# ANALOG DEVICES

# Introduction

The **EVAL-ADICUP360** is an Arduino-like platform based on the **ADUCM360** fully integrated, 3.9 kSPS, 24-bit data acquisition system that incorporates dual high performance, multichannel sigma-delta ( $\Sigma$ - $\Delta$ ) analog-to-digital converters (ADCs), a 32-bit ARM Cortex <sup>M</sup>-M3 processor, and Flash/EE memory on a single chip. The platform has an Arduino-Due form factor and has two additional PMOD connectors. An Eclipse based development environment is provided for code development and debugging. The base platform is accompanied by a set of shields provided by Analog Devices but it can also work with 3rd party Aduino shields.

This guide is structured as follows:

- Hardware Contains hardware-related information about the base board and the various shields
- Quick Start Guides Provides all the necessary steps to get the software environment up and running
- **Reference Designs** Contains detailed descriptions of the software reference designs available for the base board and the shields
- **Help and Support** Provides info on where to get support on any questions you might have regarding the hardware or the software

# **Tool Chain Guides**

This chapter provides all the necessary steps to get the software environment up and running.

It contains two main sections:

- **Tool Chain Download & Installation Guide** Provides all the necessary instructions on how to download and install the customized software development environment for ADuCM360 based on Eclipse IDE.
- Tool Chain Setup & User Guide Provides information about using the customized Eclipse IDE, in particular, the process of importing, building, debugging, and creating user applications for the ADuCM360 microcontroller.
- Using EVAL-ADICUP360 with IAR and Keil IDEs Provides detailed information how to use the EVAL-ADICUP360 board with other IDEs than Eclipse, such as IAR Embedded Workbench and Keil  $\mu$ Vision.

# **Tool Chain for EVAL-ADICUP360**

This page provides all the necessary steps to get the customized Eclipse software environment up and running in either Windows or Linux.

The software development environment for EVAL-ADICUP360 is based on open source tools, and includes the following features and components:

- ADuCM360 customized Eclipse IDE for C/C+ + Developers
- GNU Tools for ARM Embedded Processors (GCC toolchain for ARM processors)
- GNU ARM Eclipse Plug-ins, Copyright © 2009 Liviu Ionescu, A family of Eclipse CDT extensions and tools for GNU ARM development (open source ARM debug and build tools)
- GNU ARM Eclipse Build Tools, Copyright @ 2009 Liviu Ionescu (GNU make & busybox: sh, rm and echo)
- OpenOCD with support for ADuCM360 microcontroller (open source SWD)
- Mbed CMSIS-DAP/Serial drivers (for Windows)

The **Windows IDE** compared to **Linux IDE** is pretty much the same. So if the documentation mainly references the **Windows** version, things directly apply to the **Linux** version as well.

The **EVAL-ADICUP360 Toolchain** is based on **Eclipse IDE**, but because the MBED platform provides CMSIS-DAP interface to connect to the board, the EVAL-ADICUP360 can be used without problems together with **IAR Embedded Workbench IDE** or **Keil µVision IDE**.

# **Pre-Requisites and Requirements List**

There are a few things that you will need for the tools and software to work properly.

- PC or laptop computer
- (2) Micro USB to USB cables

Both USB cables needs to have ALL data lines connected, can't use a charging only micro USB cable.

• Terminal Program to interface your PC with the EVAL-ADICUP360

- Putty
- Tera Term
- Or other favorite Terminal program
- Detailed ADuCM360 User Guide

# **Windows Tool Chain Installer Instructions**

**Windows Tool Chain Installer** is a tool that facilitates the installation of the entire tool chain. It's an single executable file which will either automatically download or install bundled packs of all of the necessary tool chain components. The following open source components are included in the tool chain installer:

- Customize ADuCM360 Eclipse IDE for C/C+ + Developers
- OpenOCD with ADuCM360 support
- GNU Tools for ARM Embedded Processors
- GNU Eclipse Build Tools for ARM Processors
- Mbed Windows serial port driver

DURING INSTALLATION ONLY CONNECT HARDWARE DEVICE WHEN YOUR ARE PROMPTED BY THE DRIVER INSTALLER. WAIT TO ENSURE THAT THE HARDWARE DRIVERS ARE COMPLETELY (AND SUCCESSFULLY) INSTALLED BEFORE MOVING FORWARD WITH THE SOFTWARE INSTALL



The executable installs the components to a default local directory structure which can be found below.

- ADuCM360 Eclipse IDE installs to C:\Analog Devices\ADuCM360-IDE\Eclipse (also creates links in the Start menu)
  - The customized ADuCM360 Eclipse IDE includes the required Eclipse plug-ins for ARM processor and for the debug tools.
- OpenOCD is saved in C:\Analog Devices\ADuCM360-IDE\openocd\usr\bin
- GNU ARM Processor Tools will be saved to C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2014q4\arm-none-eabi
- GNU ARM Eclipse Build Tools will be save to C:\Program Files (x86)\GNU ARM Eclipse\Build\_Tools

# **Linux Tool Chain Installer Instructions**

There are two methods to get Linux Tool Chain Installer:

- Debian packages for 32-bits and 64-bits
- Tarball packages for 32-bits and 64-bits

The preferred way is to use the **debian packages** (.deb). You may be asked if you want to install them while there are being downloaded. If not, you can use your file browser. The **debian package** is typically recognized and associated with your distribution package management system. In any case you can also install them from the command line using following command:

# sudo dpkg -i DEB\_PACKAGE

The alternative way is to download the self-contained and relocateable **tarball packages** (.tar.gz). The only advantage is that unlike the **debian package** is installed system wide. You can extract the tarball anywhere including your *HOME* directory:

# tar xzf TAR\_PACKAGE

The **tarball package** also includes some simple shell scripts that installs a symplic link to the **aducm360-ide** executable into */usr/local/bin*, copies the *udev* rules for the OpenOCD debugger, installs a application launcher with icon etc.

Depending on your system (32-bits or 64 bits) you should pick one of the available versions:



EVAL-ADICUP360 Tool Chain Installer for Linux 32-bit (debian package) EVAL-ADICUP360 Tool Chain Installer for Linux 64-bit (debian package) EVAL-ADICUP360 Tool Chain Installer for Linux 32-bit (tarball package) EVAL-ADICUP360 Tool Chain Installer for Linux 64-bit (tarball package)

## **Tool Chain Setup User Guide**

This page provides detailed information about using the ADuCM360 customized Eclipse IDE, in particular, the process of importing, building, debugging, and creating user applications for the ADuCM360 microcontroller.

This page will outline:

- 1. How to import existing projects into your workspace
- 2. How to build the .elf files, for programming the ADuCM360
- 3. How to configure the debug session for a particular user application
- 4. How to start a debug session
- 5. How to create a new project

### **Workspace and Projects**

The workspace is a folder where Eclipse can access local copies of user application projects. When starting Eclipse, a prompt will ask you for a location of this folder. This is the location where all the ADuCM360 user applications will be stored.

## **Using the Tool Chain**

The instructions below have been tested in Windows XP and Windows 7, on both 32-/64-bit machines.

### **Importing a Project**

There are 2 methods for importing existing programs:

- Examples that come with the installer package.
- Examples which are in our GIT repository (most up to date content).

Only one method is needed to get started with the EVAL-ADICUP360.

### How to Import Existing Projects within the Installer Package

- 1. From the menu located in the tool bar, select the *File*  $\rightarrow$  *Import* option.
- 2. A window will pop-up with several importing options, select General  $\rightarrow$  Existing Projects into

	Import	
	Select Create new projects from an archive file or directory.	Ľ
	Select an import source:	
	type filter text	
Workspace.	(?) < Back Next > Ein	sish Cancel

- 3. Select *Browse* in the dialog window and search for the local copy of where the ADuCM360-IDE examples are. If you used the default directory that can be found here: C:\Analog Devices\ADuCM360-IDE\Examples
- 4. Make sure that the check-box *Copy projects into workspace* is checked (this creates a local copy of the projects and preserves the original versions) and press *Finish*.

Import					- 0 ×
Import Projects Select a directory to sea	rch for existing Eclipse projects.				
Select root directory:	C:\Analog Devices\ADuCM360-IE	DE\ADuCM360 Examp	les	•	Browse
Select archive file:				*	Browse
Projects:					
ADuCM360_demo	_blink (C:\Analog Devices\ADuCM	1360-FINAL\ADuCM36	0 Examples\ADuCM	360_demo_blink)	Select All
ADuCM360_demo	o_cli (C:\Analog Devices\ADuCM36	60-FINAL\ADuCM360 I	Examples\ADuCM36	0_demo_cli)	Deselect All
					Refresh
Options Search for nested pro Copy projects into w Hide projects that all	ojects orkspace ready exist in the workspace				
Working sets	ing cate				
Working sets:	ing sets			•	Select
3		< Back	Next >	Finish	Cancel

If you imported the example programs from the installer, you can skip ahead to "Building the .ELF/.HEX Files" section. The only time you will need to import from the GIT repository, is if you want to look for newly released/updated programs.

### How to Import Existing Projects from the GIT Repository

1. Open the GIT perspective window by navigating the menu near the tool bar. File  $\rightarrow$  Perspective  $\rightarrow$  Open Perspective  $\rightarrow$  Other  $\rightarrow$  GIT and the press "OK".

New Window	C	🖢 🔗 🕶 🔳 🕥 🖢 🗧 🕶	\$	▼ □) ▼
Editor Hide Toolbar	•			
Show View				
Perspective	•	Open Perspective	•	CodeRed
Navigation	•	Customize Perspective	<b>茶</b> 11	Debug Team Synchronizing
Preferences		Save Perspective As Reset Perspective		Other
		Close Perspective	-	outeni
		Close All Perspectives		
oen Perspective C/C++ (default) CodeRed Debug Git				
een Perspective C/C++ (default) CodeRed Debug Git Packs Remote System Expl Resource Team Synchronizing	orer			

- 2. Clone the Git repository which contains all the latest code and projects associated with the ADuCM360. Populate the URI field with the following address.
  - 1. URI: https://github.com/analogdevicesinc/EVAL-ADICUP360.git
  - 2. Click Next  $\rightarrow$  Next  $\rightarrow$  Finish

	n of the source repository.
ocation	
JR <u>I</u> :	I//github.com/analogdevicesinc/EVAL-ADICUP360.git Local File.
lost:	github.com
epository path	n: /analogdevicesinc/EVAL-ADICUP360.git
Authentication	
assword:	
- - - - - - - - - - - - - -	ure Store
_ <u>store</u> in seco	
<u>store in sect</u>	

3. In the Git Repositories window, *Right Click* on *Projects* folder and select the *Import* option.

Branches			
Tags			
Reference	25		
<ul> <li>Monting</li> </ul>	Diver	tan CAllerabilitabilitabilitati	DICHD260
Working	Direc	tory - us/users/jkubik/git/EVAL-/	ADICOP300
p 📂 .git	te		
b CA	+	Add to Index	
DOA	×	Delete	Delete
	2	Import Projects	
		Show In	Alt+Shift+W +
		Copy Path to Clipboard	Ctrl+C
		Paste Repository Path or URI	Ctrl+V
		Paste Repository Path or URI	Ctrl+V

4. Select the radio button for *Import existing Eclipse projects* and click on the *projects* folder as the destination.

▶ Import Projects from Git Repository C:\Users\jkubik\git\EVAL-ADICUP36 👝 💷 🕰
Select a wizard to use for importing projects
Depending on the wizard, you may select a directory to determine the wizard's scope
Wizard for project import
Import existing Eclipse projects
Import using the New Project wizard
Import as general project
<ul> <li>jeit</li> <li< td=""></li<></ul>
< Back Next > Finish Cancel

Import Projects Import projects from a Git repository Designets	
type filter text to filter unselected projects           Image: Comparison of the selected projects	Select All
III     Working sets	
Working sets:	Select
< Back Next > Finish	Cancel

5. Click Next  $\rightarrow$  Finish

## **Building the .ELF/.HEX Files**

It's important to build your project before setting up the debug configuration. This will allow the debug configuration to automatically populate the appropriate fields.

- 1. Starting on the C/C+ + perspective, select the project you want to debug in the *Projects Explorer* Window.
- 2. Right click on the project and select the *Build Project* option.
  - $^{\circ}$  Could also go up to the tool bar and click on the Hammer icon  $^{\circ}$ .

## Setting up a Debug Configuration for the Project

A new debug configuration must be set up for EACH application you intend on developing/debugging. So you will have many different debug configurations, depending on the number of programs you create/debug.

- 1. Go to the menu bar and follow this path,  $Run \rightarrow Debug \ Configurations...$ 
  - <sup>o</sup> Alternatively, locate the small bug icon in the tool bar and click the small downward facing arrow to the right, and select the *Debug Configurations…* option from the menu.
- 2. Double click the *GDB OpenOCD Debugging* configuration from window.

Debug Configurations		×
Create, manage, and run cont	igurations	\$
type filter text C C/C++ Application C C/C++ Attach to Applic C C/C++ Remote Applicar C GDB Hardware Debuggi G GDB OpenOCD Debuggin G GDB SEGGER J-Link Deb Launch Group	Configure launch settings from this dialog: Press the 'New' button to create a configuration of the selected type. Press the 'Duplicate' button to copy the selected configuration. Press the 'Delete' button to remove the selected configuration. Configure setter' button to configure filtering options. Configure launch perspective settings from the 'Perspectives' preference page.	
Filter matched 9 of 9 items		
?	Debug	Close

3. The necessary input fields should be populated, assuming that you built your project in the previous step. The following images should be used as a reference if some of the fields are blank.

Debug Configurations		8
Create, manage, and run configurations	5	Ť.
C 🕼 🗶 🔁 🕸 •	Name: ADuCM360_demo_blink Debug	
type filter text	🕞 Main 🕸 Debugger 🗭 Startup 🍕 Source	e 🗔 Common
C/C++ Application	Project	
C/C++ Attach to Application	ADuCM360_demo_blink	Browse
C/C++ Remote Application	C/C++ Application:	
GDB Paraware Debugging     GDB OpenOCD Debugging	Debug\ADuCM360_demo_blink.elf	
<ul> <li>C ADuCMS0_demo_blink Debug</li> <li>G GDB QEMU Debugging</li> <li>G GDB SEGGER J-Link Debugging</li> <li>► Launch Group</li> </ul>	Build (if required) before launching Build configuration: Debug Enable auto build Use workspace settings	Variables Search Project Browse Disable auto build <u>Configure Workspace Settings</u>
Filter matched 10 of 12 items		Revert Apply
3		Debug Close

4. Next, switch to the Debugger tab and ensure the following required GDB commands are present.

🗅 🕼 🗶 📄 🏇 •	Name: ADuCM36	)_demo_blink Debug			
type filter text	Main 🕸 Deb	ugger 🕨 Startup 🤤 Source 🛄 Common			
C C/C++ Application C/C++ Attach to Application C/C++ Postmortem Debugger C C/C++ Remote Application G GDB Hardware Debugging G GDB OpenOCD Debugging C ADuCM360_demo_blink Debug G GDB QEMU Debugging G GDB SEGGER J-Link Debugging Launch Group	OpenOCD Setup Start OpenOC Executable: GDB port: Telnet port: Config options: Allocate con GDB Client Setup Executable:	CD locally \${openocd_path}/\${openocd_executable} 3333 4444 -f board/eval-adicup360.cfg sole for OpenOCD \${cross_prefix}gdb\${cross_suffix}	Allocate console for the telnet connection	Browse	Variables
	Other options: Commands: Remote Target	set mem inaccessible-by-default off		Revert	* •

5. Finally, click the checkbox in the Common tab --> *Display in Favorites Menu* to make the created debug configuration appearing in the Debug Configurations section of the menu: Click "Apply", then "Close".

, and run configurations		- the second sec			
		)¢			
Name: A	DuCM360_demo_blink Debug				
Main	🕸 Debugger 🕨 Startup 🙀 Source 🕅	Common			
pplication Save as ittach to Application © Loca	l file				
emote Application O Share	ed file: \ADuCM360_demo_blink	Browse			
dware Debugging mOCD Debugging Display	in favorites menu	Encoding			
M360_demo_blink Debug /U Debugging GER J-Link Debugging Group	Debug Run	Other BO-8859-1			
Standar V Alloc	d Input and Output ate console (necessary for input)				
E Inpu	t File:				
Cout	ut File:	Workspace File System Variables			
Дарр	end	Workspace File System Variables			
2 Laund	V Launch in background				
of 12 items		Revert Apply			
Outp Appr V Launc	end h in background	Workspace File System V			

## **Debugging an Application**

- 1. Make sure the EVAL-ADICUP360 board is connected to the computer via the **DEBUG USB** port. (The micro USB connector closest to the DC barrel jack)
- Using the tool bar, navigate to the small Debug icon in and click on the downward arrow to the right. Select the debug configuration you want to download to the ADuCM360.



3. If this is the first time you have launched OpenOCD, a pop-up window will display asking for access.



Click on "Allow Access".

4. If everything goes fine, in the Console window, you will see a report without errors.



 As a reference, the full text should be similar to:Open On-Chip Debugger 0.10.0-dev-00025-g81cc011-dirty (2015-08-17-13:23)
 Licensed under GNU GPL v2

For bug reports, read http://openocd.org/doc/doxygen/bugs.html Info : only one transport option; autoselect 'swd' adapter speed: 5000 kHz adapter nsrst delay: 100 cortex m reset config vectreset init aducm Started by GNU ARM Eclipse Info : CMSIS-DAP: SWD Supported Info : CMSIS-DAP: Interface Initialised (SWD) Info : CMSIS-DAP: FW Version = 1.0 Info : SWCLK/TCK = 0 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1 Info : CMSIS-DAP: Interface ready Info : clock speed 5000 kHz Info : SWD IDCODE 0x2ba01477 Info : aducm360.cpu: hardware has 6 breakpoints, 4 watchpoints Info : accepting 'gdb' connection on tcp/3333 • The relevant EEPROM sectors of ADuCM360 are erased and the microcontroller is programmed. You can follow the progress in the Console window. As an approximate reference you will see something similar to:target state: halted target halted due to debug-request, current mode: Thread xPSR: 0x81000000 pc: 0x00000a94 msp: 0x20001fe0 semihosting is enabled **RESET: ADI halt after bootkernel** breakpoint set at 0x000001f5 Warn : Only resetting the Cortex-M core, use a reset-init event handler to reset any peripherals or configure hardware srst support. target state: halted target halted due to debug-request, current mode: Thread xPSR: 0x01000000 pc: 0x000001f4 msp: 0x20002000, semihosting flash 'aducm360' found at 0x0000000 Info : Padding image section 0 with 3 bytes RESET: ADI halt after bootkernel breakpoint set at 0x000001f5 Warn : Only resetting the Cortex-M core, use a reset-init event handler to reset any peripherals or configure hardware srst support. target state: halted target halted due to debug-request, current mode: Thread xPSR: 0x01000000 pc: 0x000001f4 msp: 0x20002000, semihosting **RESET: ADI halt after bootkernel** breakpoint set at 0x000001f5 Warn : Only resetting the Cortex-M core, use a reset-init event handler to reset any peripherals or configure hardware srst support. target state: halted target halted due to debug-request, current mode: Thread xPSR: 0x01000000 pc: 0x000001f4 msp: 0x20002000, semihosting ===== arm v7m registers (0) r0 (/32): 0x40002800 (1) r1 (/32): 0x00000001

(2) r2 (/32): 0x0000064 (3) r3 (/32): 0x0000000 (4) r4 (/32): 0x0000000 (5) r5 (/32): 0x0000000 (6) r6 (/32): 0x0000000 (7) r7 (/32): 0x0000000 (8) r8 (/32): 0x0000000 (9) r9 (/32): 0x0000000 (10) r10 (/32): 0x00000000 (11) r11 (/32): 0x0000000 (12) r12 (/32): 0x0000000 (13) sp (/32): 0x20002000 (14) lr (/32): 0xFFFFFFF (15) pc (/32): 0x000001F4 (16) xPSR (/32): 0x01000000 (17) msp (/32): 0x20002000 (18) psp (/32): 0x6A850410 (19) primask (/1): 0x00 (20) basepri (/8): 0x00 (21) faultmask (/1): 0x00 (22) control (/2): 0x00 ==== Cortex-M DWT registers (23) dwt ctrl (/32) (24) dwt cyccnt (/32) (25) dwt 0 comp (/32) (26) dwt\_0\_mask (/4) (27) dwt 0 function (/32) (28) dwt\_1\_comp (/32) (29) dwt\_1\_mask (/4) (30) dwt 1 function (/32) (31) dwt 2 comp (/32) (32) dwt 2 mask (/4) (33) dwt 2 function (/32) (34) dwt\_3\_comp (/32) (35) dwt 3 mask (/4) (36) dwt 3 function (/32)

5. The user application execution is then stopped automatically at the first breakpoint at the beginning of main() loop. From this point on, you can use the debug functions and features of the Eclipse environment. (Such as stepping through, breakpoints, register reads, variable values, etc.)



6. When finished, the debugger has to be stopped. Click on red Stop button up in the tool bar, then right click on the debug application in the "Debug" window, and select the *Terminate and Remove* option.



## **Creating a New Project**

The customized Eclipse IDE that you installed for EVAL-ADICUP360 offer the possibility to create 2 types of projects: **Empty Project** and **Hello World Project**. Both **C** and **C++** formats.

The idea of these templates is to have at the end a functional ADuCM360 project which can be run on the target. The basic system configuration is the same for both:

- startup code

- memory map and linker script

- system clock configuration

- disabling watchdog
- enabling clocks for all peripherals
- low drivers libraries for ADuCM360 microcontroller

The differences are regarding the complexity of the main() function: the **Empty** template provide an empty main() function, being in this way a good choose when you want to start your ADuCM360 project from the scratch; the **Hello World** template is for more complex projects. It provide 1 sec time base and different possibilities to display an output message to the user.

See below how to create the C projects for EVAL-ADICUP360 board. The same steps being available for C++ projects as well.

1. To create a new project, go to the menu bar and find  $File \rightarrow New \rightarrow C$  Project.

	New Alt, Children		Mahafia Dastad with Estation Code	0. • (% / • / (ii) (ii)			
	Open File	· D-	Makenie Project with Existing Code				
	Opentiness	100	C Project	B B Debug Pack			
	Close Ctrl+V		Project				
	Close All Ctrl+Shift+V			*			
	Save Ctrl+	C+	Convert to a C/C++ Project (Adds C/C++ Nature)				
	Save As	63	Source Folder				
2	Save All Ctrl+Shift+		Folder				
	Revert		Source File				
	Move		File from Template				
9	Rename F	8	Class				
0	Refresh						
	Convert Line Delimiters To	, 📮	Other	Ctrl+N			
	Print Ctrl+	-	the managed success times #/				
	C. Ach Wedenson		0, 63);	// Set timeout period for 0.5 seconds			
	Switch Workspace		M0, TCON CLK LFOSC, TCON PRE DIV256, TCON	MOD_PERIODIC   TCON_ENABLE);			
	Import		e(raneno_anen); // chaoae ranero a	ine .			
1			the system's tick interrupt */ ):	=			
10	Export						
	Properties Alt+Ente		oop */				
	1 main.c [ADuCM360_demo_blink/src]						
	2 .project [Program Files (x86)/]		{				
	3 msUartDrv.c [m360_demo_cli/src] 4 main.c [m360_demo_cli/src] Exit		ep(BLINK_TIME * TIMER_FREQUENCY_HZ);				
			cess();				
-	Dunblance ( Tracker	2.000	rata 🐑 🔲 Branastina) 🎋 Dahura) 🛛 🥼				
	CDT Build Console (ADuC	CDT Ruild Console (ADuCM360, demo, blink)					
	13:19:19 **** Incre	13:19:19 **** Incremental Build of configuration Debug for project ADuCM360_d					
	make all	make all					
	arm-none-eabi-size	arm-none-eabi-sizeformat=berkeley "ADuCM360_demo_blink.elf"					
	text data	text data bss dec hex filename					
	Finished building:	Finished building: ADuCM360_demo_blink.siz					
	13:19:20 Build Fini	hed (	took 756ms)	-			
	<ul> <li>The distance of the second s second second se Second second s</li></ul>	-	A SAN A SAN AND A SAN				

2. Provide a name for your project, and then choose Project Type: Executable  $\rightarrow$  ADuCM36x C/C+ + Project , with the Toolchains: Cross ARM GCC. Press Next.

C Project		
C Project Create C project of selected type		
Project name: my_project		
Use default location		
Location: C:\Users\vlupei\aducm360_final\my_pi	roject	Browse
Project type:	Toolchains:	
<ul> <li>Executable</li> <li>Empty Project</li> <li>Hello World ARM C Project</li> <li>ADuCM36x C/C++ Project</li> <li>Hello World ARM Cortex-M C/C++ Project</li> <li>Shared Library</li> <li>Static Library</li> <li>Makefile project</li> </ul>	Cross ARM GCC	
Show project types and toolchains only if they           Image: Constraint of the state of the s	are supported on the platform	Cancel

3. Choose as Processor core: *ADuCM360*.

Project settings		
Select the ADuCM36x p	rocessor and define project options.	
Processor core:	ADuCM360	•]
Content:	Empty (add your own content)	•
Use system calls:	Freestanding (no POSIX system calls)	•
Trace output:	Semihosting DEBUG channel	•
Check some warnings		
Check most warnings		
Enable -Werror		
Use -Og on debug	V	
Use newlib nano		
Use link optimizations		
?	< Back Next > Finish	Cancel

4. Select which type of project do you want: Content: Empty (add your own content) or Hello World (with 1s timer).

Select the ADuCM36x p	processor and define project options.	
Processor core:	ADuCM360	-
Content:	Empty (add your own content)	-
Use system calls:	Empty (add your own content) Hello World (with 1s timer)	
Trace output:	Semihosting DEBUG channel	-
Check some warnings		
Check most warnings		
Enable -Werror		
Use -Og on debug		
Use newlib nano		
Use link optimizations		

5. The Use system calls and Trace output options are available for **Hello World** template only.

_					
Processor core:	ADuCM360				
Content:	Empty (add your own content)				
Use system calls:	Freestanding (no POSIX system calls)				
Trace output:	Semihosting DEBUG channel				
Check some warnings					
Check most warnings					
Enable -Werror					
Use -Og on debug					
Use newlib nano					
Use link optimizations					

6. You can select various settings for your project (which can be changed later, in the project settings, or as different pre-processor definitions).

Processor core:	ADuCM360	•		
Content:	Empty (add your own content)	•		
Use system calls:	Freestanding (no POSIX system calls)	•		
Trace output:	Semihosting DEBUG channel			
Check some warnings				
Check most warnings				
Enable -Werror				
Use -Og on debug	<b>V</b>			
Use newlib nano				
Use link optimizations				

7. Select both the configuration check boxes you want to deploy on.

C Project	
Select Configurations	-
Select platforms and configurations you wish to depl	loy on
Project type: Executable	
Toolchains: Cross ARM GCC	
Configurations:	
V Debug	Select all
Kelease	Deselect all
	Advanced settings
Use "Advanced settings" button to edit project's prop	erties.
Additional configurations can be added after project	creation
Use "Manage configurations" buttons either on toolb	ar or on property pages.
() C Pack Nauto	Finish Cancel
C Dack Next :	

8. On the next page select the compiler toolchain. It should will automatically selected, just check or enter the right path to it.

C Project					
Cross GNU ARM Toolchain Select the toolchain and configure path					
Toolchain name:	GNU Tools for ARM En	nbedded Processors	(arm-none-eabi-gcc)	•	
Foolchain path:	C:/Program Files (x86)/	GNU Tools ARM Em	bedded/4.9 2015q2/bin	Browse	

9. Finally, press *Finish* and the project will be created and you can begin programming.

### **Options available for "Hello World" template only**

Use system calls available options are (see GNU ARM Eclipse support page):

- Freestanding a typical embedded configuration, that does not use the POSIX system calls (open, close, read, write, etc).
- POSIX (retargetting) a more elaborate embedded configuration where the application makes use of these calls, but redirects the file descriptors to local devices or files, by providing custom implementations for the system calls (like \_open, \_close, \_read, \_write etc). This configuration allows to port POSIX programs easier.

Semihosting - a special testing configuration, that bridges all system calls to the host operating system where the GDB server runs. This configurations is particularly interesting for running test programs that can leave the test results in a file stored on the host, for automated integration in a test suite.

#### Trace output available options are:

- None (no trace output) a basic configuration that doesn't use trace output messages.
- ARM ITM (via SWO) a specific configuration that help to print information via SWO pin when using J-Link.
- Semihosting STDOUT stream a more complex configuration that configure stdout to use a physical serial connection as UART or any other peripherals that offer the possibility to output messages.
- Semihosting DEBUG channel a debug specific configuration which enable semihosting in DEBUG mode and offer the possibility to use resources from the development platform n the embedded target via debugger. This can help the user to send trace stream to debugger console (like trace\_printf, trace\_puts etc).

## Assign Device to the Project using Packs

This step will allow you to access the ADuCM360 registers in debugger mode. In order to see the device list it is required to have Packs plug-ins installed. This is already done by the installer, however you do need to update the Packs list, and then install a particular family of devices.

- 1. To update your Packs library, go to the menu and Choose **Window** $\rightarrow$  *Perspective*  $\rightarrow$  *Open Perspective*  $\rightarrow$  *Other*
- 2. Click on "Packs". Once open, find the Packs window, and click on the Update arrow in the upper-righthand corner.
- 3. After updating has completed, find the folder for **Analog Devices** and navigate down to the latest version of **ADuCM36x** and right click and hit *Install*.

To assign device to your project:

- Select your project in the **Project explorer** view
- Go to Project tab from Eclipse menu and select Properties

C/C++ - my_project/src/main.c - Analog [	Devices Inc. ADuCM360 IDE			
File Edit Source Refactor Navigate Se	arch Project Run Window Help			
S ▼ S S V S V S V S V S V S V S V S V S	Open Project     Close Project     Close Project     Build All     Ctrl+B     Build Configurations     ARM E     Build Working Set     Scu	≠ 🖩 🗊 🔄 🔹	및 · 영 · 수 · Quick	Access
<ul> <li>P e src</li> <li>P e system</li> <li>P include</li> <li>P ≥ Idscripts</li> </ul>	Clean_ Build Automatically Make Target Make Target C/C++Index	<pre>d istdlib.h diag/Trace.l main(int, ch (trace via DEBUG). ing the TRACE macro defini re forwarded to the DEBUG ce or completely suppresse ed in system/src/diag/trac _USE_TRACE_SEMIHOSTING_DEE</pre>		<ul> <li>diag/Trace.h</li> <li>main(int, char*[]): int</li> </ul>
(2)	Problems 🕮 🧟 Tasks 🗳 Console 🔳 Properties			~
0.1	tems			
D	Description	Resource	Path	Location Ty

- Go to  $\textbf{C/C+} + \textbf{Build} \rightarrow Settings$ 

- Select the desired configuration

- Click on **Device** tab and expand the Analog Devices node

- Select ADuCM360 as a device and press OK

pe filter text	Settings		$(\neg \bullet \circ) \bullet$
Resource Builders C/C++ Build Build Variables	Configuration: Debug [ Active ]		▼ Manage Configurations
Environment Logging	🖲 Tool Settings 🖲 Toolchains 🔳	Parsers G Error Parsers	
Settings	Device selection (Not yet used duri	ng build)	
Tools Paths	Name	Details	
C/C++ General	<ul> <li>Boards</li> </ul>		
Project References	> Analog Devices	Vendor	
Null/Debug settings	<ul> <li>Devices</li> </ul>		
	<ul> <li>Analog Devices</li> </ul>	Vendor	
	<ul> <li>ADuCM36x Series</li> </ul>	Family (Cortex-M3, r2p0, 16 MHz, 8 kB RAM	4, 128 kB ROM)
	ADuCM360	Device	
	ADuCM361	Device	
	Device core: Cortex-M3		
	Memory map (Warning: Not yet us ADuCM360	d to generate the linker scripts!)	
	Section Start Size	Start	
	IRAM1 0x20000000 0x200	0	
	IROM1 0x0000000 0x200	0 1	
	Edit		
	1		
	1		

# Hardware

This chapter contains hardware-related information about the base board and the various shields. Each sub-section contains a general description of the board, detailed description of the connectors, jumpers, and buttons (if any). It also provides links to the Schematics, Bill of materials, design projects, and Technical documentation. It also gives internal links to the provided example demo software projects.

The following boards are currently available:

- EVAL-ADICUP360 Base Board
- EVAL-CN0216-ARDZ Shield
- EVAL-CN0357-ARDZ Shield

# **EVAL-ADICUP360** Base Board

The EVAL-ADICUP360 base board consists of two basic blocks:

- A fully integrated, 3.9 kSPS, 24-bit data acquisition system that incorporates dual high performance, multichannel sigma-delta (Σ-Δ) analog-to-digital converters (ADCs), a 32-bit ARM Cortex<sup>™</sup>-M3 processor, and Flash/EE memory, realized on a single chip **ADuCM360 microcontroller**.
- An on-board SWD interface, based on the OpenSDA platform, which is implemented with the **Freescale's K20DX128 microcontroller**. This block allows using a free Software Development Toolchain to program and debug the ADuCM360 microcontroller part.

This page describes the hardware connectors, the jumpers and switches configuration options, the USB connectors, and links to download the schematics and the layout.



## Connectors

The following connectors are populated on the base board:



- DC Power Jack: Core positive, accepts +7V to +12V DC supply voltage;
- DEBUG USB: Used for flash programming and debug interface;
- USER USB: Provides a Virtual serial port connection to ADuCM360 microcontroller;
- PMOD\_SPI: 12-pin SPI PMOD connector;
- PMOD\_I2C: 8-pin I2C PMOD connector;
- Six Arduino connectors described in the table below.

Connector	Pin No.	Pin Name	ADuCM360 pin or other function	Arduino Due Pin Name
	10	SCL	P2.0/SCL/UARTCLK	SCL1
	9	SDA	P2.1/SDA/UARTDCD	SDA1
	8	AREF	VREF+	AREF
PWMH	7	GND	AGND (Analog ground)	GND
	6	SCK	P0.1/SCLK1/SCL/SIN	PWM13
	5	MISO	P0.0/MISO1	PWM12
	4	MOSI	P0.2/MOSI1/SDA/SOUT	PWM11
	3	SS	P0.3/IRQ0/CS1	PWM10
	2	P0.4	P0.4/RTS/ECLKO	PWM9
	1	P0.5	P0.5/CTS/IRQ1	PWM8

Connector	Pin No.	Pin Name	ADuCM360 pin or other function	Arduino Due Pin Name
	8	PWM5	P2.2/BM	PWM7
	7	PWM4	P1.4/PWM2/MISO0	PWM6
	6	PWM3	P1.3/PWM1/DSR	PWM
	5	PWM2	P1.2/PWM0/RI	PWM4
	4	PWM1	P1.1/IRQ4/PWMTRIP/DTR	PWM3
	3	PWM0	P1.0/IRQ3/PWMSYNC/EXTCLK	PWM2
	2	ТХ	P0.7/POR/SOUT	TX0
	1	RX	P0.6/IRQ2/SIN	RX0
	8	P0.2	P0.2/MOSI1/SDA/SOUT	TX3
	7	P0.1	P0.1/SCLK1/SCL/SIN	RX3
	6	P1.7	P1.7/IRQ7/PWM5/CS0	TX2
	5	P1.6	P1.6/IRQ6/PWM4/MOSI0	RX2
COMMUNICATION	4	P1.5	P1.5/IRQ4/PWM3/SCLK0	TX1
	3	P1.4	P1.4/PWM2/MISO0	RX1
	2	SDA	P2.1/SDA/UARTDCD	SDA
	1	SCL	P2.0/SCL/UARTCLK	SCL
	1	A8	AIN8/EXTREF2IN-	A8
	2	A9	AIN9/DACBUFF+	A9
	3	A10	AIN10	A10
	4	A11	AIN11/VBIAS1	A11
ADCH	5	DAC	DAC	DAC0
	6	G_SW	GND_SW	DAC1
	7	VREF+	VREF+	CANRX
	8	VREF-	VREF-	CANTX
	1	A0	AINO	A0
	2	A1	AIN1	A1
	3	A2	AIN2	A2
	4	A3	AIN3	A3
ADCL	5	A4	AIN4/IEXC	A4
	6	A5	AIN5/IEXC	A5
	7	A6	AIN5/IEXC	A6
	8	A7	AIN7/VBIAS0/IEXC/EXTREF2IN+	A7
	1	NC	- not connected -	NOT USED
	2	IOREF	DVdd (+3.3V)	IOREF
	3	RESET	RESET	RESET
	4	3.3V	DVdd (+3.3V)	3V3
POWER	5	5V	+5V	5V
	6	GND	DGND (Digital Ground)	GND
	7	GND	DGND (Digital Ground)	GND
	8	Vin	The input line of the +5V linear voltage regulator	VIN

Connector	Pin No.	Pin Name	ADuCM360 pin or other function	Arduino Due Pin Name
	1	MISO	P0.0	MISO
SPI	2	+5	+5	+5
	3	SCLK	P0.1	SCLK
	4	MOSI	P0.2	MOSI
	5	RESET	RESET	RESET
	6	GND	DGND	GND
	1	CS	P1.7	CHIP SELECT
	2	MOSI	P1.6	MOSI
	3	MISO	P1.4	MISO
	4	SCLK	P1.5	SCLK
	5	GND	DGND	GND
	6	VDD	DVDD	VDD
	7	INT	P1.0	INT
	8	RESET	P1.1	RESET
	9	GPIO	P1.2	GPIO
	10	GPIO	P2.2	GPIO
	11	GND	DGND	GND
	12	VDD	DVDD	VDD
	1	SCL	P2.0	SCL
	2	SCL	P2.0	SCL
	3	SDA	P2.1	SDA
	4	SDA	P2.1	SDA
	5	GND	DGND	GND
	6	GND	DGND	GND
	7	VDD	DVDD	VDD
	8	VDD	DVDD	VDD

## **Jumper Configuration**

There are **3 jumpers groups** on the EVAL-ADICUP360 base board:


### Jumper P12

Configuration	Function
P12	ADuCM360 <b>is powered</b> from the linear voltage regulator on the baseboard
P12 1 ②	ADuCM360 <b>is not powered</b> from the baseboard and may be powered from the application shield

### Jumper REFnSel

Configuration	Function
(C)	ADuCM360 VREF- pin connected to Analog GND
REFn Sel ©	ADuCM360 VREF- pin connected to the ADCH connector, pin 8

### Jumpers J1, J2, J3, J4, J5

Configuration	Function
J1 J3 J5 1 2 1 2 1 2 1 2 J2 J4	ADuCM360's UART pins <b>are connected</b> to the Virtual serial port of the Debug adapter
J1 J3 J5 1 2 1 2 1 2 1 2 J2 J4	ADuCM360's UART pins <b>are not connected</b> to the Virtual serial port of the Debug adapter
J1 J3 J5 1 2 1 2 1 2 1 2 1 2 J4	ADuCM360's SWD lines <b>are connected</b> to the Debug adapter. ADuCM360 can be programmed
J1 J3 J5 1 2 1 2 1 2 1 2 1 2 J2 J4	ADuCM360's SWD lines <b>are not connected</b> to the Debug adapter. ADuCM360 cannot be programmed
J1 J3 J5 1 2 1 2 1 2 1 2 1 2 J2 J4	ADuCM360's RESET line <b>is connected</b> to the Debug adapter. The button <b>B1</b> can be used to invoke the Debug adapter's Bootloader.
J1 J3 J5 1 2 1 2 1 2 1 2 1 2 J2 J4	ADuCM360's RESET line <b>is not connected</b> to the Debug adapter. The button <b>B1</b> is just an ADuCM360 reset button.

## **USB/Connector Multiplexer**

There are **4 switches** on the EVAL-ADICUP360 base board, which are used to multiplex pairs of pins **(P0.1/P0.2, and P0.6/P0.7)** to various different connectors on the board. Depending on how the pins are configured you may route them to the **USB ports**, use them for **SPI communication** or for **UART communication**.



#### Switches S1, S2, S3, S4

The **S1**, **S2**, **S3**, **S4** switches are used to route the P0.1/SCLK1/SCL/SIN, P0.2/MOSI1/SDA/SOUT, P0.6/IRQ2/SIN and P0.7/POR/SOUT pins when they have been assigned a UART function to either the **Arduino I/O** and the **PMOD** connectors or to the Virtual Serial ports implemented via the **USER USB** or the **DEBUG USB** connectors. Each pin can be routed separately, but the routing is usually done for the pairs TxD/RxD.

Most commonly used configurations are given in the table below. For any other more 'exotic' configuration, consult with the Schematics and the Layout of the board.

ADuCM360's pair of pins	Required connection	Configuration			
	to the User USB (FT232RL)				
P0.1/SCLK1/SCL/SIN P0.2/MOSI1/SDA/SOUT	to the Debug USB (mbed's Serial Port)				
	to the Arduino PWMH (pin 6, pin 3) and the SPI header (pin 1, pin 3)				

ADuCM360's pair of pins Required connection		Configuration		
P0.6/IRQ2/SIN	to the User USB (FT232RL)	$\begin{bmatrix} 1 \\ 54 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ $		
P0.7/POR/SOUT	to the Arduino PWML (pin 1, pin 2)			

#### **Switch Schematic**

Here is the schematic of the switching network, the switches allow to route the P0.1/P0.2 and P0.6/P0.7 signals to multiple connector depending how you want to configure the pins. Above are the common configurations, but for complete details please reference the diagram.



### **Buttons**

The EVAL-ADICUP360 base board provides two buttons **RESET** and **BOOT**.



Button	Function
RESET	Provides a hardware RESET to ADuCM360 microcontroller. If the RESET line is connected to the Debug adapter, this button can be used to invoke the Debug adapter's Bootloader, see section Jumper Configuration.
BOOT	When BOOT is held down during the reset and after, the ADuCM360 microcontroller enters UART download mode via P0.1 and P0.2. In this case, the user can download program via DEBUG USB or USER USB, depending on the jumpers settings, see section Jumper Configuration. BOOT button should be held press while a reset from the button is performed.

# **EVAL-CN0216-ARDZ Shield**

CN-0216 is a precision weigh scale signal conditioning system. It uses the AD7791, a low power buffered 24-bit sigma-delta ADC along with dual external ADA4528-2 zero-drift amplifiers. This solution allows for high dc gain with a single supply.

Ultralow noise, low offset voltage, and low drift amplifiers are used at the front end for amplification of the low-level signal from the load cell. The circuit yields 15.3 bit noise-free code resolution for a load cell with a full-scale output of 10 mV.



This circuit allows great flexibility in designing a custom low-level signal conditioning front end that gives the user the ability to easily optimize the overall transfer function of the combined sensor-amplifier-converter circuit. The AD7791 maintains good performance over the complete output data range, from 9.5 Hz to 120 Hz, which allows it to be used in weigh scale applications that operate at various low speeds.

### **Connectors and Jumper Configurations**

#### **Sensor Connector**



Pin Number	Pin Function
Pin 1	Not Used
Pin 2	- Excitation
Pin 3	+ Signal
Pin 4	- Sense
Pin 5	+ Sense
Pin 6	- Signal
Pin 7	+ Excitation
Pin 8	Not Used

### **Bridge Configuration**



**NOTE** - Any 4 or 6 wire load cells can be used with the **EVAL-CN0216-ARDZ**. The Tedeah Huntleigh Model 1042 load cell was used during testing.



Position "0" (shown below) is used for

6-wire resistive bridges

- P2 Connects REFIN+ to Sensor +Sense pin
- P3 Connects REFIN- to Sensor -Sense pin



Position "1" is used for 4-wire resistive bridges

- P2 Connects REFIN+ to 5V supply
- P3 Connects REFIN- to GND

# **EVAL-CN0357-ARDZ Shield**

CN0357 single-supply, low noise, portable gas detector circuit using an electrochemical sensor. The Alphasense CO-AX carbon monoxide sensor is used in this example. Electrochemical sensors offer several advantages for instruments that detect or measure the concentration of many toxic gases. Most sensors are gas specific and have usable resolutions under one part per million (ppm) of gas concentration.

The circuit shown in below uses the ADA4528-2, dual auto zero amplifier, which has a maximum offset voltage of 2.5  $\mu$ V at room temperature and an industry leading 5.6  $\mu$ V/ $\sqrt{Hz}$  of voltage noise density. In addition, the AD5270-20 programmable rheostat is used rather than a fixed transimpedance resistor, allowing for rapid prototyping of different gas sensor systems, without changing the bill of materials. The ADR3412 precision, low noise, micropower reference establishes the 1.2 V common-mode, pseudo ground reference voltage with 0.1% accuracy and 8 ppm/°C drift. For applications where measuring fractions of ppm gas concentration is important, using the ADA4528-2 and the ADR3412 makes the circuit performance suitable for interfacing with a 16-bit ADC, such as the AD7790.



### **Connectors and Jumper Configurations**

PICTURE OF THE BOARD FILE with JUMPERS AND CONNECTORS HIGHLIGHTED

#### **Sensor Footprint**

**NOTE** - Three electrode electrochemical toxic gas sensors can be used with the **EVAL-CN0357-ARDZ** The footprint can accommodate 3 different sizes of sensors. The Alphasense CO-AX electrochemical gas sensor was used during

**Recommended PCB Sockets(for Alphasense Sensors)** 

• A Series Sensors - Mill-Max 0364-0-15-15-13-27-10-0

testing and programming.

- B Series Sensors Mill-Max 0294-0-15-15-06-27-10-0
- D Series Sensors Mill-Max 0667-0-15-15-30-27-10-0
- The sensor may be connected to the M1 footprint using the appropriate pin sockets

#### **Jumper P1 Settings**

- "0" position Sensor output connected to ADC(defualt)
- "1" position Sensor output connected to A1 pin of ANALOG header, for connection to external ADCs

# **Reference Designs**

This chapter contains various reference designs available for the base board and the various shields. Each sub-section describes the demo program, how to setup the hardware, how to obtain the source code, and finally, how to import the project in the Eclipse workspace and to run it.

The following reference designs are currently available:

- **Blinking LEDs Demo** Shows the basic steps of creating a new project for the EVAL-ADICUP360 base board, running and debugging the software.
- **Command Line Interpreter Demo** A Command Line Interpreter (CLI) demo project for the EVAL-ADICUP360 base board.
- Accelerometer Demo Illustrates the functionality of the ADXL362 3-axes accelerometer. It works with the EVAL-ADXL362-ARDZ Shield.
- Weigh Scale Demo Weigh Scale measurement example for 4-/6-wire bridge sensors. It works with the EVAL-CN0216-ARDZ Shield.
- **pH Measurement Demo** pH Measurement System with Temperature Compensation that works with the EVAL-CN0326-PMDZ Pmod.
- Data Acquisition for Input Current Demo Handles data of the acquisition system for 4-20 mA inputs current that works with the EVAL-CN0336-PMDZ Pmod.
- RTD Temperature Measurement Demo RTD temperature measurement example that works with the EVAL-CN0337-PMDZ Pmod.
- **Carbon Dioxide Gas Detection Demo** Non Dispursive Infrared Gas Detection that works with the EVAL-CN0338-ARDZ Shield.
- Toxic Gas(CO) Detection Demo Measuring toxic gases using electrochemical sensors that work with the EVAL-CN0357-ARDZ Shield.

# **Blinking LEDs demo**

The **ADuCM360\_demo\_blink** is the simplest possible demo project for the EVAL-ADICUP360, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

The project includes basic initialization - stopping the watchdog, configuring the system clock, disabling the clocks for all peripherals and setting two digital outputs for driving the two LEDs on the board: LED2 and LED3. The automatically generated code by the GNU ARM Eclipse Plug-ins provide a system tick interrupt at 1ms intervals and a simple delay function.

This project uses the low level drivers available for ADuCM360 microcontroller. It provide the possibility to choose the LEDs blinking method: use the delay function or use timer interrupt service.

When the project is compiled and run, the two LEDs flash alternatively in predefined intervals (1 second for delay function method and 0.5 seconds for timer interrupt method).





### Setting up the hardware

• To program the EVAL-ADICUP360, set the jumpers as shown in the next figure. The important jumpers are highlighted in red.



- Connect the PC to the EVAL-ADICUP360 via DEBUG USB
- Load the program and run it.

### Obtaining the source code

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

To learn how to import the ADuCM360\_demo\_blink project from the projects examples in the Git

repository, please click on How to import existing projects from the GIT Repository.

The source code and include files for the **ADuCM360\_demo\_blink** can be found in projects examples which comes with installer package, or the latest version of the project can be found on Github:

ADuCM360\_demo\_blink at Github

### Importing the ADuCM360\_demo\_blink project

The necessary instructions on how to import **ADuCM360\_demo\_blink** project in your workspace can be found in the section, Import a project into workspace.

### Debugging the ADuCM360\_demo\_blink project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug it. To do this, follow the instructions from Setting up a Debug Configuration.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.



• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

### **Project structure**

The **ADuCM360\_demo\_blink** project use basic ARM Cortex-M C/C++ Project structure:

င်္ခ Project Explorer 🖾	🗎 🔽 🗸 🗸 🖻	° 🗆
⊿ 🕞 > ADuCM360_demo_blink [E\	/AL-ADICUP360 master]	
Binaries		
b j Includes		
🔺 🚑 src		
blink.c		
b 📝 main.c		
Fimer.c		
🔺 🚑 system		
a 🗁 include		
ADuCM360		
Er arm		
CMSIS		
Cortexm		
b 🗁 diag		
4 🗁 src		
ADuCM360		
CMSIS		
Cortexm		
D Coordinate		
D Cornewlib		
Debug		
4 🔄 include		
blink.h		
Timer.h		
Idscripts		

In the **src** and **include** folders you will find the source and header files related to blink application. You can modify as you wanted those files.

Here you can configure:

- **LEDs blinking method**: in order to use LEDs blinking in a Timer 0 interrupt routine you need to set use\_irq parameter to 1 (main.c). When use\_irq = 0 then you use only a delay function for LEDs blinking.
- **Time for blinking delay**: *BLINK\_TIME* (*blink.h*).

The **system** folder contains system related files (try not to change these files):

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: *ADuCM360.h* (registers definitions), system\_ADuCM360.c/h (system clock), vectors\_ADuCM360.c (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).

# **Command Line Interpreter Demo**

The **ADuCM360\_demo\_cli** is a Command Line Interpreter (CLI) demo project for the EVAL-ADICUP360 base board, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

The purpose of this project is to help you to get used with UART peripheral of **ADuCM360** microcontroller. The source code example can serve as a template for a resident command line interpreter, complementing any other user application functionality. Interrupt-based receiving of text commands from the UART is implemented. As soon as a command is entered, an execution request flag is raised to signal the main loop. The commands are recognised and may be executed immediately or later depending on the priority of the current tasks.

You can use any Terminal session you want, such as **Putty** or **Serial Terminal with Eclipse Kepler** (incorporated in Eclipse environment).

A serial connection of a PC to the EVAL-ADICUP360 board using the user USB connector is required to test and use the CLI application (**EVAL-ADICUP360 board** incorporates an FTDI USB-to-serial converter). Any terminal application run on a PC at 9600-8-N-1 without flow control can be used to 'talk' to the **EVAL-ADICUP360 board**. After connecting and sending CR (by pressing Enter), the command prompt '»' and a welcome message should appear.

COM16 - PUTTY		
Welcome to Com	mand Line Interpreter !!!	*
>>Type in <help> to see &gt;&gt;help</help>	available commands.	
Available commands: help version dump [begaddr][,count] reset	<ul> <li>Display available commands</li> <li>SW version</li> <li>Dumps the memory</li> <li>Reset application</li> </ul>	
>>version SW version of CLI Demo:	V0.1	
>>dump 0 0x20 HEX dump of the memory 00000000: 20002000 0000 00000010: 00000275 0000 >>	(0x20 bytes from 0x00000000); 001F5 000001FD 00000201 00279 000002C1 00000000	
>>reset Welcome to Comm	mand Line Interpreter !!!	
>>Type in <help> to see &gt;&gt; &gt;&gt;</help>	available commands.	-

#### **Available commands**

Command	Description
help	Display available commands
version	Display SW version of CLI project
dump [begaddr] [count]	Display up to 0x40 consecutive byte-size locations from any address of the ADuCM360 memory space. One should be careful not to request locations which are not decoded because the hardware_fault exception code will block the board.
<i>reset</i> Perform a HW reset which also initialize the application	

### Setting up the hardware

In order to program the EVAL-ADICUP360 you need to use the **DEBUG USB**. The jumper set up is shown in the next figure. The important jumpers are highlighted in red.



The ADuCM360\_cli\_demo can connect to the serial port of a PC through two different USB ports on the board:

- USER USB (using P0.1, P0.2 or P0.6, P0.7 of the ADuCM360)
- **DEBUG USB** (only P0.1, P0.2 of the ADuCM360)

A bank of jumpers provided near the PMOD ports of the EVAL-ADICUP360, makes this easy to configure. The jumpers required for particular configurations are provided in the images below. Ensure that the pins you select in the hardware configuration, also match what is in your software pin definition.(*UART\_PINS*)

#### Using UART via USER USB (P0.1, P0.2)



Using UART via USER USB (P0.6, P0.7)



Using UART via DEBUG USB (P0.1, P0.2)



If using UART in **USER USB** configuration, you first need to program the board using **DEBUG USB** 

If using UART in **DEBUG USB** configuration you first need to program the board using **DEBUG USB** and after the program runs on target, you need to change jumper (J1 and J2) positions

## **Obtaining the source code**

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

To learn how to import the **ADuCM360\_demo\_cli** project form the projects examples in the Git repository, please click on How to import existing projects from the GIT Repository.

The source code and include files of the **ADuCM360\_demo\_cli** can be found in the projects examples which comes with the installer package, or the latest version of the project can be found on Github:



ADuCM360\_demo\_cli at Github

### Importing the ADuCM360\_demo\_cli project

The necessary instructions on how to import **ADuCM360\_demo\_cli** project in your workspace can be found in the section, Import a project into workspace.

## Debugging the ADuCM360\_demo\_cli project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the ADuCM360\_demo\_cli project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.

Debug - ADuCM348, demo, cls/sec/main.c - Analog Devices In	N. ADUCMOR DE							
File Edit Source Refactor Navigate Search Project	Run Window Help			2 51 x 45 x				
			5	Quick Access		L C/C++	L GA (D) D	ebug 🐁 Packs
O Debug C	16 at 14 TT 0	N- Variables 98	Breakpoir	ts IIII Registers	2 Periph	erals 22	Module:	
ADuCMINE_demo_cli Debug (SDB OpenOCD Debugging     ADuCMINE_demo_cli.eff     ADuCMINE_demo_cli.eff     Demot #1 Companded : Restancief()	91	Peripheral		Address	Descriptio			148
i≣ man() st main.c.46 ba54 sill operocet.co: sill ann-none-eabi-g6b		Add ADC3 8-M8030884 Analog to Digital Converter     Add ADC1 8-M8030884 Analog to Digital Converter     Add ADC1 8-M803086 Analog to Digital Converter     Add ADC2118 8-M803086 Analog to Digital Converter     Add ADC2118 8-M803086 Analog to Digital Converter				weiter weiter weiter weiter		
		4						
<pre>immarc II \(C)gdd[]predCHUMPHEADeopld[gdd] '/Toherupt handler () {     clf_Int_Handler();     clf_Int_Handler</pre>	()proc(4000).comves()).thread)	[Arawe]9]				utine 11 11 stdio/i 12 stdio/i 12 times 12 times 12 times 12 times 12 times 12 times 12 times 13 times 13 times 14 times 14 times 14 times 14 times 15 times 15 times 15 times 16 times 16 times 16 times 16 times 17 times 16 times	E 2 k	viil
Console II Takin II Problems O Decudation ( Abu/CASB demo, c6 Debug 1008 OpenOCD Debugging) openo (32) det 2 Function (/32) (34) det 2 Function (/32) (35) det 2 Function (/32) (35) det 2 Function (/32) (36) det 3 Function (/32)	() Memory accluse			• X % 8	21 30   		9 2 9	
5. C.	Wota	ble Sme	rtinier	3:1				1

• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

### **Project structure**

The **ADuCM360\_demo\_cli** project use basic ARM Cortex-M C/C++ Project structure:

ADuCM360_demo_cli [EVAL-ADICUP360 master]
> 🖑 Binaries
Includes
🔺 🚑 src
Cli.c
En Communication.c
b in main.c
Timer.c
🔺 🚝 system
a 🗁 include
ADuCM360
b Cr arm
D CMSIS
Cortexm
Þ 📴 diag
4 🔊 src
ADuCM360
D CMSIS
Cortexm
b Corr diag
b Correction newlib
Debug
a 🔄 include
🕞 Cli.h
🚯 Communication.h
🖹 Timer.h
Idscripts

This project contains: initialization part - disabling watchdog, setting system clock, enabling clock for peripheral; UART interrupt service; port configuration for UART use; UART read/write management; command line interpreter application.

In the **src** and **include** folders you will find the source and header files related to CLI application. You can modify as you wanted those files. The *Communication.c/h* files contain UART specific data, meanwhile the *cli.c/h* files contain the command interpreter data.

Here you can configure:

- **UART pin configuration** *UART\_PINS* paramater use for P0.1, P0.2 connection UART\_PINS\_12 or use for P0.6, P0.7 connection UART\_PINS\_67 (*Communication.h*).
- **UART mode** UART\_MODE paramater interrupt or polling mode (Communication.h).
- UART baud rate available baud rates for serial port can be changed at initialization part (main).
- UART data bits 5 to 8 bits can be changed at initialization part (main).

The **system** folder contains system related files (try not to change these files):

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: *ADuCM360.h* (registers definitions), *system\_ADuCM360.c/h* (system clock), *vectors\_ADuCM360.c* (interrupt vector table).
- **cortexm** contains files for system management (start-up, reset, exception handler).

# Weigh Scale Measurement Demo

The **ADuCM360\_demo\_cn0216** is a weigh scale measurement demo project for the EVAL-ADICUP360 base board with additional EVAL-CN0216-ARDZ shield, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

This project is a good example for how to use EVAL-ADICUP360 board in different combinations with various shield boards. It expand the list of possible applications that can be done with the base board.

The **ADuCM360\_demo\_cn0216** project uses the EVAL-CN0216-ARDZ shield which is a precision weigh scale system using a **24-bits** sigma-delta converter, and auto-zero amplifiers providing high gain for the bridge sensor input



The CN0216 circuit translates the resistance changes on the bridge into very small voltages. The bridge is excited by a regulated 5V and that determines the full scale range of the bridge output. Those

values are passed through very low noise, auto zero amplifiers to remove as many error sources as possible before being gained up to levels that will be compatible with the ADC. The 24-bit ADC value is received via SPI interface of the EVAL-ADICUP360 board.

	COM30 - PuTTY	×
	Zero scale calibration measurement complete.	*
	Please place calibration weight on scale, to obtain full scale calibration point Press <enter> to calibrate.</enter>	h
The	Full scale calibration measurement complete.	
ADuCM360 demo	Press the <enter> key to measure and display the weigh scale data.</enter>	
_cn0216	ADC Data Register Value = 0xe95b30	
application	ADC Input Voltage input = 4.557738 V	
processes ADC	Sensor input weight = 999.962158 grams	н
output value and	ADC Data Register Value = 0xe958e3	
make all necessary	ADC Input Voltage input = 4.557563 V	
conversions in	Sensor input weight - 999.869629 grams	
	ADC Data Register Value = 0xe95f6a	
order to provide	ADC Input Voltage input = 4.558061 V	
the weight results.	Sensor Input Weight = 1000.132141 grams	
A UART interface	ADC Data Register Value = 0xe960e5	
(9600 baud rate	ADC Input Voltage input = 4.558174 V	
and 8-bits data	Sensor Input Weight = 1000.191711 grams	7

length) is used to send the results to terminal window: ADC Data Register **codes**, ADC Input Voltage **volts**, and Sensor Input Weight **grams** are the outputs provided in the terminal window.

At the start of the project, a calibration of the upper and lower input range of the weigh scale is taken to remove both offset and gain errors in the circuit, providing the most accurate weigh scale measurements possible. Make sure you open up the serial terminal to your PC in order to do the calibration. Once the program is running, it will ask you to make the zero scale calibration, you **MUST** press <ENTER> to begin the zero scale calibration(takes about 5 seconds). Once that calibration has taken place, the serial terminal will prompt you to add the calibration weight to the scale and then press <ENTER> to make the full scale calibration(again takes about 5 seconds). Those measurements are averaged over 100 samples and then stored into memory as the upper and lower calibration coefficients.

Once calibration is complete, measurements of the output values (weights and conversion information) are displayed every time you press <ENTER> key from the keyboard. Measurements should be between the lower and upper calibration limit can be made at the beginning of the program.

### Setting up the hardware

Connect the EVAL-CN0216-ARDZ to the Arduino connectors P4, P5, P6, P7, P8 of the EVAL-ADICUP360 board.

Extremely important to plug in an acceptable power supply to the barrel jack **P11** to supply power for the **EVAL-CN0216-ARDZ**. The boards will not work if you try only to power it from the DEBUG\_USB or the USER\_USB.

In order to program the base board you need to use the **DEBUG USB**, and you will need to use the **USER USB** to communicate with the serial terminal program. The important jumpers and switches configurations are highlighted in red.



The ADuCM360\_demo\_cn0216 uses **UART** connection via **P0.6/P0.7** and **SPI1** channel of the ADuCM360 to communicate with EVAL-CN0216-ARDZ shield.

## **Obtaining the source code**

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

The source code and include files of the **ADuCM360\_demo\_cn0216** can be found on Github:



### Importing the ADuCM360\_demo\_cn0216 project

The necessary instructions on how to import the **ADuCM360\_demo\_cn0216** project form the projects examples in the Git repository, can be found in How to import existing projects from the GIT Repository page.

### Debugging the ADuCM360\_demo\_cn0216 project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the ADuCM360\_demo\_cn0216 project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.

Debug II	1 II	09- Variables 🗣 Breakpoints IIII Registers 🖳 Per	phenals 🗋 💼 Modules 🐡 🖸	
Cebug II     ADuCM660, demo, cn0216, and: 2460 OpenOCD Debugging]     ADuCM660, demo, cn0216, and: aff     ADUCM660,		Peripheral Address Description Peripheral Address Description Analog to Digital Converter ADLADCD 0 0x40030000 Analog to Digital Converter ADLADCDDMA 0x40030000 Analog to Digital Converter ADLADCDMA 0x40030000 Analog to Digital Converter ADLADCDMA 0x40030000 Analog to Digital Converter ADLADC DMA 0x40030000 Click Control		
🗟 maine 11 👔 Communication.e 🛛 🗟 CN6216.e 🛛 🗟 A07760	.c 🕅 main.c 🔳 0x0	🖉 main.c 🔛 🗠 🗇	E Outine 10 - 0	
<pre>2 - oprages dcc diagnostic ignored "-viewuszd-parameter" 3   Byragma dcc diagnostic ignored "-viewiszing-declarati 4   Byragma dcc diagnostic ignored "-writising-declarati 4   De int main (int argc, char" argv[]) 4   E 4   J</pre>	ons*		Itdia.h     Itdia.h     Itdia.h     Itdia.h     AbucM360.h     Dotab.h     GotLab.h     GotLab.h     UtLab.h     UtLab.h     UtLab.h     Communication.h     diagTrace.h     Timer.h	
Contole Tasks Problems O Executables II Memory				
Aonitors 🔶 🗶 👻				

• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

## **Project structure**

The **ADuCM360\_demo\_cn0216** project use ADuCM36x C/C++ Project structure.

This project contains: system initialization part - disabling watchdog, setting system clock, enabling clock for peripherals; port configuration for SPI1, UART via P0.6/P0.7; SPI, UART read/write functions; AD7791 control and weight conversions.

In the **src** and **include** folders you will find the source and header files related to CN0216 software application. The *Communication.c/h* files contain SPI and UART specific data, meanwhile the *AD7791.c/h* files contain the ADC control data and the *CN0216.c/h* files contain the calibration and measurements management.

a 🚰 ADuCM360_demo_cn0216_ardz	
Binaries	In the appropriate header files you can <b>configure</b>
Includes	next parameters:
🔺 🚰 src	
⊳ 💽 AD7791.c	
⊳ 💽 CN0216.c	
Communication.c	
b 💼 main.c	
Dimer.c	
🖌 🔁 system	
🔺 🗁 include	
ADuCM360	
þ 🗁 arm	
CMSIS	
👂 🗁 cortexm	
þ 🗁 diag	
🔺 🗁 STC	
ADuCM360	
Dis CMSIS	
🔈 🗁 cortexm	
🔈 🗁 diag	
Description	
Debug	
a 🗁 include	
h AD7791.h	
lh CN0216.h	
Communication.h	
lh) Timer.h	
Idscripts	

• Converter reference voltage - VREF - reference voltage (V) for AD7791 converter (AD7791.h).

#define VREF

• **Full scale calibration weight** - CAL\_WEIGHT - this parameter can be set to the numeric value of the full scale calibration weight you are using. (in grams) (*CN0216.h*).

#define CAL\_WEIGHT 1000

The **system** folder contains system related files (try not to change these files):

5

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: *ADuCM360.h* (registers definitions), *system\_ADuCM360.c/h* (system clock), *vectors\_ADuCM360.c* (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).

# pH Monitor with Temperature Compensation Demo

The **ADuCM360\_demo\_cn0326** is a pH monitor with automatic temperature compensation demo project, for the EVAL-ADICUP360 base board with additional EVAL-CN0326-PMDZ pmod, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

This project is a good example for how to use EVAL-ADICUP360 board in different combinations with pmod boards. It expand the list of possible applications that can be done with the base board.

The **ADuCM360\_demo\_cn0326** project uses the EVAL-CN0326-PMDZ pmod which is a pH sensor signal conditioner and digitizer with automatic temperature compensation.



The CN0326 circuit provides a complete solution for pH sensors with internal resistance between **1**  $M\Omega$  and several  $G\Omega$ . It consist of **pH probe** buffer, **Pt1000 RTD** for temperature compensation and **24-bits ADC** with 3 differential analog inputs.

The pH probe consists of a glass measuring electrode and a reference electrode, which is analogous to a battery. When the probe is place in a solution, the measuring electrode generates a voltage depending on the hydrogen activity of the solution, which is compared to the potential of the reference electrode. As the solution becomes more **acidic** (pH < 7) the potential of the glass electrode becomes more positive (+**mV**) in comparison to the reference electrode; and as the solution becomes more **alkaline** (pH > 7) the potential of the glass electrode becomes more negative (-**mV**) in comparison to the reference electrode becomes more negative (-**mV**) in comparison to the reference electrode becomes more negative (-**mV**) in comparison to the reference electrode.

The change in temperature of the solution changes the activity of its hydrogen ions. When the solution is heated, the hydrogen ions move faster which result in an increase in potential difference across the two electrodes. In addition, when the solution is cooled, the hydrogen activity decreases causing a decrease in the potential difference. Electrodes are designed ideally to produce a zero volt ( $\mathbf{0}$  **V**) potential when placed in a buffer solution with a pH of 7 (**neutral** pH).

The **EVAL-CN0326-PMDZ** comes with an evaluation software which can help you to test and to calibrate your pmod before you use it.

Please visit CN0326 Software User Guide page to find out how to get and how to use the CN0326 evaluation software.

The potential changes are outputted as ADC 24-bits value which is received via SPI interface of the EVAL-ADICUP360 board. The ADC analog differential channels are:

- AIN1(+)/AIN1(-) pH probe (voltage full range: ±414 mV at 25°C to ±490 mV at 80°C)
- AIN2(+)/AIN2(-) Pt1000 RTD (voltage full range: 210 mV to 290 mV with 210 μA excitation current)
- AIN3(+)/AIN3(-) Bias current (used to minimized tne voltage errors)

BCOM11 - PuTTY		
Welcome to CN Type <help> to see av</help>	0326 application! ailable commands	<u>^</u>
:help		
Available commands: help calibrate <ch></ch>	<ul> <li>Display available commands</li> <li>Calibrate selected channel or all channels.</li> <li>Cob = BIN1, BIN2, BIN3 or all</li> </ul>	
temp ph reset	<ul> <li>Display temperature value</li> <li>Display pH value</li> <li>Reset ADC converter</li> </ul>	
:temp Temperature = 27.21[*	c]	
:ph pH = 3.11		
:reset AD7793 reset complete :	d1	-

The **ADuCM360\_demo\_cn0326** application purchase ADC outputs from input channels, calculates voltage, temperature and pH values. You can choose to use internal excitation current of the ADC (IOUT2) or calculate bias current of the circuit (see *USE\_IOUT2* parameter).

A UART interface (9600 baud rate and 8-bits data length) is used, as a command line interpreter, to send the results to terminal window: **temperature** and **ph** values. Beside this two the interpreter process other three commands: **help**, **calibrate** channel/channels and ADC **reset**.

To start the command line interpreter you need to press ENTER key (CR) from the keyboard and after

that just type in <help> to see available commands. The output data are send via UART using semihosting mechanism.

The project uses below formula to determine output **ADC code** for an input voltage on either channel:

$$Code = 2^{N-1} \left( \frac{AIN \times GAIN}{V_{REF}} + 1 \right)$$

AIN - analog input voltageGAIN - gain value in the in-amp settingN - ADC resolution (24)

The **temperature** value is calculated using RTD resistance value and it varies from 0°C (1000  $\Omega$ ) to 100°C (1385  $\Omega$ ):

$$T = \frac{R_{rtd} - R_{\min}}{\alpha \cdot R_{\min}}$$

**Rrtd** - RTD resistance at T°C **Rmin** - RTD resistance at 0°C  $\alpha$  - temperature coefficient (0.00385  $\Omega/\Omega/°C$ )

To calculate **pH** value is used Nernst equation:

$$E = a - \frac{2.303 R (T + 273.1)}{nF} \times (pH - pH_{ISO})$$

mol) **pHiso** - reference hydrogen ion concentration (7)

**E** - voltage of the hydrogen electrode with unknown activity

 $\alpha$  - zero point tolerance (±30 mV)

**T** - ambient temperature in °C

**n** - valence, number of charges on ion (1 at 25  $^{\circ}$ C)

**F** - Faraday constant (96485 coulombs/mol)

**R** - Avogadro's number (8314 mV-coulombs /°K

#### Semihosting with ARM

Semihosting is a mechanism that connect the target firmware's standard IO (printf, scanf/fgets, open, write, read, close, etc) to your host PC via JTAG or SWD. It's easy to configure it with open source tooling - the **newlib** C standard library and **OpenOCD JTAG** implementation.

You can automatic enable semihosting and configure it by using the project ADuCM36x C/C+ + Project template, which offer you the ability to select how do you want to use semihosting.

This example present the possibility to use semihosting to output messages with **printf()** by using a physical serial connection as UART. It uses the **newlib** GNU ARM library which actually links the UART physical port to standard C functions. You need only to overwrite **\_write()** function, which is marked as *weak* function in the GNU ARM library, with your own function that write characters to UART (the same for **\_read()** function when you want to use **scanf()** in your code).

### Setting up the hardware

Connect the **EVAL-CN0326-PMD** to the **SPI\_PMOD** connector of the **EVAL-ADICUP360** board. In order to program the base board you need to use the **DEBUG USB**. After you program the board you can switch to **USER USB** and you are set to use the application. The important jumpers and switches configuration are highlighted in red.



The ADuCM360\_cn0326\_demo use **UART** connection via **P0.6/P0.7** and **SPI0** channel of the ADuCM360 to communicate with CN0326 board.

### Obtaining the source code

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

The source code and include files of the ADuCM360\_demo\_cn0326 can be found on Github:



## Importing the ADuCM360\_demo\_cn0326 project

The necessary instructions on how to import the **ADuCM360\_demo\_cn0326** project form the projects examples in the Git repository, can be found in How to import existing projects from the GIT Repository page.

## Debugging the ADuCM360\_demo\_cn0326 project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the ADuCM360\_demo\_cn0326 project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon \*\* and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.

	n Project Run Window Help			
3 • 🗟 🕲 📓 🔺 🖉 📓 🖓 🗶 🖉	H 馬 図 6 % キ・O・	9. · 3 4 · 1 1 · 1 ·	00.	19.*
			Qu	ick Access 🛛 😰 C/C++ 🕸 Deb
Debug 2	※計14 マクロ	IIII Registers Modules	riphera	🕄 💊 Breakpoi 👓 Variables 🧮
- C ADuCM360_demo_cn0326 Debug [GD8 Or	penOCD Debugging)	and the second second		む 44 日
- 28 ADuCM360_demo_cn0326.elf	1 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	Peripheral	Ac	ddress Description
▲ P Thread #1 (Suspended : Breakpoint)		ADI ADI	(Doc	40030000 Analog to Digital Converte
main() at main.c.92 0x10a0		ADLADC1 0x40030080 Analog to Digital Conve		
🚽 openocd.exe		ADI_ADCOMA	Obe	400300F0 Analog to Digital Converte
📲 arm-none-eabi-gdb		ADI_ADCSTEP	Ove	400300E0 Analog to Digital Converte
		2 ADLANA	Ove	40008810 Analog Control
			- Ov	40002000 Clock Control
0wr50 @ main c 12 @ 4D 2703 c	CN02267 R CN0226 h		- m	Pr Outline to
60 **/	CHOSELC M CHOSECHI			
910 int main(int arec, char *arev[])				Stdioh
92 【			stdlib.h	
93				and the second s
93			1000	diag/Traceh
94 AD7793_Init();	/* AD7793 converter in:	itialization part */		diag/Trace.h
94 AD7793_Init(); 95 (N8326 Toit():	/* AD7793 converter in: /* CN8326 application in	itialization part */		<ul> <li>diag/Traceh</li> <li>Timer.h</li> <li>AD7793.h</li> </ul>
94 AD7793_Init(); 95 96 CN0326 Init():	/* AD7793 converter in: /* CN0326 application in	itialization part */		U diag/Traceh U Timer.h U AD7793.h
94 AD7793_Init(); 95 96 CN0326 Init(): 4 Problems © Console 3 Tasks O Exec	/* AD7793 converter in: /* CN0326 application in utables 0 Memory @ CN0326/	itialization part */		diag/Traceh     Timer.h     AD7793.h     Timer.h     AD7793.h     Timer.h     AD7793.h     Timer.h
93 94 AD7793_Init(); 95 96 CN0326 Init(); 7 Problems Console 23 Tasks O Exec DuCM360 demo cn0326 Debug (GDB Ogen00	/* AD7793 converter in: /* CN0326 application in utables	itialization part */ mitialization part */	) , ( )), ()	diag/Traceh     Timer.h     AD7793.h     Timer.h     AD7793.h     Timer.h     AD7793.h     Timer.h     AD7793.h     Timer.h
94 AD7793_Init(); 95 CN0326 Init(); 96 CN0326 Init(): 97 Problems Console 3 Tasks © Exec DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dut 2 comp (/32)	/* AD7793 converter in: /* CN0326 application in utables	itialization part */ nitialization part */ <pre># Search</pre>	) , k ( )k ()	diag/Traceh     Timer.h     AD7793.h     M
94 AD7793_Init(); 95 96 CN0326 Init(): 4 Problems Console 3 Tasks © Exect DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dut_2_comp (/32) 32) dut_2_mask (/4)	/* AD7793 converter in: /* CN0326 application in utables	itialization part */ mitialization part */ <pre># Search</pre>		diag/Traceh     Timer.h     AD7793.h     M
94 AD7793_Init(); 95 96 CN0326 Init(): 4 Problems Console 3 Tasks © Exec DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dut_2_comp (/32) 32) dut_2_mask (/4) 33) dut_2_function (/32)	/* AD7793 converter in: /* CN0326 application in utables	itialization part */ hitialization part */ <pre>%Search</pre>	- - -	u diag/Traceh u Timerh u AD7793h ∢ m → →
94 AD7793_Init(); 95 96 CN0326 Init(); 4 Problems  Console 3 Tasks  Exect DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dut_2_comp (/32) 32) dut_2_mask (/4) 33) dut_2_function (/32) 34) dut_3_comp (/32)	/* AD7793 converter in: /* CN0326 application in utables @ Memory @ CN0326.c D Debugging] openocd.exe	itialization part */ nitialization part */ <pre>     Search</pre>		u diag/Traceh u Timerh u AD7793h ∢ m → →
94 AD7793_Init(); 95 96 CN0326 Init(); 4 Problems  Console 33 Tasks ○ Exect DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dut_2_comp (/32) 32) dut_2_mask (/4) 33) dut_2_function (/32) 34) dut_3_comp (/32) 35) dut_3_mask (/4) 36) dut_3_function (/32)	/* AD7793 converter in: /* CN0326 epplication in utables	itialization part */ nitialization part */ <pre>     Search</pre>		u diag/Traceh u Timerh u AD7793h ∢ m →
94 AD7793_Init(); 95 96 CN0326 Init(); 4 Problems  Console ≅ Tasks  Exec DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dwt_2_comp (/32) 32) dwt_2_mask (/4) 33) dwt_2_function (/32) 34) dwt_3_comp (/32) 35) dwt_3_function (/32)	/* AD7793 converter in: /* CN0326 epplication in utables	itialization part */ nitialization part */ <pre>% Search</pre>		u diag/Traceh u Timerh u AD7793h ∢ m →
94 AD7793_Init(); 95 96 CN0326 Init(); 4 Problems  Console ≅ Tasks  Exec DuCM360_demo_cn0326 Debug[GDB OpenOC 31) dwt_2_comp (/32) 32) dwt_2_mask (/4) 33) dwt_2_function (/32) 34) dwt_3_comp (/32) 35) dwt_3_function (/32) 16	/* AD7793 converter in: /* CN0326 epplication in utables @ Memory @ CN0326.c D Debugging) openocd.exe	itialization part */ nitialization part */ <pre>     Search</pre>		U diag/Traceh U Timer.h U AD7793.h ∢ III → III (IIII) → III → IIII → III → III → IIII → IIII → IIII → IIII → IIII → III → IIII → IIII → III → I

• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

### **Project structure**

The **ADuCM360\_demo\_cn0326** project use ADuCM36x C/C++ Project structure.

This project contains: system initialization part - disabling watchdog, setting system clock, enabling clock for peripherals; port configuration for SPI0, UART via P0.6/P0.7; SPI, UART read/write functions; AD7793 control, voltage conversion, command interpreter, temperature and pH calculations.

In the **src** and **include** folders you will find the source and header files related to CN0326 software application. The *Communication.c/h* files contain SPI and UART specific data, meanwhile the *AD7793.c/h* files contain the ADC control data and the *CN0326.c/h* files contain the pH monitor application data.

ြာ Project Explorer 🖾	□ 🕏	
ADuCM360_demo_cn0326		
Includes		
🔺 😂 src		
▷ 🖻 AD7793.c		
▶ 🖻 CN0326.c		
Communication.c		
Main.c		
Timer.c		
4 🐸 system		
🔺 🗁 include		
ADuCM360		
🕨 🗁 arm		
Empiries		
👂 🗁 cortexm		
🖻 🗁 diag		
4 🗁 src		
ADuCM360		
👂 🗁 cmsis		
🖻 🗁 cortexm		
🕨 🗁 diag		
🕨 🗁 newlib		
🔺 🗁 include		
脑 AD7793.h		
🖹 CN0326.h		
Communication.h		
脑 Timer.h		
Idscripts		

In the appropriate header files you can **configure** next **parameters**:

• ADC gain - AD7793\_GAIN - POWER\_DOWN set gain value for AD7793 converter (AD7793.h).

#define AD7793\_GAIN

AD7793 GAIN 1

Excitation current - USE\_IOUT2 - select if you want to use bias current from the AIN3 channel: YES
or you want to use internal excitation current, 210 μA: NO(CN0326.h).

#define USE\_IOUT2 NO

• Zero point tolerance - TOLERANCE - tolerance used in Nernst equation (CN0326.h).

#define	TOLERANCE	Θ	ļ
			4

The **system** folder contains system related files (try not to change these files):

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: ADuCM360.h (registers definitions), system ADuCM360.c/h (system clock), vectors ADuCM360.c (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).
# **Data Acquisition for Input Current Demo**

The **ADuCM360\_demo\_cn0336** is a data acquisition demo project for 4-20 mA inputs, for the EVAL-ADICUP360 base board with additional EVAL-CN0336-PMDZ pmod, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

This project is a good example for how to use EVAL-ADICUP360 board in different combinations with pmod boards. It expand the list of possible applications that can be done with the base board.

The **ADuCM360\_demo\_cn0336** project uses the EVAL-CN0336-PMDZ pmod which is a completely isolated **12-bits**, **300 kSPS** data acquisition system (with only three active devices) that processes **4 mA** to **20 mA** input signals.



The CN0336 circuit consists of an input current-to-voltage converter, a level shifting circuit, an ADC stage and an output isolation stage. The **4 mA** to **20 mA** input signal is converted into **voltage** levels compatible with the input range of the ADC (**0 V** - **2.5 V**). The 12-bits ADC value is received via SPI interface of the EVAL-ADICUP360 board.

The **EVAL-CN0336-PMDZ** comes with an evaluation software which can help you to test and to calibrate your pmod before you use it.

Please visit CN0336 Software User Guide page to find out how to get and how to use the CN0336 evaluation software.



The **ADuCM360\_demo\_cn0336** application processes ADC outputs and provide current and voltage values. You can decide how often the ADC measurements take place (see *SCAN\_TIME* parameter).

A UART interface (115200 baud rate and 8-bits data length) is used to send the results to terminal window: **input current** value, **voltage** calculation and **ADC code**. If the input value is out of range you get an error message which means that you need to check your settings.

To start displaying data acquisition results on a terminal (putty in this case) you need to press ENTER key (CR) from the keyboard and after that the data are updated every time the input values are changed. The output data are send via UART using semihosting mechanism.

The project offers two method to calculate the input current, giving you the possibility to get more accurate results (see CN0336 circuit note). You can use **transfer function** of the circuit which calculate input current based on voltage changed value and circuit gain:

I = Imin + (Vout - Voffset)/Gain

Or you can use the **two-point calibration** method which used the ADC output values for 2 different measurements: first at Imin = 4 mA (ADC1) and second at Imax = 20 mA (ADC2):

Ix = Imin + [(Imax - Imin)/(ADC2 - ADC1)]\*(ADCx - ADC1)

#### Semihosting with ARM

Semihosting is a mechanism that connect the target firmware's standard IO (printf, scanf/fgets, open,

write, read, close, etc) to your host PC via JTAG or SWD. It's easy to configure it with open source tooling - the **newlib** C standard library and **OpenOCD JTAG** implementation.

You can automatic enable semihosting and configure it by using the project ADuCM36x C/C+ + Project template, which offer you the ability to select how do you want to use semihosting.

This example present the possibility to use semihosting to output messages with **printf()** by using a physical serial connection as UART. It uses the **newlib** GNU ARM library which actually links the UART physical port to standard C functions. You need only to overwrite **\_write()** function, which is marked as *weak* function in the GNU ARM library, with your own function that write characters to UART (the same for **\_read()** function when you want to use **scanf()** in your code).

### Setting up the hardware

Connect the **EVAL-CN0336-PMD** to the **SPI\_PMOD** connector of the **EVAL-ADICUP360** board. In order to program the base board you need to use the **DEBUG USB**. After you program the board you can switch to **USER USB** and you are set to use the application. The important jumpers and switches configuration are highlighted in red.



The ADuCM360\_cn0336\_demo use **UART** connection via **P0.6/P0.7** and **SPI0** channel of the ADuCM360 to communicate with CN0336 board.

### Obtaining the source code

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

The source code and include files of the **ADuCM360\_demo\_cn0336** can be found on Github:

AduCM360\_demo\_cn0336 at Github

## Importing the ADuCM360\_demo\_cn0336 project

The necessary instructions on how to import the **ADuCM360\_demo\_cn0336** project form the projects examples in the Git repository, can be found in How to import existing projects from the GIT Repository page.

## Debugging the ADuCM360\_demo\_cn0336 project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the **ADuCM360\_demo\_cn0336** project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon \*\* and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.



• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

### **Project structure**

The **ADuCM360\_demo\_cn0336** project use ADuCM36x C/C++ Project structure.

This project contains: system initialization part - disabling watchdog, setting system clock, enabling clock for peripherals; port configuration for SPI0, UART via P0.6/P0.7; SPI, UART read/write functions; AD7091R control and current-voltage conversion.

In the **src** and **include** folders you will find the source and header files related to CN0336 software application. The *Communication.c/h* files contain SPI and UART specific data, meanwhile the *AD7091R.c/h* files contain the ADC control data and the *CN0336.c/h* files contain the data acquisition parts.

ြာ Project Explorer 🛛 🕞 🔄 🗢 🗖 🗖	la tha anna àirte ha dha Channa an C
ADuCM360_demo_cn0336	In the appropriate header flies you can <b>configure</b>
Binaries	
Includes	
🔺 😂 src	
AD7091R.c	
▶ 🖻 CN0336.c	
Communication.c	
▶ 🖻 main.c	
Timer.c	
🔺 🐸 system	
🔺 🗁 include	
ADuCM360	
Þ 😅 arm	
🖻 🗁 cmsis	
🖻 🗁 cortexm	
🖻 🗁 diag	
🔺 🗁 src	
ADuCM360	
Þ 🗁 cmsis	
🖻 🗁 cortexm	
Þ 🗁 diag	
🖻 🗁 newlib	
🛛 🗁 Debug	
🔺 🗁 include	
脑 AD7091R.h	
📓 CN0336.h	
📓 Communication.h	
🚡 Timer.h	
Idscripts	

• **Converter operation mode** - AD7091R\_OPERATION\_MODE - POWER\_DOWN to select power-down AD7091R mode of operation or NORMAL for normal mode (*AD7091R.h*).

• **Converter scan time** - SCAN\_TIME - how often (msec) to read conversion results (*AD7091R.h*).

#define SCAN\_TIME

• Converter reference voltage - VREF - reference voltage (V) for AD7091R converter (AD7091R.h).

#define VREF

2.5

500

• **Current calculation formula** - CALC\_FORMULA - this parameter can be set as TRANSFER FUNCTION or TWO POINT CALIBRATION (*CN0336.h*).

```
#define CALC_FORMULA TWO_POINT_CALIBRATION
```

• Data acquisition parameters - all needed parameters for data calculations (CN0336.h).

#define	IMIN	4	/* Imin [mA] */
#define	IMAX	20	/* Imax [mA] */
#define	ADC_MIN	147	/* ADC min for IMIN */
#define	ADC_MAX	3960	/* ADC max for IMAX */

The **system** folder contains system related files (try not to change these files):

- **ADuCM360** contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: *ADuCM360.h* (registers definitions), *system\_ADuCM360.c/h* (system clock), *vectors\_ADuCM360.c* (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).

## **Test procedure**

The **ADuCM360\_demo\_cn0336** project was tested using the base HW configuration (EVAL-ADICUP360 board together with EVAL-CN0336-PMDZ pmod) and by using additional EVAL-CN0179-PMDZ pmod which was choose because it can generate the input current between required range **4mA** - **20 mA**.

In order to generate input current with **CN0179 circuit** is necessary just to use ADI available evaluation software for this pmod (CN-0179 Software User Guide).

## **RTD Temperature Measurement Demo**

The **ADuCM360\_demo\_cn0337** is a RTD temperature measurement demo project for the EVAL-ADICUP360 base board with additional EVAL-CN0337-PMDZ pmod, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

This project is a good example for how to use EVAL-ADICUP360 board in different combinations with pmod boards. It expand the list of possible applications that can be done with the base board.

The **ADuCM360\_demo\_cn0337** project uses the EVAL-CN0337-PMDZ pmod which is a completely isolated **12-bits**, **300 kSPS RTD** temperature measuring system (with only three active devices) that processes the output of a **Pt100 RTD** and includes an innovative circuit for lead-wire compensation using a standard **3-wire** connection.



temperature) into **voltage** levels compatible with the input range of the ADC (**0 V** - **2.5 V**). The 12-bits ADC value is received via SPI interface of the EVAL-ADICUP360 board.

The **EVAL-CN0337-PMDZ** comes with an evaluation software which can help you to test and to calibrate your pmod before you use it with an RTD sensor.

Please visit CN0337 Software User Guide page to find out how to get and how to use the CN0337 evaluation software.

COM11 - PUTTY		
Chec	CN0337 output data:	
Temperature	= 25.8981[*C]	
Resistance	= 110.0831[Ω]	
Voltage	= 0.3070[V]	
ADC Code	= 503 (0x1f7)	
Temperature	= 25.8981[°C]	
Resistance	= 110.0831[Ω]	
Voltage	= 0.3070[V]	
ADC Code	= 503 (0x1f7)	
Temperature	= 25.8215[°C]	
Resistance	= 110.0534[Q]	
Voltage	= 0.3064[V]	
ADC Code	= 502 (0x1f6)	
Temperature	= 25.9748[°C]	
Resistance	= 110.1129[g]	
Voltage	= 0.3076[V]	
ADC Code	= 504 (0x1f8)	

The **ADuCM360\_demo\_cn0337** application processes ADC output value and make all necessary conversions in order to provide RTD measure results. A UART interface (9600 baud rate and 8-bits data length) is used to send the results to terminal window: RTD **temperature** and **resistance** values, **voltage** calculation and **ADC code**. If the resistance and temperature values are out of range you get an error message which means that you need to check your settings.

The output values are displayed when you press ENTER key (CR) from the keyboard. Also you can decide how often the measurements take place (see *SCAN\_TIME* parameter).

The project offers two method to calculate the RTD resistance, giving you the possibility to get more accurate RTD measurement results (see CN0337 circuit note).

You can use **transfer function** of the circuit which calculate RTD resistance based on voltage changed value and circuit gain:

Rrtd = (Vout - Voffset)/Gain

Or you can use the **two-point calibration** method which used the ADC output values for 2 different measurements: first using Rmin = 100  $\Omega$  (ADC1) precision resistor and second with Rmax = 212.05  $\Omega$  (ADC2) resistor.

Rrtd = Rmin + [(Rmax - Rmin)/(ADC2 - ADC1)]\*(ADCrtd - ADC1)

Because the transfer function of the RTD (resistance vs. temperature) is nonlinear is needed a software linearization to eliminate the nonlinearity error of the RTD Pt100 sensor. This project used so called **Piecewise Linear Approximation** method.

#### **Piecewise Linear Approximation Method**

This method characterized by taking linear approximation one step further, one can conceptualize any number of linear segments strung together to better approximate the nonlinear RTD transfer function. Generating this series of linear segments so that each segment's endpoints meet those of neighboring segments results in what can be viewed as a number of points connected by straight lines.

These coefficients is calculated once to best match the RTD's nonlinear transfer function and then stored permanently in a look-up table (see *C\_rtd[]* table). From this table of coefficients, the software can perform simple linear interpolation to determine temperature based on measured RTD resistance.

The look-up table can have how many coefficients you needed depending how accurate you want to be. For this project the RTD resistance range is separated into 100 linearization segments.

This method is also used in the AN-709 application note which provide also an RTD coefficient generator tool that you also can use.

## Setting up the hardware

Connect the **EVAL-CN0337-PMD** to the **SPI\_PMOD** connector of the **EVAL-ADICUP360** board. In order to program the base board you need to use the **DEBUG USB**. After you program the board you can switch to **USER USB** and you are set to use the application. The important jumpers and switches configuration are highlighted in red.



The ADuCM360\_cn0337\_demo use **UART** connection via **P0.1/P0.2** and **SPIO** channel of the Rev 04 Dec 2015 11:31 | Page 3

ADuCM360 to communicate with CN0337 board.

### Obtaining the source code

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

The source code and include files of the **ADuCM360\_demo\_cn0337** can be found on Github:

AduCM360 demo cn0337 at Github

### Importing the ADuCM360\_demo\_cn0337 project

The necessary instructions on how to import the **ADuCM360\_demo\_cn0337** project form the projects examples in the Git repository, can be found in How to import existing projects from the GIT Repository page.

## Debugging the ADuCM360\_demo\_cn0337 project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the ADuCM360\_demo\_cn0337 project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.



• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

### **Project structure**

The **ADuCM360\_demo\_cn0337** project use ADuCM36x C/C++ Project structure.

This project contains: system initialization part - disabling watchdog, setting system clock, enabling clock for peripherals; port configuration for SPI0, UART via P0.1/P0.2; SPI, UART read/write functions; AD7091R control and RTD conversions.

In the **src** and **include** folders you will find the source and header files related to CN0337 software application. The *Communication.c/h* files contain SPI and UART specific data, meanwhile the *AD7091R.c/h* files contain the ADC control data and the *CN0337.c/h* files contain the RTD measurements management.

ြာ Project Explorer 🛛 📄 🔄 🔻 🗖 🗖	
ADuCM360_demo_cn0337	In the appropriate header files you can <b>configure</b>
Binaries	
Includes	
🔺 🐸 src	
▷ ▲ AD7091R.c	
▶ 🖻 CN0337.c	
Communication.c	
🖻 🖻 main.c	
Timer.c	
4 🐸 system	
🔺 🗁 include	
ADuCM360	
🖻 🗁 arm	
Cmsis	
👂 🗁 cortexm	
Þ 🗁 diag	
🔺 🗁 src	
ADuCM360	
🖻 🗁 cmsis	
🖻 🗁 cortexm	
🖻 🗁 diag	
🖻 🗁 newlib	
🛛 🗁 Debug	
🔺 🗁 include	
📓 AD7091R.h	
📓 CN0337.h	
📓 Communication.h	
📓 Timer.h	
Idscripts	

• **Converter operation mode** - AD7091R\_OPERATION\_MODE - POWER\_DOWN to select power-down AD7091R mode of operation or NORMAL for normal mode (*AD7091R.h*).

• Converter scan time - SCAN\_TIME - how often (msec) to read conversion results (AD7091R.h).

#define SCAN\_TIME 500

• **Converter reference voltage** - VREF - reference voltage (V) for AD7091R converter (*AD7091R.h*).

#define VREF

• **RTD resistance calculation method** - RTD\_FORMULA - this parameter can be set as TRANSFER\_FUNCTION or TWO\_POINT\_CALIBRATION (*CN0337.h*).

```
#define RTD_FORMULA TRANSFER_FUNCTION
```

• RTD parameters - all needed parameters for RTD calculations (CN0337.h).

#define	TMIN	(0)	<pre>/* Tmin [°C] */ /* Tmax [°C] */ /* Resistance [Ohms] at Tmin */ /* Resistance [Ohms] at Tmax */ /* Nr. of sections in look-up</pre>
#define	TMAX	(300)	
#define	RMIN	(100)	
#define	RMAX	(212.052)	
#define	NSEG	100	
<pre>table */ #define #define #define</pre>	RSEG	1.12052	/* Resistance of each segment */
	ADC_MIN	152	/* ADC min for RMIN */
	ADC_MAX	4095	/* ADC max for RMAX */

The **system** folder contains system related files (try not to change these files):

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: *ADuCM360.h* (registers definitions), *system\_ADuCM360.c/h* (system clock), *vectors\_ADuCM360.c* (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).

### **Test procedure**

The **ADuCM360\_demo\_cn0337** project was tested using the base HW configuration (EVAL-ADICUP360 board together with EVAL-CN0337-PMDZ pmod) and by connecting the 3-wire **PT100 CZUJNIK** temperature sensor.

# **Toxic Gas (CO) Measurement Demo**

The **ADuCM360\_demo\_cn0357** is a toxic gas(CO) detector demo project for the EVAL-ADICUP360 base board with additional EVAL-CN0357-ARDZ shield, created using the GNU ARM Eclipse Plug-ins in Eclipse environment.

### **General description**

This project is a good example for how to use EVAL-ADICUP360 board in different combinations with various shield boards. It expand the list of possible applications that can be done with the base board.

The **ADuCM360\_demo\_cn0357** project uses the EVAL-CN0357-ARDZ shield which is a single-supply, low noise, portable gas detector circuit using an electrochemical sensor.



The EVAL-CN0357-ARDZ shield circuit provides a potentiostatic circuit for biasing the electrochemical sensor, along with a programmable TIA and 16-bit Sigma-Delta ADC. The TIA converts the small currents passing in the sensor to a voltage that can be read by the ADC. The 16-bit ADC value is received via SPI interface of the EVAL-ADICUP360 board, where the gas concentration is computed.

	💆 COM7:9600baud - Tera Term VT 📃 📼 🔀	
	File Edit Setup Control Window Help	
The ADuCM360 demo	ADC Data Register Value = 0x007fda ADC Input Voltage input = -0.001392 V Gas Concentration = 2.322360 PPM	-
cn0357	ADC Data Register Value = 0x007fda ADC Japut Valtage japut = -0.001392 V	
_ application	Gas Concentration = 2.322360 PPM	
configures the	ADC Data Register Value = 0x007fda ADC Input Voltage input = -0.001392 V	
necessary	Gas Concentration = 2.322360 PPM	
components,	ADC Data Register Value = 0x007fda ADC Japut Valtage japut = -0.001392 V	
processes ADC	Gas Concentration = 2.322360 PPM	
output value and	ADC Data Register Value = 0x007fda ADC Joput Voltage joput = -0.001392 V	
make all necessary	Gas Concentration = 2.322360 PPM	
conversions in	ADC Data Register Value = 0x007fda ADC Japut Valtage japut = -0.001392 V	
order to provide	Gas Concentration = 2.322360 PPM	-
the gas		-

concentration. A UART interface (9600 baud rate and 8-bits data length) is used to send the results to terminal window: ADC Data Register **codes**, ADC Input Voltage **volts**, and Gas Concentration **Parts Per Million(PPM)** are the outputs provided in the terminal window.

At the start of the project, the software computes the necessary parameters and configure the digital rheostat(AD5270) of the TIA. The required parameters are the sensor sensitivity and sensor range. These can be modified by changing the values of the constants **SENSOR\_SENSITIVITY** and **SENSOR\_RANGE** found in the **CN0357.h** header file of the project. See the "*Project Structure*" section for more details.

Once configuration is complete, the software remains in a loop and continuously reads data from the ADC. Data can be read from a terminal by pressing the **<Enter>** key on the computer's keyboard.

### Setting up the hardware

Connect the EVAL-CN0357-ARDZ to the Arduino connectors P4, P5, P6, P7, P8 of the EVAL-ADICUP360 board.

Extremely important to plug in an acceptable power supply to the barrel jack **P11** to supply power for the **EVAL-CN0357-ARDZ**. The boards will not work if you try only to power it from the DEBUG\_USB or the USER\_USB.

In order to program the base board you need to use the **DEBUG USB**, and you will need to use the

**USER USB** to communicate with the serial terminal program. The important jumpers and switches configurations are highlighted in red.



The ADuCM360\_demo\_cn0357 uses **UART** connection via **P0.6/P0.7** and **SPI1** channel of the ADuCM360 to communicate with EVAL-CN0357-ARDZ board.

### Obtaining the source code

We recommend not opening the project directly, but rather import it into Eclipse and make a local copy in your Eclipse workspace.

The source code and include files of the ADuCM360\_demo\_cn0357 can be found on Github:



## Importing the ADuCM360\_demo\_cn0357 project

The necessary instructions on how to import the **ADuCM360\_demo\_cn0357** project form the projects examples in the Git repository, can be found in How to import existing projects from the GIT Repository page.

## Debugging the ADuCM360\_demo\_cn0357 project

- A debug configuration must be set up for this project in order to have the possibility to program and to debug the ADuCM360\_demo\_cn0357 project. To do this, follow the instructions from Setting up a Debug Configuration Page.
- Make sure the target board is connected to the computer (via **DEBUG USB**) and using the tool bar, navigate to the small Debug icon and select the debugging session you created. The application will programmed and the program execution will stop at the beginning of the main() function.

Colory Abort Mills down and Mills American American	NUT IN ADDINIO/102				= 🗆 X
File Edit: Source Refactor Nevigete Search Project	Run Window Help				
	+==  2 % 0.0.	9-9-2-20-50	50.0.	Quick Access and	C/C++ (B Debug
the Debuse 11	36 17 10 11	- D Othe Variables 12 Bas	akaninta IIII Remistera	R. Parisharah B. Madulat	= 0
<ul> <li>C cheminated &gt;ADuCM368_demo_cn8357 Debug (608</li> </ul>	OpenOCD Debugging)				
al <terminated, 0="" exit="" value:="">openocd.exe</terminated,>		Name	Name Type		
🚽 <terminated, 0="" exit="" value:="">arm-none-eabi-gdb</terminated,>				ar.	
		200			
		*			
@ main.c 12				Cuttine E Disastembly	
= =>= int madm(int args, char *argv[])				Enter location here	- 0.0000
st {					
<pre>86 float fAdeVoltage = 0;</pre>				No debug context	
<pre>87 float fConcentration = 0; 88 float fRCACvalue = 0;</pre>					1
#9 float fsensitivity = 0;					
90 uintă t ulăStatus Reg = 0;					
92 uinti6 t uli6ADACdata = 0;					
93			(78)		
95 /# Start the Sortem Tick Imag. #/			1		
96 timer_start();			1.00		
9					
<pre>99 CM2957 Initiative inels/ components */ 99 CM2957 Initiation</pre>					
100					
<pre>101 /* Initialize and set REAL value of t 102 AD5270_Init(FEEDEACK_RESISTOR);</pre>	he 405270 */				
104 /* Initialize the 407790 */					
105 AD7790_Init();			*	1.00	
The second second					
Console 2 Tasks C Prohama C O Executables					
Passielles *		Painting		Dut-	
Description		MERCURCE.		FRO	
• [					

• Use step-by-step execution or directly run the program.

After completion of the steps above the program will remain written into the system flash and it will run by default every time the board is powered up.

#### **Project structure**

#### The **ADuCM360\_demo\_cn0357** project use ADuCM36x C/C++ Project structure.

This project contains: system initialization part - disabling watchdog, setting system clock, enabling clock for peripherals; port configuration for SPI1, UART via P0.6/P0.7; SPI, UART read/write functions, AD7790 control, AD5270 control and gas concentration computation.

In the **src** and **include** folders you will find the source and header files related to CN0357 software application. The *Communication.c/h* files contain SPI and UART specific data, the *AD7790.c/h* files contain the ADC control, the *AD5270.c/h* files contain the rheostat control and the *CN0357.c/h* files contain configurations and computations specific to the gas detector application.



• **Sensor Range** - SENSOR\_RANGE - maximum value of the gas conentration (ppm) that can be detected by the electrochemical gas sensor being used (*CN0357.h*).

#define SENSOR\_RANGE 2000

• Sensor Sensitivity - SENSOR\_SENSITIVITY - sensitivity (nA/ppm) of the electrochemical sensor being used (*CN0357.h*).

#define SENSOR\_SENSITIVITY 65

The **system** folder contains system related files (try not to change these files):

- ADuCM360 contains low levels drivers for ADuCM360 microcontroller.
- **CMSIS** contains files related to ADuCM360 platform, such as: ADuCM360.h (registers definitions), system ADuCM360.c/h (system clock), vectors ADuCM360.c (interrupt vector table).
- cortexm contains files for system management (start-up, reset, exception handler).

# **Help and Support**

This page wants to help you when you have a specific issue which required a different approach or when the wiki information are not enough.

### ADuCM360 questions

If you have any questions regarding the **base platform** or any of the **shields/pmods** or are experiencing any problems while using the boards or while following any of the user guides feel free to ask us a question. Questions can be asked on our **SengineerZone support community**.

When asking a question please take the time to give a detailed description of your problem. If you are experiencing a problem please state the steps you have executed, the result you expected you would get and the result you actually got. By doing so you enable us to provide you precise and detailed answers in a timely manner.

Before asking questions please take the time to check if somebody else already asked the same question and already got an answer.

### **IDE** questions

If you need additional information about **Eclipse IDE** which is part of the EVAL-ADICUP360 Tool Chain you can visit GNU ARM Eclipse page.

# **ADICUP360 Compliance Results**

### Introduction

WRegulatory compliance means conforming to a rule, such as a specification, policy, standard or law. Most products that ships into a country need to pass a variety of tests and regulations specific to that country.

Due to the increasing number of regulations, organizations are increasingly adopting the use of consolidated sets of compliance controls. This means once you normally get one, you can have them all.

#### Reports

The ADICUP360 passes all requirements of the WCE tests.

• ADICUP360 EMC emissions and immunity test report

### What are all these logos?

- WCE Mark : a mandatory conformity marking for certain products sold within the European Economic Area (EEA).
- WElectrical and Electronic Equipment Waste Directive : a European Community directive 2002/96/EC on waste electrical and electronic equipment (WEEE).
- WFederal Communications Commission : is an independent agency of the United States government, this logo means we pass part 15, class B.