XC800 toolchain has been preconfigured to compile and link the 3 applet examples. To build the examples, start Dave Bench and set the Workspace to:

C:\CIC61508AppletExamples\ExamplesforCIC61508

After initializing, the Bench will show three projects:

XC800 Development - DAVE-Bench for XC800	
e <u>E</u> dit <u>N</u> avigate Search <u>P</u> roject Debug Tools <u>W</u> indow <u>H</u> elp	
** 🖓 🔊 🕲 👘 🛍 👌 🏄 🖄 • 🖓 • 👘 👘 👘 👘 👘 👘	◎ ダ • 月 • 司 • ゆ • o • PP - 司 • ゆ • o • PP - 司 • ゆ • o •
E/C++ Projects 🛛 👘 🗆	
수 수 형 🖻 🕹 📚 🎽	
App1_SDCC [ Active - Debug ]	
App2_SDCC	
App3_SDCC	
🛞 Active Project Problems 🥅 Console 😚	
Deleted Tile - C:\DAVE-Bench-201	VSDCC XCSUUVexamples/XCSUUVexamplesrorCic615US/ADD1 SDCC/Debud/SecSFigers.adp
Deleted file - C:\DAVE-Bench-201	\SDCC XC800\examples/XC800\ExamplesforCIC61508\App1 SDCC\Debug\App1 SDCC.1kr
Deleted file - C:\DAVE-Bench-201	\SDCC XC800\examples\XC800\ExamplesforCIC61508\App1 SDCC\Debug\App1 SDCC.cdb
Deleted file - C:\DAVE-Bench-201	\SDCC_XC800\examples\XC800\ExamplesforCIC61508\App1_SDCC\Debug\App1_SDCC.map
Deleted file - C:\DAVE-Bench-201	\SDCC_XC800\examples\XC800\ExamplesforCIC61508\App1_SDCC\Debug\App1_SDCC.mem
Deleted file - C:\DAVE-Bench-201	\SDCC_XC800\examples\XC800\ExamplesforCIC61508\App1_SDCC\Debug\App1_SDCC
Deleted file - C:\DAVE-Bench-201	\SDCC_XC800\examples\XC800\ExamplesforCIC61508\App1_SDCC\Debug\App1_SDCC.ihx

The project "App1\_SDCC" will be shown as the active project. To check that the installation

has been successful, select from the top line, Project-Build All (Ctrl +B).



The build operation should complete successfully as shown.



project, the output from the build process will be two source files:

For the App1\_SDCC

"C51codeApp1.C"

PRELIMINARY - Application Note



"C51codeApp1.H"

Organize 🔹 Include in library 👻	Share with 🔻 New folder		38 • 🗇
Favorites	App1_SDCC	Crtxstack.rst	
E Desktop	App1_SDCC.cdb	crtxstack.sym	
Downloads	App1_SDCC.htx	Main.adb	
Recent Places	App1_SDCC.ihx	Main.lst	
	App1_SDCC.lkr	Main.rel	
Desktop	App1_SDCC.map	Main.rst	
📴 Libraries	App1_SDCC.mem	Main.sym	
Documents	C51CODE.TPL	makefile	
A Music	C51codeApp1_SDCC.c	objects.mk	
<ul> <li>Pictures</li> </ul>	C51codeApp1_SDCC.h	SecSPIdefs.adb	
Videos	Crtcall.Ist	SecSPIdefs.lst	
mbeach	Crtcall.rel	SecSPIdefs.rel	
	Crtcall.rst	SecSPIdefs.rst	
Network	Crtcall.sym	SecSPIdefs.sym	
Control Panel	Crtstart.lst	sources.mk	
Recycle Bin	Crtstart.rel	startupxc866.lst	
	Crtstart.rst	startupxc866.rel	
	Crtstart.sym	startupxc866.rst	
	Crtxinit.lst	startupxc866.sym	
	Crtxinit.rel	subdir.mk	
	Crtxinit.rst	Trans.	
	Crtxinit.sym		
	Crtxstack.lst		
	Crtxstack.rel		

These contain the applet as a compilable C const array:

// Flag that this is C51code.c

#define C51CODE

// Include the necessary include files
#include "C51codeApp1\_SDCC.h"

/\* Size of this applet \*/

unsigned int const uiCIC61508 AppletSizeApp1 = 0x32U;

The name of the C array created by Hex2Carray.exe will have a suffix related to the second parameter passed on the Hex2Carray.exe command line. Here it was:

Hex2Carray.exe App1 SDCC App1

These auto-generated files are then included in the TC1782 driver example for download and execution in the CIC61508.



### 4.1 Adapting Dave Bench For CIC61508 Applet Creation

The adaptations made to the default XC800 project settings to allow applets to be created are covered below.

### 4.1.1 Setting The CIC61508 Applet Memory Map

The applets memory map must be entered:

type filter text	Memory Settings Page	(	
type filter text Resource Builders □ C/C++ Build Build Variables Discovery Options Environment MCU Selection Page Segment Location Page Settings Tool Chain Editor ▷ C/C++ General Project References Refactoring History Run/Debug Settings ▷ Task Repository WikiText	Memory Settings Page Memory Settings Internal RAM start address Internal RAM end address Bit segment start address Bit segment end address XRAM start address in hex (0xNNNN) XRAM end address in hex (0xNNNN) Flash/ROML start address in hex (0xNNNN) Flash/ROML and address in hex (0xNNNN) Flash/ROML end address in hex (0xNNNN) Flash/ROML and address in hex (0xNNNN) Flash/ROM2 start address in hex (0xNNNN) Flash/ROM3 start address in hex (0xNNNN) Flash/ROM3 start address in hex (0xNNNN) Flash/ROM3 end address in hex (0xNNNN)	0x00       0xFF       0x20       0x2F       0xf0R0       0xf0FF       0xf100       0xf1FF	
< »		Restore Defaults         Apply	Figure 3 DAVE Bench SDCC Memory Map Page



### 4.2 Running The Hex To C Converter

The Hex2Carray.exe utility is assumed to be in the applet project root directory. Here the HEX file output from the SDCC toolchain is "App1\_SDCC.ihx". The name of the C array created will be appended with the name "App1" (second parameter).

pe filter text	Settings 🔅 🗧	$\Box_{p}^{2} =$
Resource Builders C/C++ Build Build Variables Discovery Options	Configuration: Debug [Active]	irations
Environment MCU Selection Page Memory Settings Page	Tool Settings       Perebuild Steps         Pre-build steps       Generated	
Segment Location Page Settings Tool Chain Editor	copy .\*.tpl Description:	
<ul> <li>C/C++ General</li> <li>Code Style</li> <li>Documentation</li> <li>File Types</li> <li>Indexer</li> <li>Language Mappings</li> </ul>	Copy C51 Code template file	
	Command: \hex2carray App1_SDCC.ihx App1	
Paths and Symbols	Description:	
Refactoring History Run/Debug Settings Task Repository WikiText	Convert hex file to compilable C const array	_
	Restore <u>D</u> efaults	<u>A</u> pply

Figure 4 DAVE Bench SDCC Build Steps Page



### 4.2.1 Naming The Hex File Output

The SDCC tools must be told to create a HEX output with the extension ".IHX":

/pe filter text	Settings	쳦 🔹 🕁 💌
rpe filter text Resource Builders a C/C++ Build Build Variables Discovery Options Environment MCU Selection Page Memory Settings Page Segment Location Page Settings Tool Chain Editor a C/C++ General Code Style Documentation File Types Indexer Language Mappings Paths and Symbols Project References Refactoring History Run/Debug Settings Task Repository WikiText	Settings Configuration: Debug [ Active ] Tool Settings P Build Steps Build Artifact Artifact Type: Artifact name: App1_SDCC Artifact extension: ihx Output prefix:	

Figure 5 DAVE Bench SDCC Output File Naming Page



# 5 Special Settings Required In Keil VISION

The uVISION projects are configured for the XC866-4FR, using the LX51 linker. As supplied, the uVISION IDE has been pre-configured with the necessary special settings. To aid the creation of user's own applets, the relevant settings are shown in the following screen captures:

Infineon XC866-4FR									
	0	Xtal (MHz)	): [20	T Use On-chip RO	M (UXU-UX2F)	FF,UXAUU	U-UKAFFF)		
Memory Model:	Small: variabl	les in DATA	A <u> </u>						
Code Rom Size:	Large: 64K p	rogram		J I✓ Use On-chip XR.	AM (0xF000-	-OxF1FF)			
Operating system:	None		_	<u>.</u>					
				Use multiple DP	<b>FR</b> registers				
- Off-chin Code me	mon			- Off-chin Xdətə məmo	n/				
Off-chip Code me	mory	Start:	Size:	Off-chip Xdata memo	ny	Start:	Size:		
- Off-chip Code me	mory Eprom	Start: 0xF100	Size:	Off-chip Xdata memo	ny Ram Ox	Start: xF000	Size:		
└Off-chip Code me	mory Eprom	Start: 0xF100	Size:	Off-chip Xdata memo	Ram 😡	Start: xF000	Size:		
←Off-chip Code me	mory Eprom Eprom	Start: 0xF100	Size:	Off-chip Xdata memo	Ram Ox Ram Ox	Start: xF000	Size:		
Off-chip Code me	mory Eprom Eprom Eprom	Start: 0xF100	Size: 0x0100	Off-chip Xdata memo	Ram Ox Ram Ram Ram	Start: xF000	Size:		
Off-chip Code mer	Eprom Eprom Eprom	Start: 0xF100	Size: Ox0100	Off-chip Xdata memo	Ram 0x Ram Ram Ram	Start: xF000	Size: 0x0080		
Off-chip Code mer	nory Eprom Eprom Eprom	Start: 0xF100 Start:	Size: 0x0100  . End:	Off-chip Xdata memo	Ram 🕅 Ram 🗍 Ram 🗍 support	Start: xF000	Size: 0x0080		





Create Executive Create Executive Create Executive Create Executive Create UE	able: .\App2 mation V E	Browse Information	Merge32K Hexfile				
	HIE HEX Format: THE	^-300 <u> </u>	1	Offset:			
C Create Library	.\App2.LIB			Create Batch	File		
			Defender		1145	Figure 7 Output Tat	uVISIC o
tions for Target	OK Target 1'	Cancel				×)	
tions for Target Device   Target	OK Target 1' Output   Listing User	Cancel	51 Locate   LX51 Mi	sc   Debug   Utilities		×	
tions for Target Device   Target   Run User Prog	OK Target 1' Output   Listing User ams Before Compilation o	Cancel	51 Locate   LX51 Mi	sc   Debug   Utilities			
tions for Target Device   Target   Run User Prog Run #1: Run #2:	OK 'Target 1' Output   Listing User ams Before Compilation o	Cancel	51 Locate   LX51 Mi	sc   Debug   Utilities 	DOS16		
Device   Target   Run User Prog Run #1: Run #2: Run User Prog	OK 'Target 1' Output   Listing User ams Before Compilation o	Cancel	51 Locate LX51 Mi	sc   Debug   Utilities	     T DOS16   T DOS16		
tions for Target Device   Target   Run User Prog Run #1: Run #2: Run User Prog Run User Prog Run #1:	OK Target 1' Output   Listing User ams Before Compilation o	Cancel	51 Locate   LX51 Mi	sc   Debug   Utilities 	rep ] ] □ DOS16 ] □ DOS16 ] □ DOS16		
tions for Target Device   Target   Run User Prog Run #1: Run #2: Run User Prog Run #1: Run #1: Run #2:	OK Target 1' Output   Listing User ams Before Compilation o	Cancel	51 Locate LX51 Mi	sc   Debug   Utilities	DOS16 DOS16 DOS16		
Device   Target   Pevice   Target   Run User Prog Run #1: Run #2: Run User Prog Run #1: Run #2: Run #2:	OK Target 1' Output   Listing User ams Before Compilation o ams Before Build/Rebuild ams After Build/Rebuild	Cancel	51 Locate   LX51 Mi	sc   Debug   Utilities   	Hep □ DOS16 □ DOS16 □ DOS16 □ DOS16 □ DOS16 □ DOS16		
tions for Target Device   Target   Run User Prog Run #1: Run #2: Run User Prog Run #1: Run #2: Run User Prog Run User Prog Run User Prog	OK Target 1' Output   Listing User ams Before Compilation o ams Before Build/Rebuild ams After Build/Rebuild Hex2Carray.exe #H App	Cancel	51 Locate LX51 Mi	sc   Debug   Utilities	DOS16 DOS16 DOS16 DOS16 DOS16		
tions for Target Device   Target   Run User Prog Run #1: Run #2: Run User Prog Run #1: Run User Prog Run #1: Run #2:	OK Target 1' Output   Listing User ams Before Compilation o ams Before Build/Rebuild ams After Build/Rebuild Hex2Carray.exe #H App	Cancel	51 Locate LX51 Mi	sc   Debug   Utilities	   DOS16   DOS16   DOS16   DOS16   DOS16   DOS16   DOS16		

The HEXfile format must be Intel386. The name of the applet is also specified here.

the Hex 2 C-array converter

Here, the Hex2Carray tool is called from the User –Run box after the HEXfile has been created. The command line is:

Hex2Carry <HEXFILENAME> <APPLETNAME>

The uVISION IDE supplies the HEXfile name from the Output tab via the "#H" token. The application name chosen by the user is supplied by the second command line parameter.

Hex2Carray.exe #H App2

The result is that the PK51 program will exist within a C source file called in this case, C51CodeApp2.H and have a header file C51CodeApp2.H. These provide the input to the Tasking CTC application that will load and run the applet in TC1782 board.

DEVELOPMENTTOOLS

### **PRO-SIL Family**

hitex

Options for Target 'Targe	et 1'	×		
Device   Target   Output	:   Listing   User   C51   AX51   LX51 Locate LX51 Misc   Debug   Uti sable Warning Numbers:	lities		
Greate reincetable	e: .VCICApplet.lin			
Assign				
Misc controls		*		
Linker control string		*	Figure 9 the linker	Specifying control file
	OK Cancel Defaults	Help		

## 6 Using Batchfiles For CIC61508 Applets in SDCC

Each of the three supplied SDCC applet examples may also be compiled and linked using simple MS-DOS batchfiles. In each applet subdirectory, make batchfiles can be found. These are of the form:

```
echo off
cls
echo *
echo * Assemble Startup Files
C:/DAVE-Bench-201\SDCC XC800\bin\as-xc800 -plosgffcx crtstart.s
C:/DAVE-Bench-201\SDCC XC800\bin\as-xc800 -plosgffcx crtxinit.s
C:/DAVE-Bench-201\SDCC XC800\bin\as-xc800 -plosgffcx crtxstack.s
C:/DAVE-Bench-201\SDCC XC800\bin\as-xc800 -plosqffcx startupxc866.s
echo *
echo * Compile Applet Files
C:/DAVE-Bench-201\SDCC XC800\bin\sdcc -c SecSPIDefs.c --main-return --debug
I"C:\c51dev\CICApps\App3 SDCC"
                                   -I"C:/DAVE-Bench-201\SDCC XC800\include"
I"C:/DAVE-Bench-201\SDCC XC800\include\xc800"
                                                                 -I"C:/DAVE-Bench-
201\SDCC XC800\include\asm\xc800" -mXC800 -pXC866 4FR --model-small --iram-size
0x100
C:/DAVE-Bench-201\SDCC XC800\bin\sdcc
                                       -c
                                            main.c --main-return
                                                                       --debug
I"C:\c51dev\CICApps\App3_SDCC"
                               -I"C:/DAVE-Bench-201\SDCC_XC800\include"
I"C:/DAVE-Bench-201\SDCC_XC800\include\xc800"
                                                                 -I"C:/DAVE-Bench-
201\SDCC XC800\include\asm\xc800" -mXC800 -pXC866 4FR --model-small --iram-size
0x100
C:/DAVE-Bench-201\SDCC XC800\bin\sdcc
                                        main.rel
                                                    SecSPIdefs.rel
                                                                     crtstart.rel
crtxinit.rel crtxstack.rel startupxc866.rel --no-xinit-opt --main-return --debug -
I"C:\c51dev\CICApps\App3 SDCC\Common\inc" -I"C:/DAVE-Bench-201\SDCC XC800\include"
-I"C:/DAVE-Bench-201\SDCC XC800\include\xc800"
                                                                 -I"C:/DAVE-Bench-
201\SDCC XC800\include\asm\xc800" -mXC800 -pXC866 4FR --model-small --iram-size
0x100 -WI -bBSEG=0x20 --xram-loc 0xF080 --xram-size 0x80 --code-loc 0xF100 --code-
size 0x100 --data-loc 0x00 --idata-loc 0x80 --stack-loc 0x80 -Wl -bPSEG=0xF000 -o
"App3 SDCC.ihx"
echo *
echo * Make a copy of the hex file
copy App1 SDCC.ihx App1 SDCC.hex
copy *.ihx debug
copy *.CDB debug
echo *
echo * Convert HEX to C const array
Hex2Carray.exe App1 SDCC.ihx App1 SDCC
echo *
echo * Completed
```

They show at the lowest level how the SDCC tools are used.



## 7 Applet Examples In Detail

### 7.1 Including CIC61508 Applets In Tricore Applications

In the example, the files containing the applet C const arrays are place in a directory "Applets", subdirectory "CIC61508\_Src":

	Name	Date modified	Туре	Size	
	🐌 Applets	18/03/2011 13:59	File folder		
	退 Dave	18/03/2011 13:59	File folder	-	
	退 Debug	18/03/2011 14:07	File folder		
	🕌 Debugger	18/03/2011 13:59	File folder		
	\mu Docs	18/03/2011 13:59	File folder		
	JARDISS	18/03/2011 13:59	File folder		
	cproject	18/03/2011 14:11	CPROJECT File	36 KB	
	.project	16/12/2010 17:38	PROJECT File	1 KB	
	CIC_ProdProg_TC1782.lsl	16/12/2010 10:00	LSL File	4 KB	
	CIC_ProdProg_TC1782.simulator.launch	16/12/2010 10:00	LAUNCH File	3 KB	
	cstart.c	16/12/2010 10:00	C Source	35 KB	
	cstart.h	16/12/2010 10:09	C/C++ Header	8 KB	
Within this	DConfig	16/12/2010 10:00	File	1 KB	directory are 4
further	MConfig	16/12/2010 10:00	File	1 KB	directories:
	Name	Date modified	Туре	Size	
	鷆 CIC61508_Include	18/03/2011 13:59	File folder		
	CIC61508_Src	18/03/2011 13:59	File folder		
	🐌 Host_Include	18/03/2011 13:59	File folder		
The	July Host_Src	18/03/2011 13:59	File folder	<u>"</u> Цо	et Sre" directory
contains a				mo	dule

"Host\_Applet\_Manager.c" that holds the Tricore code that loads the applet into the CIC65108 and runs it. For example, App1 is managed by:



The function uiLoad\_CIC61508Applet() is located in the CIC\_SecSPI\_Drv project and takes care of all the preparatory actions required to get an applet into the CIC61508 RAM, apart from entering Secure SPI Mode, which is done by the main() function.



### 7.2 Example Applets Tricore Program

The example applets are called from a single demonstration main() function so that they run sequentially.

```
/* Run Applet Examples */
  /* Toggle SYSDISA every 19.2 us */
#ifdef ENABLE APP1
  vExecute_App1(); /* Warning - this does not allow the CIC to return so a hard reset will
be needed */
#endif
  /* Continuously read all CIC61508 pins (except SPI) for 30secs */
  vExecute App2();
  /* Restart CIC as App2 destroys IO setup */
  uiReset CIC();
  /* Enter Secure SPI Mode */
  CIC status = uiCIC SPI EnterSecureMode() ;
  /* Continuously toggle the SYSpins for 30 secs */
  vExecute App3();
  /* Wait for a short time by using the reset timer in the SPI driver */
  CIC SPI state = Start CS Wait For CIC Reset ;
  /* Set state of SYSDIS Pins */
  /* Wait for a short time */
  CIC_SPI_state = Start_CS_Wait_For_CIC_Reset ;
  vSYSDIS State(0,0,1); /* SYSDISA = 0, SYSDISB = 0, SYSDISC = 1 */
  /* Wait for a short time */
  CIC_SPI_state = Start_CS_Wait_For_CIC_Reset ;
  vSYSDIS_State(0,1,0); /* SYSDISA = 0, SYSDISB = 1, SYSDISC = 0 */
  /* Wait for a short time */
  CIC_SPI_state = Start_CS_Wait_For_CIC_Reset ;
  vSYSDIS State(0,1,1); /* SYSDISA = 0, SYSDISB = 1, SYSDISC = 1 */
  /* Wait for a short time */
  CIC SPI state = Start CS Wait For CIC Reset ;
  /* Wait for a short time */
  CIC_SPI_state = Start_CS_Wait_For_CIC_Reset ;
  vSYSDIS State(1,0,1); /* SYSDISA = 1, SYSDISB = 0, SYSDISC = 1 */
  /* Wait for a short time */
  CIC_SPI_state = Start_CS_Wait_For_CIC_Reset ;
  vSYSDIS State(1,1,1); /* SYSDISA = 1, SYSDISB = 1, SYSDISC = 1 */
  /* End of test area */
```



Please note that Example 1 is a special case as it does not return. This is why it has to be specifically enabled by adding:

#define \_ENABLE\_APP1\_

At the top of the MAIN.C.

All the example applet drivers are located in "Host\_Applet\_Manager.C".



### 7.3 Applet Example 1

### 7.3.1 C51 Applet

This is the simplest example. It makes sure that the SYSDISA (Port3.0) pin is enabled as an output and then toggles it every 19.2us. The toggle rate is controlled by the CIC61508's unused Timer0. This applet does not return to the normal secure SPI mode.

```
/* Main program */
void main (void)
{
  /* Declare locals */
    /* Set the direction of Port3.0 and Port3.1 to output */
   P3 DIR = (P3 DIR | SPC P3 DIR MASK);
  CPU TMOD = 0x02U; /* 8-bit autoreload mode */
  CPU TL0 = 0x00U; /* 19.2us period */
  CPUTHO = 0x00U;
  TRO = 1U; /* Start timer */
  while(1U)
  {
     /* Toggle P3.3 */
     P3_DATA ^= SPC_P3_SYSDIS_A;
     /* Wait for 19.2us timeout */
     while(TF0 == OU)
     {
      ;
     }
      /* Clear timer0 overflow flag */
     TFO = OU;
  }
} /* Total code size = 31 bytes */
```



### 7.3.2 Tricore Driver

This is the simplest possible applet driver. It loads the applet contained in C51CodeApp1.C to 0xF100 in the CIC61508 and then executes a JumpToAddress command.

The loader function exists in the Secure SPI mode library. It consists of just:

```
CIC ErrorType uiLoad CIC61508Applet(uint8 *uiAppletCode, uint16 uiAppletSize,
                                    uint16 uiAppletCodeBaseAddr)
{
  /* Declare locals */
  uint16 uiI;
  CIC_ErrorType uiCIC_status;
  /* Load a CIC61508 applet */
  for(uiI = 0x00 ; uiI < uiAppletSize ; uiI++)</pre>
  {
     uiCIC status = uiCIC WriteSecureAddress(uiAppletCode[uiI],
                                              (uiAppletCodeBaseAddr + uiI), XDATA);
  }
  if(uiCIC_status == CIC_No_Error)
  {
     /* Jump to applet */
     uiCIC status = uiCIC JumpAddress(0x00U, uiAppletCodeBaseAddr , Jump Address);
  }
  return(uiCIC_status);
}
```

### 7.3.3 Running Example 1

This example is disabled as supplied. To enable it, add:

#define \_ENABLE\_APP1\_



At the top of the MAIN.C and rebuild the Tricore application. Start the PLS debugger and load the C:\CIC\_ProdProg\_TC1782\Debugger\UDE.WSX workspace. Load the Tricore application and run it. You should find that SYSDISA now toggles every 19.2us.

This kind of applet is useful where the applet cannot return. An example may be where the SPI pins are being checked on an automatic tester, under which circumstances no further CIC communications would be possible.



### 7.4 Applet Example 2

The standard secure SPI mode read and write functions do not allow the state of CIC61508 IO pins to be directly checked. This applet provides a simple way to read all the CIC IO port pins (except those used for SPI) via the normal secure READ function in the Secure Mode library.

The example relies on an XDATA array at X:0xF080 which is loaded by the applet with the values currently on the IO ports. It makes sure that all port pins (except SPI) are converted into input mode first. Thus this allows pins that are normally outputs to be read.

If you have just run Example 1, make sure you remove the

#define \_ENABLE\_APP1\_

previously added otherwise Example 2 will never be reached.

```
void IO_ReadTest(void)
{
    /* Declare Locals */
    /* Copy the values of the port pins to XDATA */
    uiPx_Image[0] = P0_DATA;
    uiPx_Image[1] = P1_DATA;
    uiPx_Image[2] = P2_DATA;
    uiPx_Image[3] = P3_DATA;
    /* XDATA pin images can now be read via secure READ command */
}
```

The applet exits immediately back to secure SPI mode.



### 7.5 Example 2 Tricore Driver

This driver uses a different approach to Example 1. Here the applet that reads the IO port pins into the XDATA uiPx\_Image[] runs once and then exits. The uiCIC\_ReadSecureAddress() library function is then used to read the port images into global variables "uiCIC61508\_Px" in the Tricore application. These can be displayed in the debugger real time update window.

However as the port-capture applet is now loaded into the CIC61508 at 0xF100, we can call it repeatedly to update the port pin status in the Tricore global variables. If the ADC pots on P2 are adjusted, the uiCIC61508\_P2 value in the debugger watch window will change.

```
void vExecute App2(void)
{
   /* Declare locals */
   uint32 uiTimer;
   /* Load a CIC61508 applet */
   CIC status = uiLoad CIC61508Applet(CIC61508 AppletCodeApp2,
                                             uiCIC61508 AppletSizeApp2,
                                             CIC61508 APPLET CODE BASE);
   if(CIC status != CIC No Error)
   {
      /* User can decide how to handle errors */
      while(1) { ; }
   }
   /* Run this test for about 30 secs */
   for(uiTimer = 0x00 ; uiTimer < 20000U ; uiTimer++)</pre>
   {
      /* Read IO Port images in XDATA */
      uiCIC61508_P0 = (uint8)uiCIC_ReadSecureAddress(0xF080, XDATA) ;
     uiCIC61508_P1 = (uint8)uiCIC_ReadSecureAddress(0xF081, XDATA) ;
uiCIC61508_P2 = (uint8)uiCIC_ReadSecureAddress(0xF082, XDATA) ;
uiCIC61508_P3 = (uint8)uiCIC_ReadSecureAddress(0xF083, XDATA) ;
      /* Jump to the applet again to re-read IO ports */
      CIC status = uiCIC JumpAddress(0x00U, CIC61508 APPLET CODE BASE ,
                                             Jump Address);
   }
}
```

Note: as this example destroys the



# 7.6 Applet Example 3

The standard secure SPI mode read and write functions do not allow the state of CIC61508 SYSDIS pins to be directly controlled. During early board testing, These pins can be very important as they often have a big influence on other hardware. This example along gives a way to control them.

The example is supplied in 2 forms. The first one is a simple demonstration of toggling each SYSDIS pin in sequence. The second one uses the same applet to allow individual SYSDIS pin control via an interface function.

The principle of operation is similar to Example 2 except that the XDATA array of port images is now written to by the secure mode write command and the array itself then written to the actual port pins by the applet. The writes to the port pins are triggered by the repeated calling of the applet.

```
void IO_SYSDISx_WriteTest(void)
{
    /* Declare Locals */
    uint8 Data;
    /* Update SYSDIS A & B on Port3 */
    Data = (P3_DATA & SPC_P3_DATA_MASK);
    Data |= uiPx_Image[3];
    P3_DATA = Data & (uint8)~SPC_P3_DATA_MASK;
    /* Update SYSDIS C on Port0 */
    Data = (P0_DATA & SPC_P0_DATA_MASK);
    Data |= uiPx_Image[0];
    P0_DATA = Data & (uint8)~SPC_P0_DATA_MASK;
    /* Opdata = Data & (uint8) & (uint8)
```

}



### 7.6.1 Example 3, Part2 Tricore Driver

This is an example how applets are used in a real situation. A simple interface function "vSYSDIS\_State()", allows the user to specify the value of each SYSDIS pin individually:

```
void vSYSDIS_State(boolean SYSDIS_A_state, boolean SYSDIS_B_state,
                  boolean SYSDIS C state)
{
  /* Declare locals */
  uint8 SYSDISAB State ;
  /* Load a CIC61508 applet */
  CIC_status = uiLoad_CIC61508Applet(CIC61508_AppletCodeApp3, uiCIC61508 AppletSizeApp3,
CIC61508 APPLET CODE BASE);
  if(CIC_status != CIC No Error)
  {
     /* User can decide how to handle errors */
    while(1) { ; }
  }
  /* Combine SYSDISA and SYSDISB states into a single variable */
  SYSDISAB_State = (uint8)SYSDIS_A_state | ((uint8)SYSDIS_B_state * 2);
  /* Update the SYSDISx pins from the Port images in XDATA */
  CIC status = uiCIC WriteSecureAddress(SYSDISAB State, Port3 Image, XDATA);
  CIC status = uiCIC WriteSecureAddress(((uint8)SYSDIS C state * 0x04U), Port0 Image,
XDATA);
  /* Jump to the applet again to write the SYSDISx pins */
  CIC status = uiCIC JumpAddress(0x00U, CIC61508 APPLET CODE BASE , Jump Address);
}
```

It is called from the main() demonstration as per:

On each invocation, the applet App3 loaded by vExecute\_App3() is re-called to write the new SYSDIS pin pattern to the CIC61508 pins.



### 8 CIC61508 Secure SPI Mode Driver Library

This library has been design to run on any Tricore variant and requires just a pre-initialised SSC port and GPTA compare register interrupt. The functions contained with it are listed below:

extern void vCIC SPI Driver(void) ; extern void vInit CIC SPI Driver (void) ; extern uint16 uiCIC SPI SendMessage(uint16 uiTx data) ; extern CIC\_ErrorType uiCIC\_SPI\_ReadSFR(uint8 uiSFR\_addr, uint8 \*uiSFR\_data) ;
extern CIC\_ErrorType uiCIC\_SPI\_WriteSFR(uint8 uSFR\_addr, uint8 uiSFR\_data) ; extern CIC ErrorType uiCIC SPI EnterSecureMode(void) ; extern uint16 uiCIC ReadSecureAddress(uint16 uiAddress, uint16 uiMspace) ; extern CIC\_ErrorType uiCIC\_EraseDFLASH(void) ; extern CIC\_ErrorType uiCIC\_WriteEntireDFLASH(uint8 \*uiInputData, uint16 uiLength) ; extern uint16 uiCIC PollDFLASH Busy(void) ; extern CIC\_ErrorType vVerify\_DFLASH(uint8 \*uiVerifyArray, uint16 uiDFLASH\_length, uint16 uiDFLASH base addr) ; extern CIC ErrorType uiReset CIC legacy(void) ; extern CIC\_ErrorType vGet\_CIC61508\_SFRs(void) ; extern void vCIC Format DFLASH(uint8 \*uiDFLASH ptr, uint16 uiDFLASH length) ; extern void CIC wait (void) ; extern void vDeInit CIC SPI Driver(void) ; extern void vRead DFLASH Tune ID(int8 \*iBuffer) ; extern uint16 uiCIC\_SPI\_FastWriteSFR(uint8 uiSFR\_addr, uint8 uiSFR\_data) ; extern CIC\_ErrorType uiCIC\_SPI\_FastReadSFR(uint8 uiSFR\_addr, uint8 \*uiSFR\_data) ; extern CIC ErrorType uiReset CIC(void); extern CIC ErrorType uiCIC WriteSecureAddress (uint8 uiData, uint16 uiAddress, uint16 uiMspace); extern CIC\_ErrorType uiReset\_CIC\_Immediate(void); extern CIC\_ErrorType uiLoad\_CIC61508Applet(uint8 \*uiAppletCode, uint16 uiAppletSize, uint16 uiAppletCodeBaseAddr); extern CIC ErrorType uiCIC JumpAddress (uint8 uiData, uint16 uiAddress, uint16 uiMspace);

The function prototypes for the functions in the library are accessed via the CIC\_SecureMode.h header file.

A full version of the SPI driver library is available under a specific licence for use in customers' own applications.

www.hitex.com