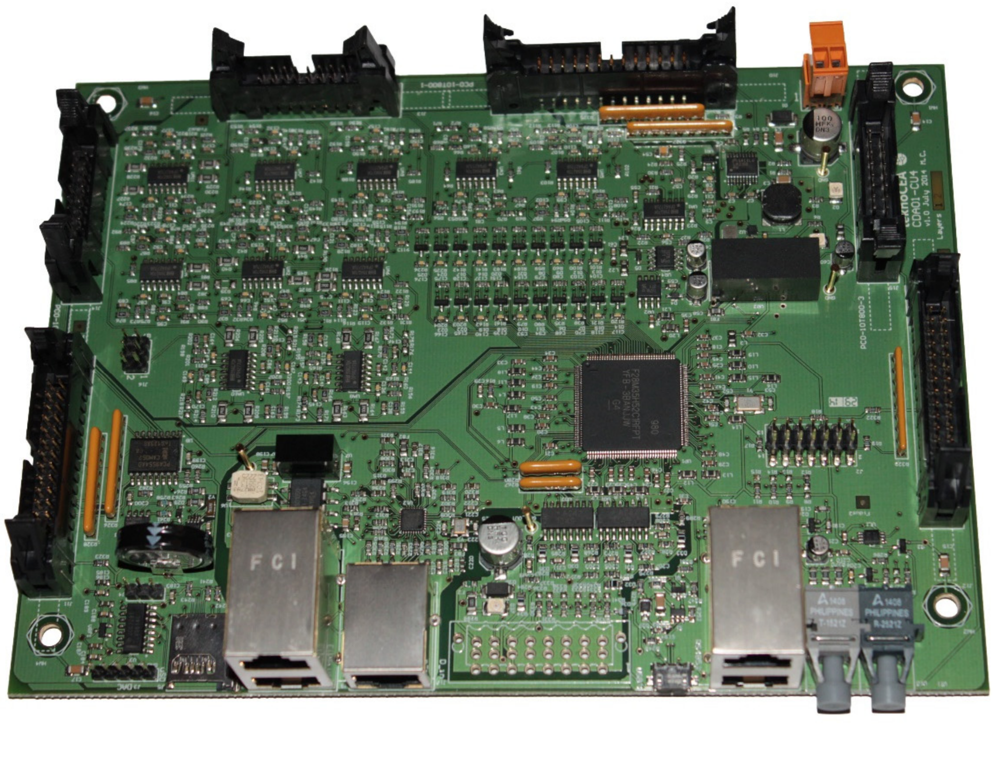
Introduction to programming the CDA01-CU3

Rev. 3.1

November 2019



SUPPORT

For support, please contact suport@teknocea.cat.

Contents

1. Introduction to Concerto by CCS 6.1.0

The control MCU in the CDA01-CU3 board is a Concerto F28M35H52C1 from Texas Instruments. The MCU is a multicore system-on-chip microcontroller unit with independent communication and real-time control subsystems. Concerto family microcontroller is programmed by Code Composer Studio (CCS). This manual uses CCS 6.1.0 and ControlSUITE both from Texas Instruments. Both have to be properly installed in the programming computer.

IMPORTANT NOTE: Compiler versions for these base programs must be:

- For C28xx core:	v6.4.4 or later
- For ARM core:	v5.2.4 or later

Be sure that these versions are installed for proper operation.

ControlSUITE is a free software from Texas Instruments useful for learning and programming all the TI family. Both can be downloaded from the TI web page (www.ti.com). A lot of examples and information is available on ControlSUITE and CCS. Most of them can be downloaded from http://www.ti.com/tool/controlsuite and http://processors.wiki.ti.com web pages.
Also, a JTAG debugging link must be available. teknoCEA provides the CDA03-JTAG2 JTAG-USB low cost isolated interface that can be used for programming and debugging.

1.1. Creating a new project

The document F28M35x-FRM-EX-UG that can be downloaded from TI webpage describes how to create and debug a new project.

1.2. ControlSUITE examples

Many examples for Control (DSP) and Master (M3) are available in ControlSUITE. To import the desired example project:

1. Menu File->Import
2. In the CCS folder select Existing CCS/CCE Eclipse Projects
3. Click Next
4. With the "Select search-directory" radio button checked, browse to the device_support sub-folder in the ControlSUITE folder
5. Navigate to the F25M35x directory
6. Examples can be found in the F28M35x_examplesControl, F28M35x_examplesMaster or F28M35x_examplesDual folders.
7. Click OK and CCS will parse all the projects in the folder
8. Import any projects you wish to run into workspace

These projects link to external resources with a relative path, so taking them out of ControlSUITE will break the project.

All these examples are programmed to be used by Concerto F28M35xx ControlCARD. To be used in the CDA01-CU3 control board, change GPIO functions according to CDA01-CU3 Technical Reference using API functions described in TI F28M35x Peripheral Driver Library.
If only C28x DSP project want to be used, example setup_m3 of M3 has to be executed. M3 is the master subsystem of the MCU, and minimal configuration must be done in order to start DSP.
This M3 example does the following:

- PLL Configuration - Configures the Concerto device to operate at the maximum speed for a 20 MHz input clock (75 MHz M3, 150 MHz C28).
- Enable All GPIO - Turns on all pins for use as GPIOs or alternate functions.
- Map all GPIO to control subsystem - Gives control of all GPIOs to the DSP control subsystem.
- Release control subsystem from reset - Deasserts reset driven to the control subsystem and allows it to run.

1.3. teknoCEA base project

Two base projects by teknoCEA are provided, DSP_base project and ARM_base project. These projects are based on ControlSUITE examples but adapted to CDA01-CU3 control board and PCO-10T800 power converter.
The two projects have to be imported in CCS. Both projects are programmed independent. Include search path must be modified in order to include the include folder for both projects. Also, MWare folder from ControlSUITE must be included at search path of the M3.

1.3.1 DSP_base project

Different DSP peripherals are programmed in this base project. The objective is to execute a basic control algorithm of a power converter. The base program has been designed for the CDA01-CU3 control board and the PCO-10T800 converter.
ADC, ePWM and GPIOs are configured and programmed in this base project. Two interrupts are used for the algorithm. The ADC interrupt is serviced on the End-Of-Conversion event of the ADC and is used for the control algorithm. The Trip Zone interrupt, TZ, is used to handle the error signal of the converter.
The main functions used in the program are:

- GPIO. Only one file and one function is needed to control the GPIO. The function *InitGpio()* is stored in the *Config_Gpio.c* file, and is used to configure all the GPIO functions. Refer to Concerto data manual for all the functions.
- ADC. ADC configuration functions are *InitAdc1()* and *InitAdc2()* and are placed in the *Config_ADC.c* file. ADC interrupt function is declared in the *main.c* file, and interrupt vector is placed in the *Interruption_ADC.c* file, pointing *Interrupcio_ADC()* interrupt function.
  This function implements the control routine at a fixed frequency synchronised with the PWM, because the PWM is configured to start the ADC at PWM period value.
  Inside ISR, ADC channels are read and PWM duty cycles for all the converters are passed to the PWM registers. Hardware and software alarms are managed also in this routine.

- PWM. The configuration function for the PWM is *InitEPwm()* and is placed in the *F28M35x_EPwm.c* file. In this case, ePWM1, ePWM2 and ePWM3 are programmed to control the three legs of the three phase power converter in the PCO-10T800.
- TZ. Trip Zone interrupt is a function called when the TZ pin detects a driver error. The interrupt function *Interrupt_TZ_EPWM123()* is declared in the *main.c* file. The function is stored in *Interrupt_TZs.c* file.
- MAIN. The main part of the *main.c* file is an infinite loop that determines the state of the system. A state machine is implemented to recognize this state.

1.3.2 ARM_base project

Different ARM peripherals are programmed in this base project. Each communication peripheral is configured according to CDA01-CU3 pinout. GPIOs that are not necessary for peripheral communications are mapped to Control Subsystem (DSP) by function *GiveGPIOControl_C28()*. Pinout is programmed in function *PinoutPeripheral()*.
Main functions used in the program are:

- CAN. CAN interface is configured using the *Config_CAN()* function. This function sends and incremental hexadecimal value pattern. The CAN message is sent using the *exe_CAN()* function.
- RS485. RS485 is configured using the *Config_RS485()* function. RS-485 is configured as an echo. Any value received is sent back followed by value 1. The functions are defined in *Interrupt_RS485.c* file
- DAC. DAC is configured using the *Config_DAC()* function. Sending data to the DAC is done with the *exe_DAC()* function. Set the J4 switch to DAC position in the CDA01-CU3 control board.

1.3.3 Errors while compiling DSP_base and ARM_base projects

While compiling for first time any base project, some errors may appear. For example, following errors will be solved with these steps:

#1965 cannot open source file "*Example_file.h*"

To solve this, make sure that include paths are well configured:

- ARM_base configuration:
    1. Select ARM_base project.
    2. Delete *Debug* folder if existing.
    3. On main tools bar: Project / Properties
    4. Build / ARM Compiler / Include Options
    5. Add dir to #include search path:
       "${CG_TOOL_ROOT}/include"
       "${PROJECT_LOC}"
    6. Build / ARM Linker / File Search Path
    7. Add <dir> to library search path:
       "${CG_TOOL_ROOT}/lib"
       "${CG_TOOL_ROOT}/include"
    8. Click "OK". Configuration of ARM_base project is finished.

- DSP_base configuration:
    1. Select DSP_base project.
    2. Delete *Debug* folder if existing.
    3. On main tools bar: Project / Properties
    4. Build / C2000 Compiler / Include Options
    5. Add dir to #include search path:

        "${CG_TOOL_ROOT}/include"

        "${PROJECT_LOC}/Include"
    6. Build / C2000 Linker / File Search Path
    7. Add <dir> to library search path:

        "${CG_TOOL_ROOT}/lib"

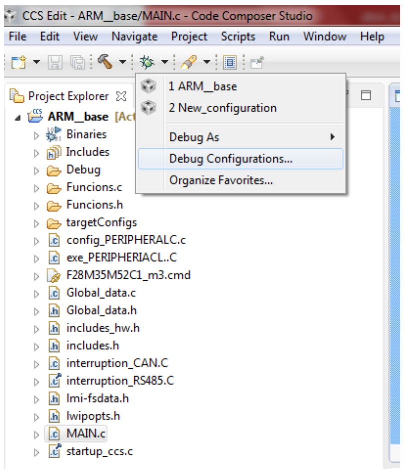
        "${CG_TOOL_ROOT}/include"

        "${PROJECT_LOC}/lib"
    8. Click "OK". Configuration of DSP_base project is finished.

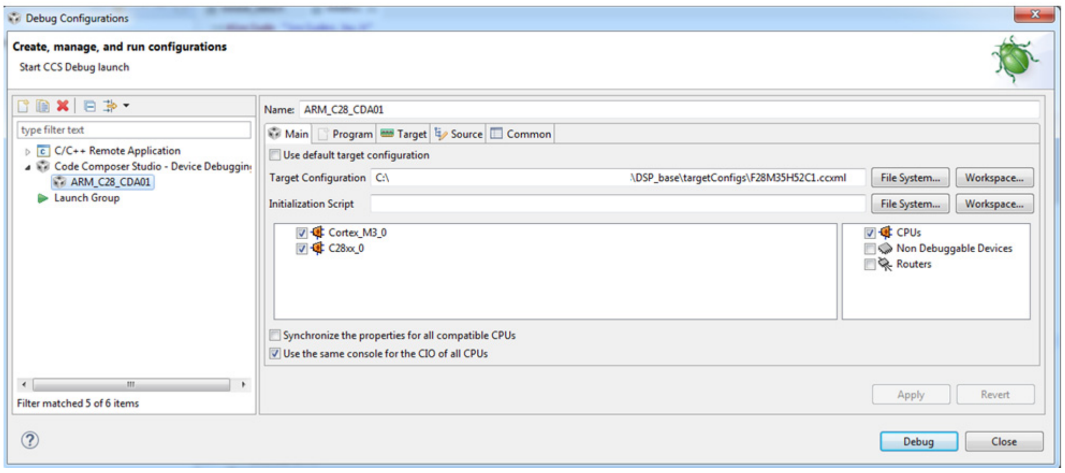
2. Debugging Concerto by CCS 6.1.0

This chapter explains how to configure CCS 6.1.0 to burn and debug both DSP and ARM.

1. Access to the debug configuration using the drop-down menu in the debug button and select *Debug Configurations…*



2. Under *Code Composer Studio – Device Debugging* view, delete all existing debug configurations. Afterwards, create a new debug configuration and rename it at the top of the window as ARM_C28_*ProjectName*. In *Main* view, select the Target Configuration included in your Project located at the Workspace. Both ARM Cortex_M3 and C28xx cores will appear.



Name: ARM_C28_CDA01

Use default target configuration

Target Configuration C:\ \DSP_base\targetConfigs\F28M35H52C1.ccxml

Initialization Script

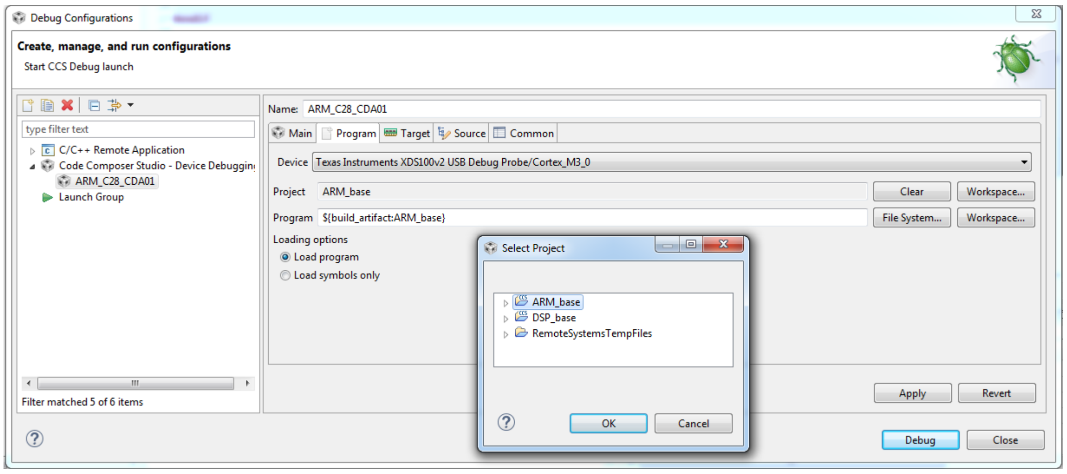
Cortex_M3_0

C28xx_0

Synchronize the properties for all compatible CPUs

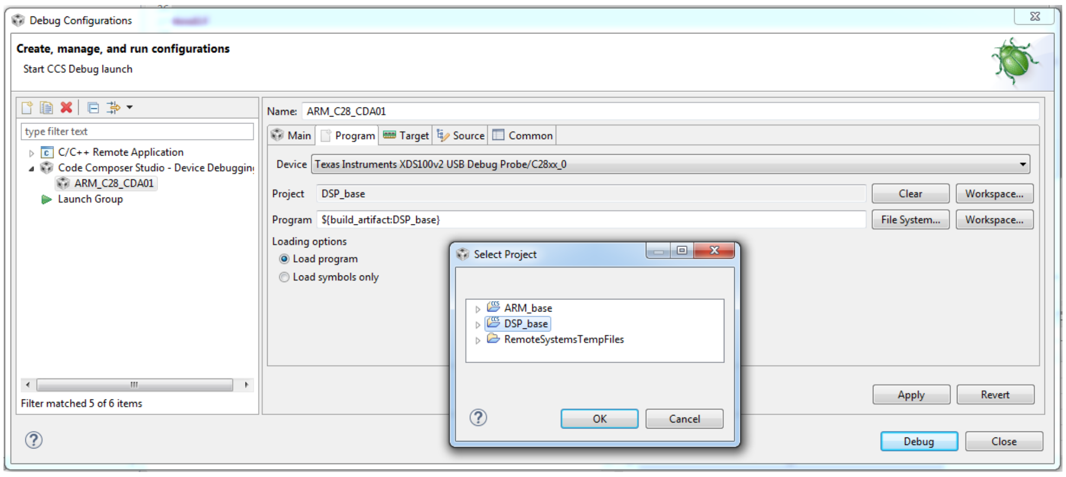
Use the same console for the CIO of all CPUs

3. Change view to *Program*. Select Cortex_M3_0 under *Device* drop-down menu. Click on *Project Workspace* button and select your ARM project folder.



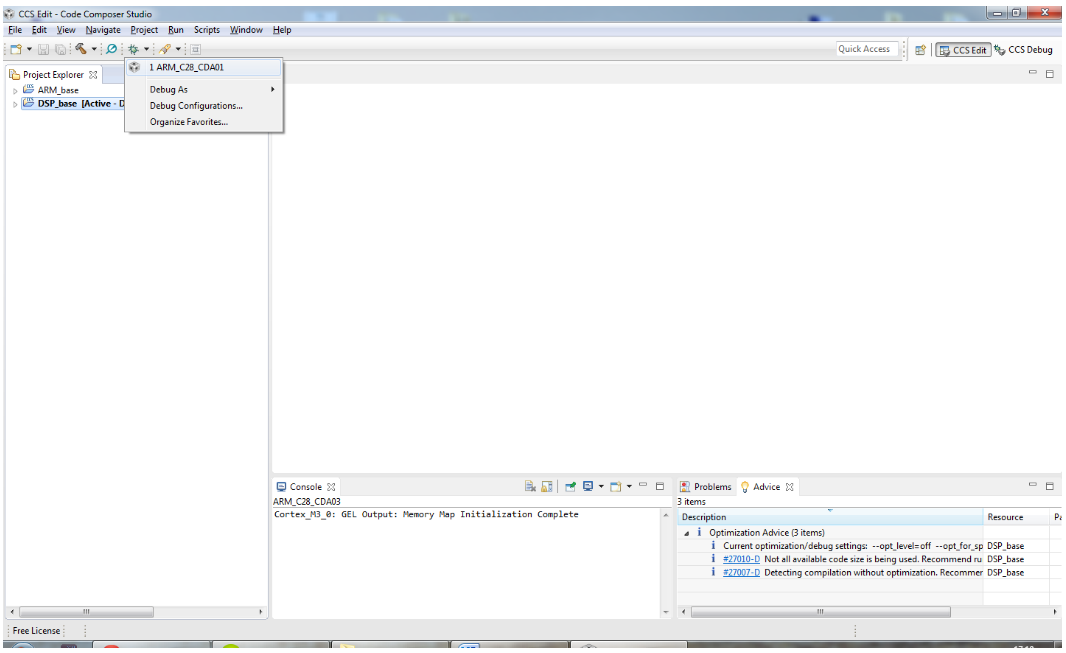
Name: ARM_C28_CDA01
Device: Texas Instruments XDS100v2 USB Debug Probe/Cortex_M3_0
Project: ARM_base
Program: ${build_artifact:ARM_base}
Loading options: (● Load program) ○ Load symbols only

4. Select C28xx_0 under *Device* drop-down menu. Click on *Project Workspace* button and select your C28 project folder.



Name: ARM_C28_CDA01
Device: Texas Instruments XDS100v2 USB Debug Probe/C28xx_0
Project: DSP_base
Program: ${build_artifact:DSP_base}
Loading options: (● Load program) ○ Load symbols only

5. Click on *Apply* button and *Close*. Debugging configuration is already finished.
6. From now on, to debug both cores you just have to select ARM_C28_*ProjectName* debug configuration under main debug drop-down menu.



```
ARM_C28_CDA03
Cortex_M3_0: GEL Output: Memory Map Initialization Complete
```

3 items

Description	Resource
Optimization Advice (3 items)	
Current optimization/debug settings: --opt_level=off --opt_for_sp	DSP_base
#27010-D Not all available code size is being used. Recommend ru	DSP_base
#27007-D Detecting compilation without optimization. Recommer	DSP_base

teknoCEA

www.teknocea.cat