APPLICATION NOTE

Keil uVision 5.X Support for the PAC52XX

Power Application Controller™

Marc D Sousa
Director, PAC Applications and Systems

Table of Contents

1 OVERVIEW

The Keil MDK (Micro-controller Development Kit) for ARM is a complete software development environment for ARM Cortex-M MCU-based devices. Some customers who want to use the PAC platform want to use the Keil tools for software development and debugging.

The uVision IDE (integrated development environment) contains a project manager, compiler, assembler and debugger that can be used to develop and debug application firmware.

This document specifies the installation and usage instructions for the PAC52XX family of products using the Keil uVision version 5.X IDE.

2 INSTALLATION INSTRUCTIONS

2.1 Install Keil uVision

The user should install the latest version of the Keil MDK-ARM Micro-controller Development Kit from ARM. The URL for this is: http://www.keil.com/arm/mdk.asp.

Contact ARM for licensing information for this tool.

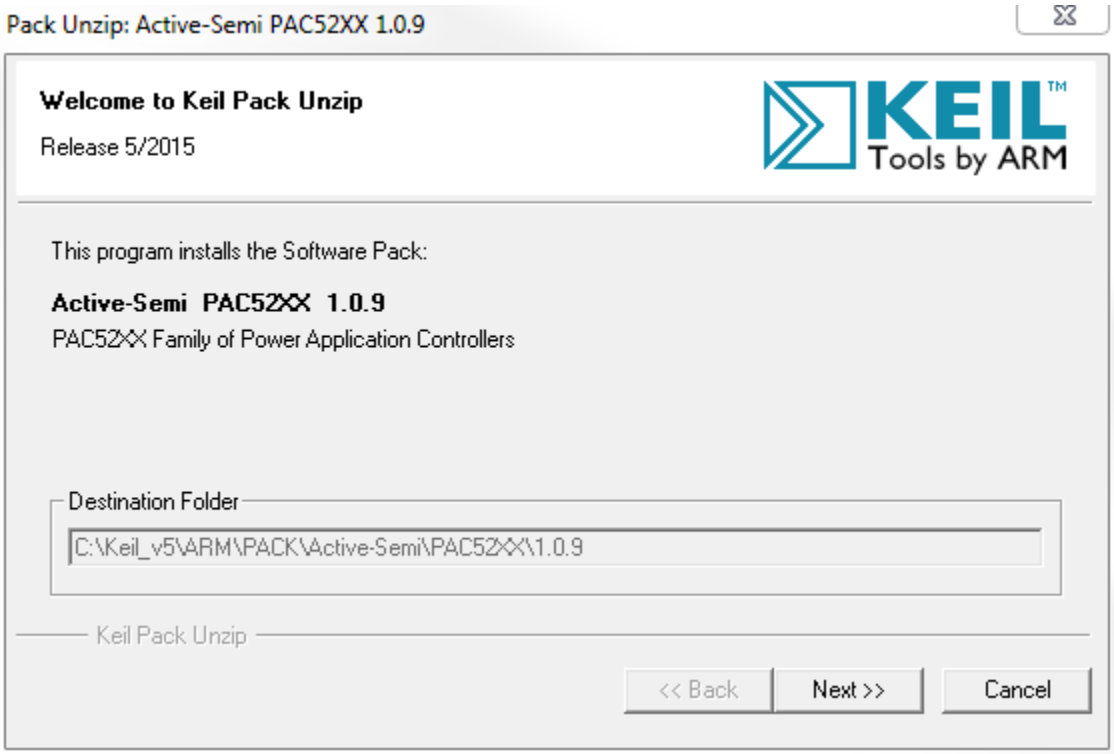
2.2 Install PAC52XX Support

The Keil uVision IDE supports various vendors' ICs by *software packs*. A software pack is a collection of software that defines the device for debugging and building programs using the IDE.

To support the PAC52XX, the user must install the PAC52XX software PACK provided by Active-Semi. This software PACK will have a filename like shown below:

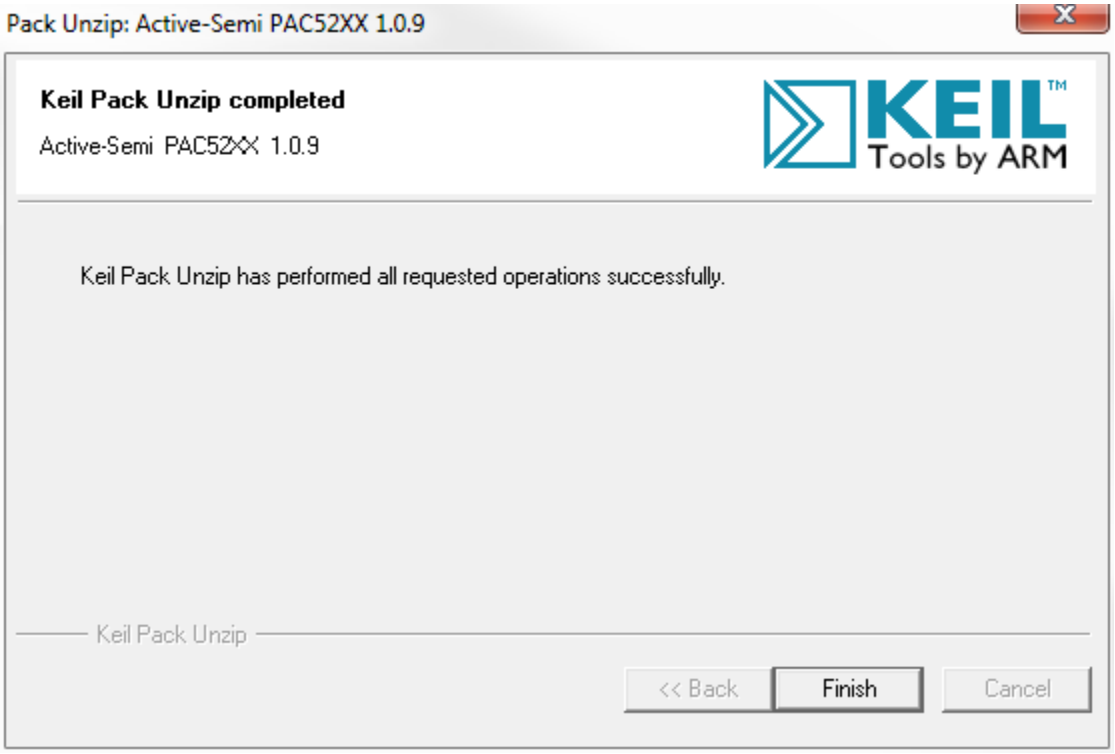
```
Active-Semi.PAC52XX.1.0.9.pack
```

This file is Windows self-extracting executable that will install the support for the PAC52XX family of devices when it is run. To install this PACK file, simply double-click the file. When you double-click this file, you will see a dialog box like shown below:



Press the "Next" button to install the PACK support for the PAC52XX.

If the installation was successful, you will see a dialog box similar to the one shown below:



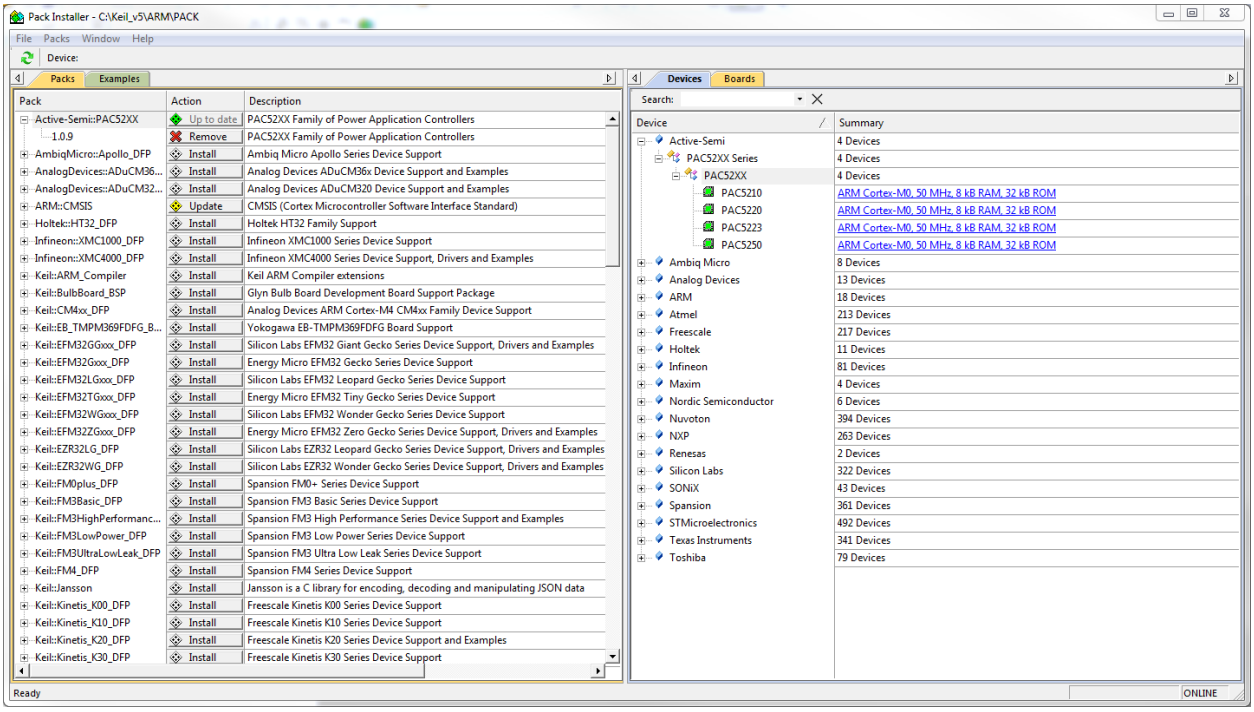
Before opening up the Keil uVision tool, you will need to add Active-Semi to the list of supported vendors for the Keil uVision IDE.

As system administrator, open up the file: `C:\Keil_v5\UV4\PACKJ.xsd`

Add the following line into this file under “registered device vendors”:

```
<xs:enumeration value=”Active-Semi:123”/>
```

You can verify that the software got installed properly by launching the Keil uVision IDE and opening the Pack Installer Tool (Project >> Pack Installer…). You should see the Active-Semi family of devices as shown below.



For more details, see the ARM application note on the Pack Installer here:

http://www.keil.com/support/man/docs/uv4/uv4_ca_packinstaller.htm

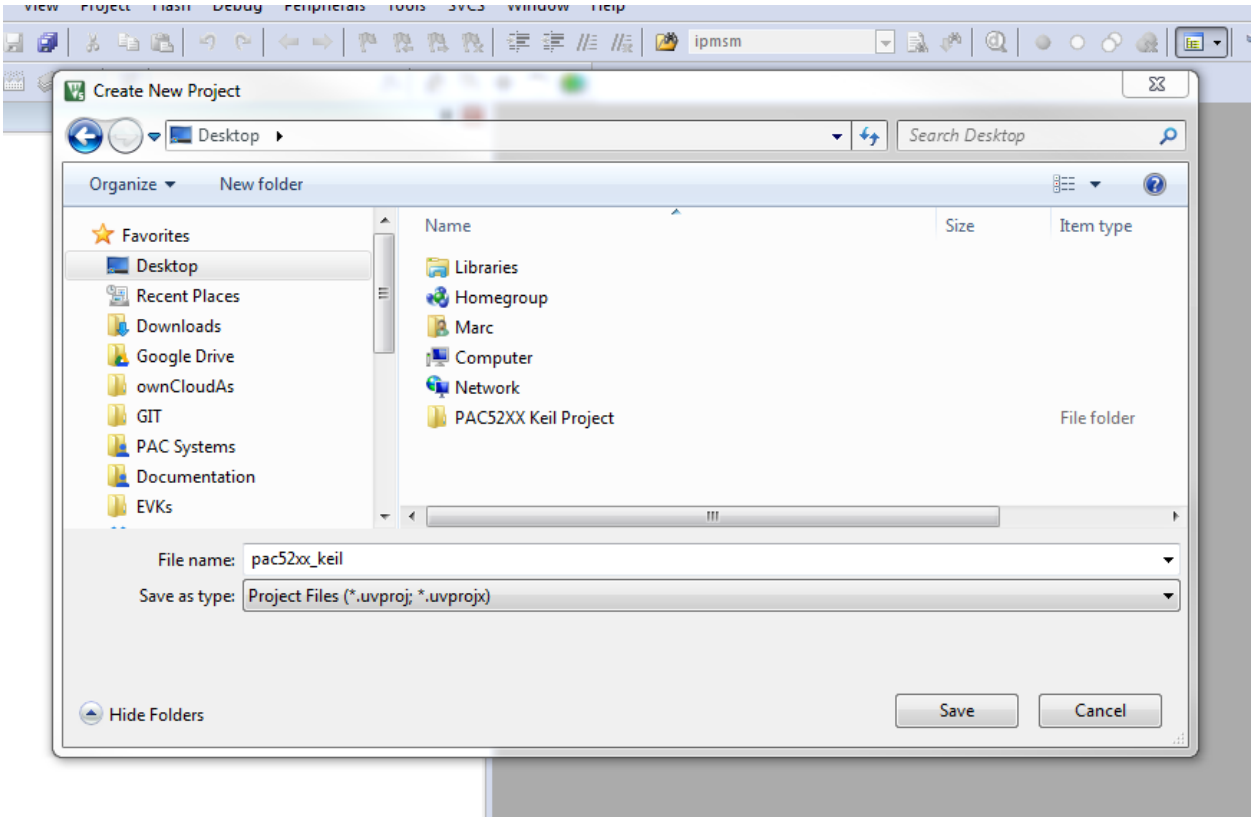
You are now ready to use the Keil uVision IDE to create projects for the PAC52XX.

3 CREATING A PROJECT FOR THE PAC52XX

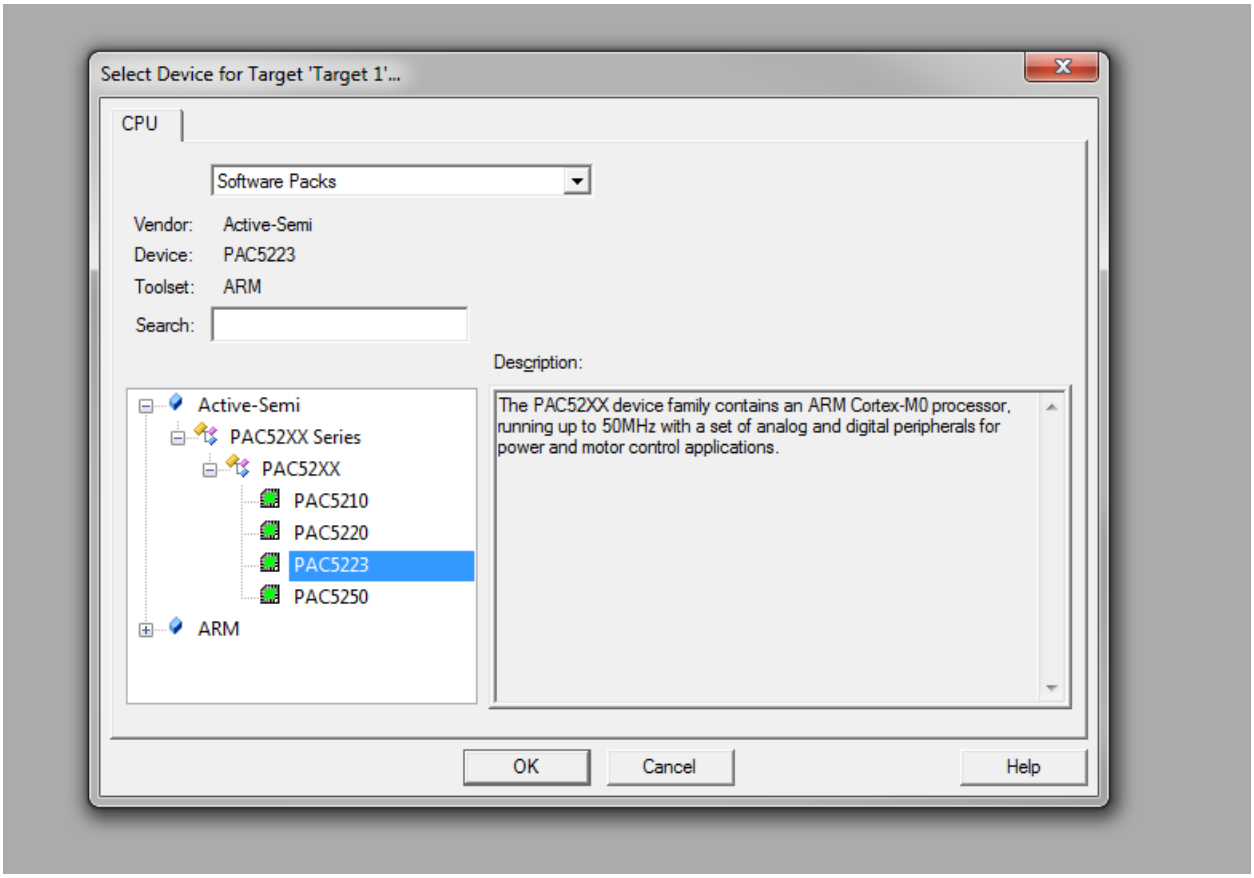
3.1 Create Project

To create a project for the Keil uVision IDE for the PAC52XX, follow the steps below.

First, create a new project by selecting ***Project >> Create new uVision Project***… from the main menu.

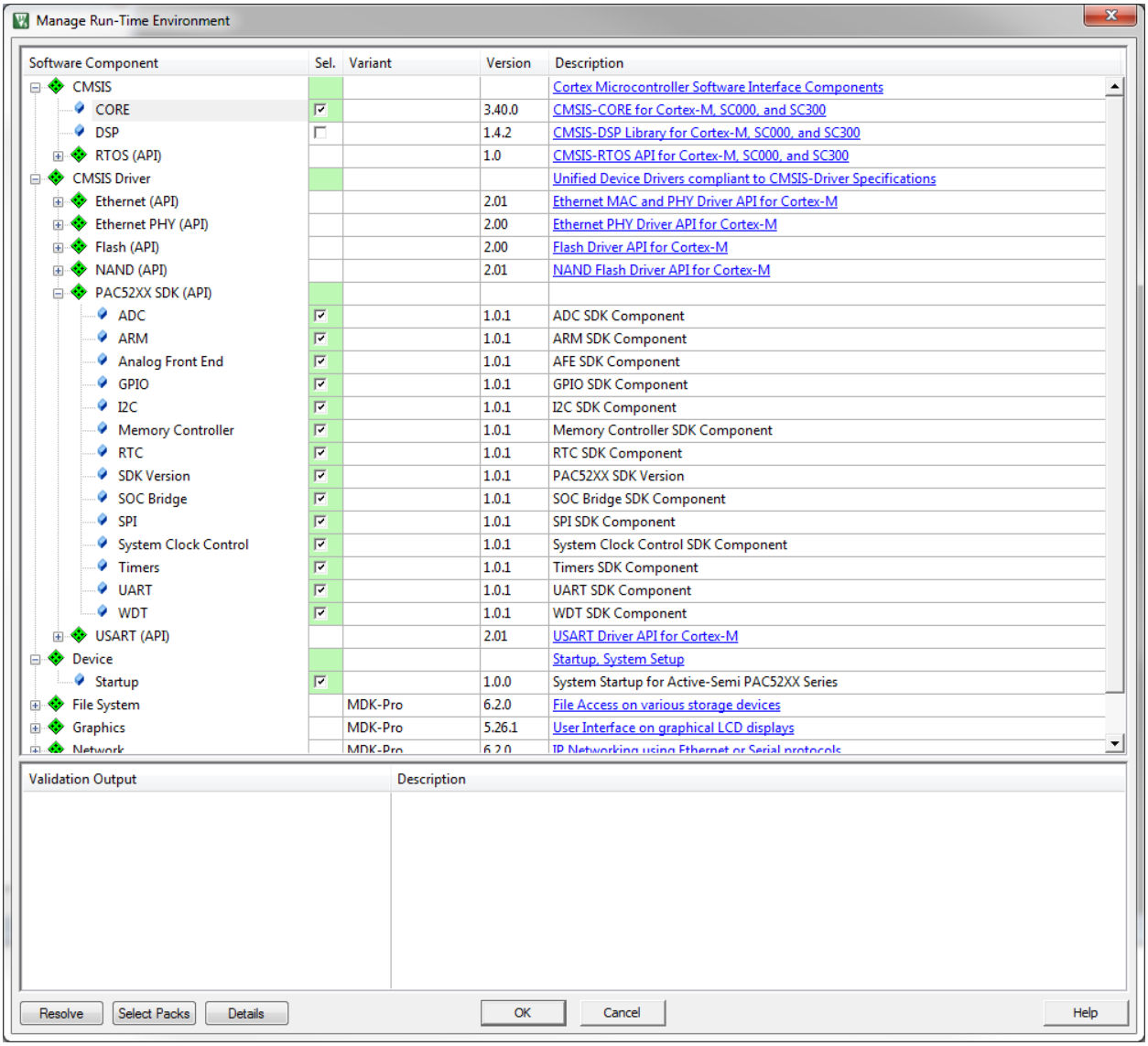


The IDE will ask which device for the project. Use this dialog box to select any of the PAC52XX devices. In the example below, I have selected the PAC5223.



Press OK when you have selected the device.

Next, the IDE will ask which software components to install. In order for this to work properly for the PAC52XX family of devices, be sure to select the following software components, as shown below:



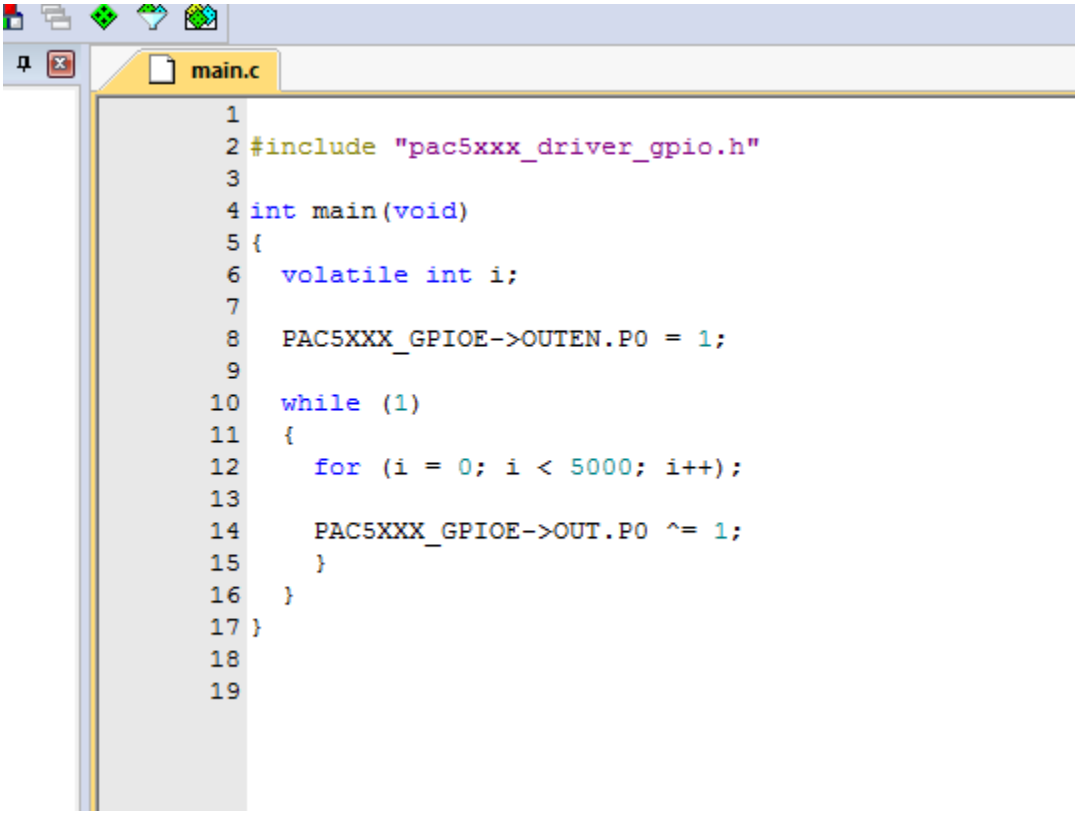
The CMSIS CORE component is required for all ARM Cortex-M0 device support.

All of the components under the CMSIS Driver >> PAC52XX SDK (API) are for the SDK that supports the PAC52XX programming. All should be selected.

Under Device >> Startup, the system startup files for the PAC52XX should be selected. This is where the interrupt vector table and other device specific startup will happen.

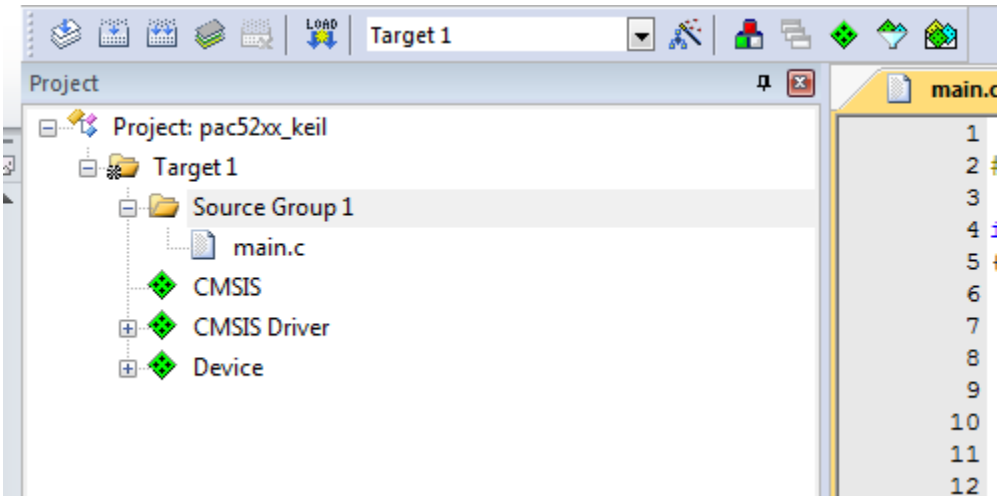
3.2 Add Source Files

Once the project is created, source files may be added to the project. Create a new source file by selecting ***File >> New…*** from the main menu. The file below shows a simple GPIO toggle program.

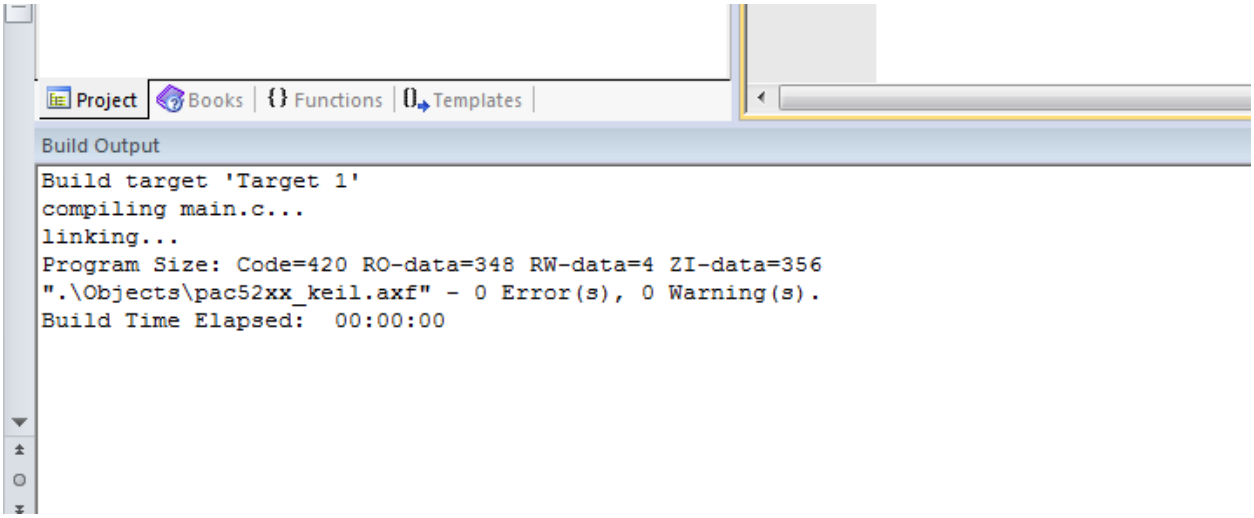


To add this file to the project, right-click on the “Target 1” group, and select “Add existing files to group ‘Source Group 1’”, and select this new file.

In this case, I have named this file “main.c”. Once you have added this file, you will see it under “Source Group 1” as shown below.



To build the project, select Project >> Build Target or press F7. You should see the following output in the ‘Build Output’ window showing a successful build of the application:



4 LINKING FUNCTIONS INTO RAM

For high-performance functions, it is sometimes useful to place them into RAM, instead of FLASH for faster execution.

Functions may be linked into RAM on a file basis. There are two ways to accomplish this:

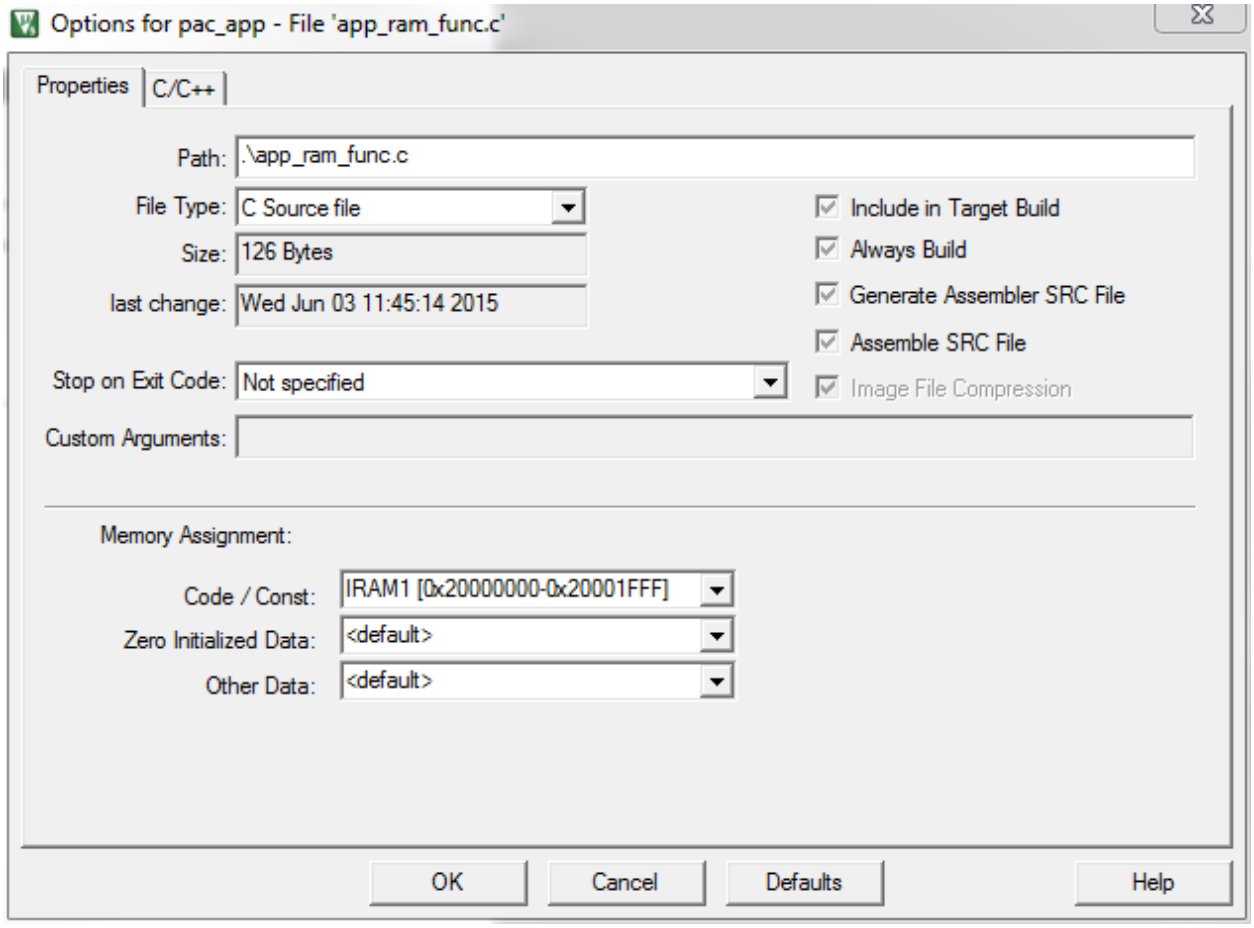
1. By modifying the file properties
2. By creating a custom scatter file

NOTE: If you are using a library and want to link a library function into RAM, you must use method 2: custom scatter file.

4.1 Modifying File Properties

To modify functions in a source file to be linked into RAM, you can modify the file properties.

Open the options for the file, by right-clicking on the file and selecting ***Options for <proj> - File <filename.c>***. You will see a dialog box as shown below:



At the bottom of this form, you can assign the memory segment for the code segment for this file. In the example above, we have selected the segment IRAM1 at address 0x20000000 (RAM) for the functions in this file.

Press OK and build your program. When you link the program, you can inspect the memory map and see that any functions in this file will now reside in RAM in the 0x20000000-0x20001FFF memory range.

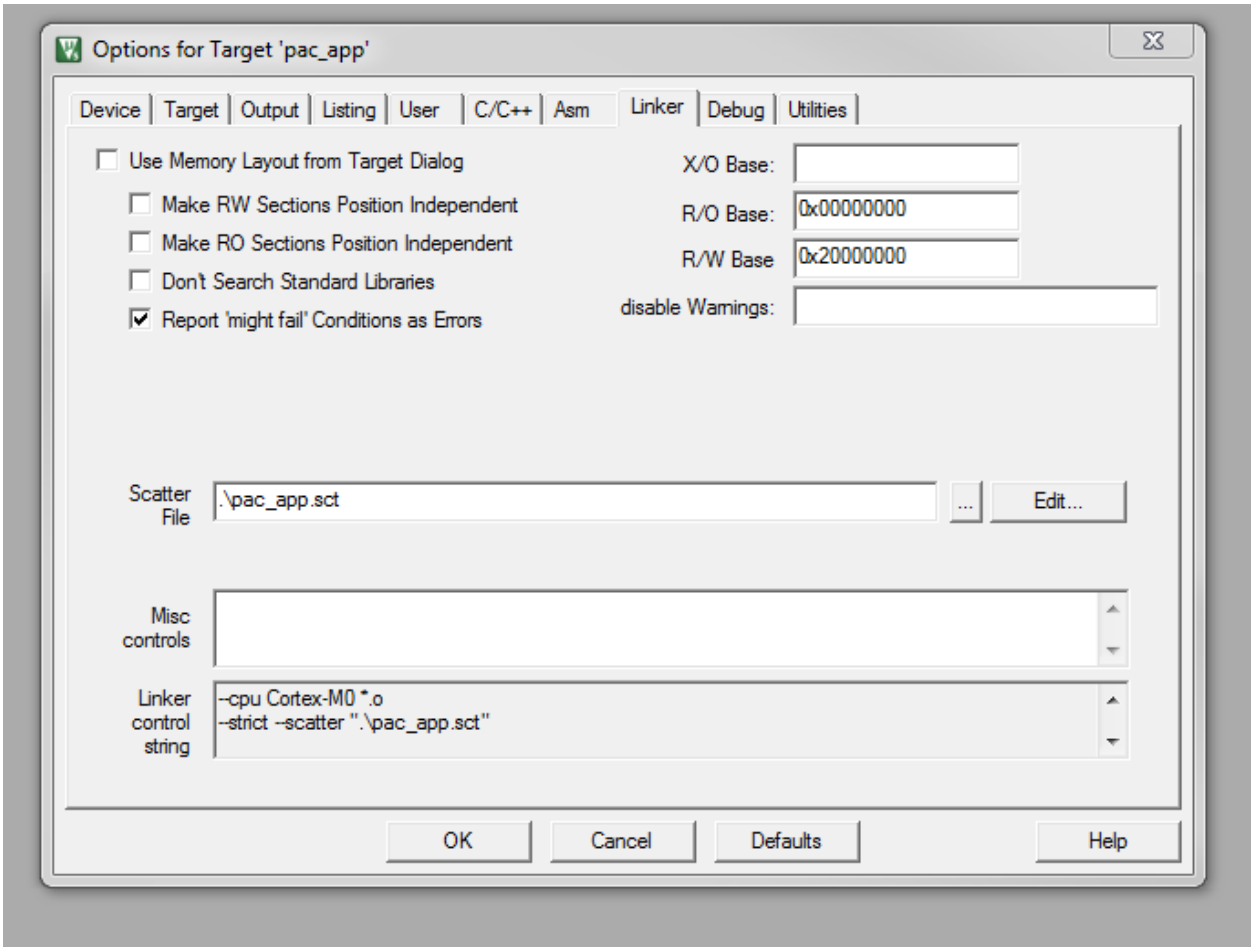
4.2 Custom Scatter Files

Having a customer scatter file is another way to link the desired functions into RAM. This technique must be used if the desired functions are in a library.

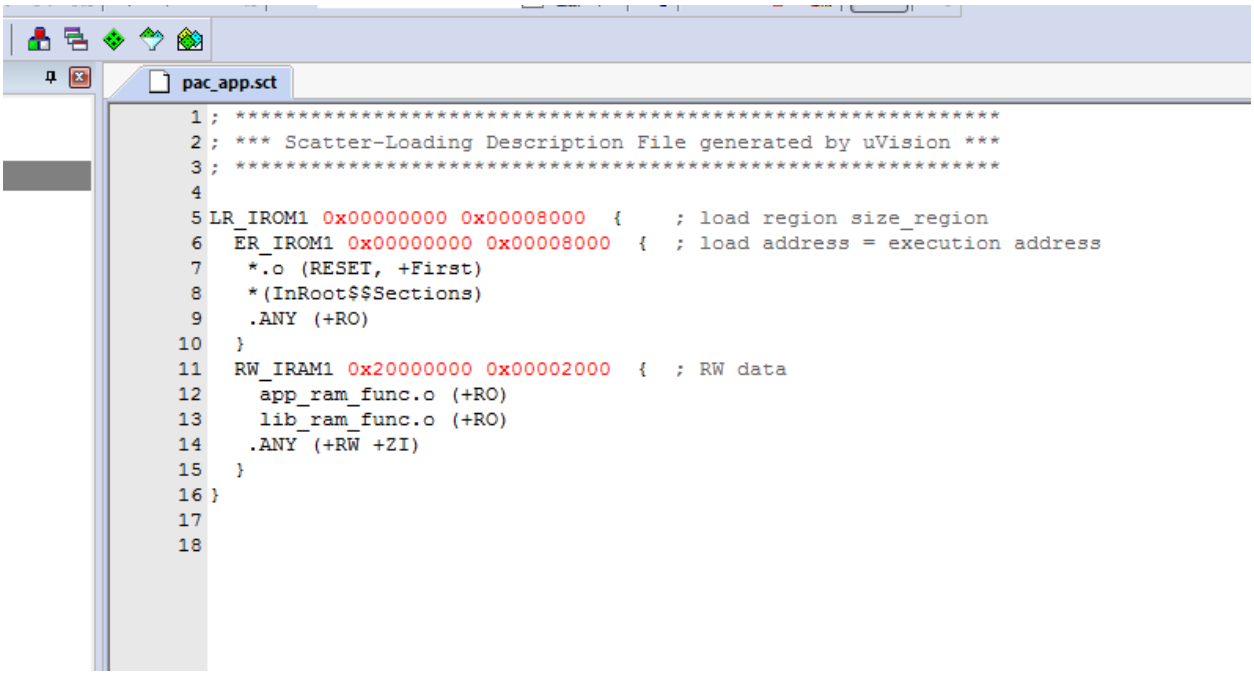
Note that if you use this technique, the technique described above (modifying the file options) will no longer work.

To create a custom scatter file, copy the existing scatter file to a location where it will not be overwritten (for example, where the project files are stored).

From the IDE, right-click the project and select ***Options for Target <proj name>*** and select the “linker” tab. You should see a form like the one shown below:



To select your scatter file, press the "…" button to the right of the "Scatter File" text box. To edit this file, press the "Edit…" button and then press "OK". You should see the scatter file in the editor window as shown below:



In the RW_IRAM1 segment, you need to list out all the object files that you want linked into the RAM section.

In the example above, I have four files. Two with functions that go into FLASH and two with functions that go into RAM. The two files that get linked into RAM are as follows (these are added into the scatter file above).

- app_ram_func.o – Functions for RAM in the main application
- lib_ram_func.o – Functions for RAM in a library

Make all the changes to the scatter file, then save the file and re-build the application. You will now see the desired functions in RAM in the memory map.

4.3 Validating the Memory Map

To validate that the correct functions got placed into RAM, you should inspect the memory map.

Open the memory map in a text editor and you should see something like the following:

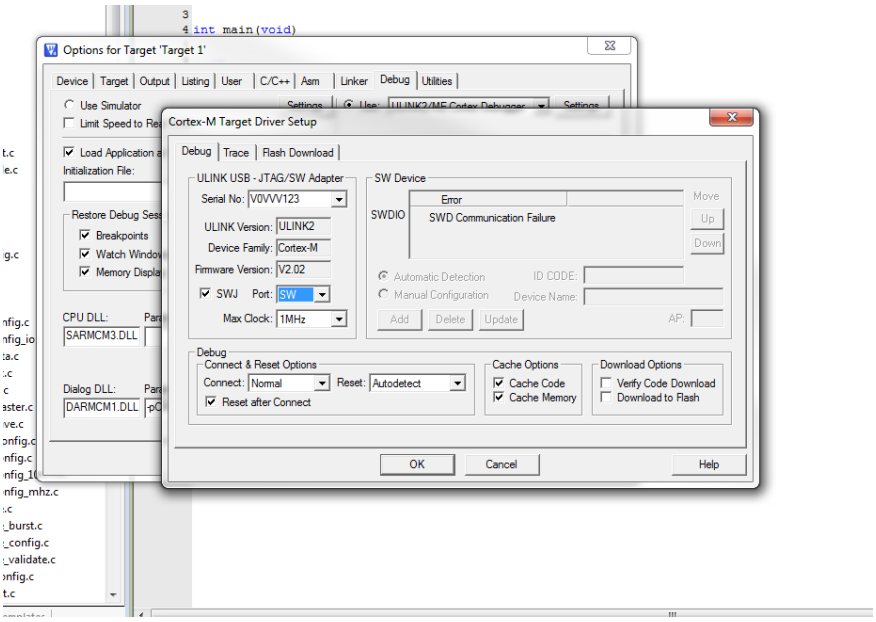
```
3775    __semihosting_library_function          0x000001d7   Thumb Code     0  indicate_
3776    Long Thumb to Thumb Veneer to app_ram_func 0x000001d9   Thumb Code    12  anon$$c
3777    Long Thumb to Thumb Veneer to lib_ram_func 0x000001e5   Thumb Code    12  anon$$c
3778    SystemInit                              0x000001f1   Thumb Code     8  system_pa
3779    app_flash_func                          0x00000201   Thumb Code    12  app_flash
3780    lib_flash_func                          0x00000211   Thumb Code    10  lib_flash
3781    main                                    0x00000221   Thumb Code    22  main.o(i.
3782    Region$$Table$$Base                     0x00000238   Number         0  anon$$obj
3783    Region$$Table$$Limit                    0x00000268   Number         0  anon$$obj
3784    app_ram_func                            0x20000001   Thumb Code    12  app_ram_f
3785    lib_ram_func                            0x20000011   Thumb Code    10  lib_ram_f
3786    app_flash_func_data                     0x20000020   Data           4  app_flash
3787    app_ram_func_data                       0x20000024   Data           4  app_ram_f
```

You can see that the two functions to be linked into RAM (app_ram_func and lib_ram_func) are correctly placed in the RAM memory by the function address (0x200000XX).

5 DEBUGGING WITH THE PAC52XX

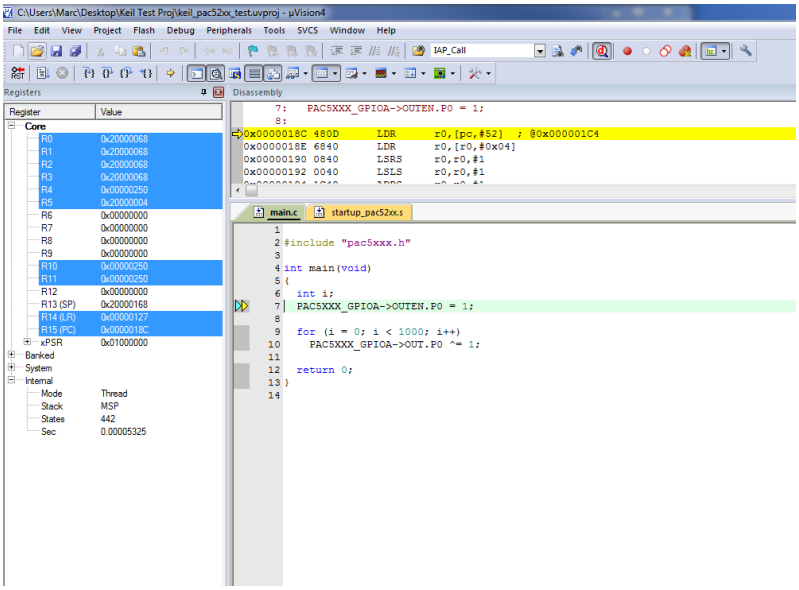
To configure the Keil uVision IDE for debugging, the user must first select a debugger emulator to use. For this example, we are using the Keil Ulink2 adapter.

To configure the adapter, select the Project Options and then select the Keil ULINK2/ME Cortex Debugger selection on the right and press the *Settings* button. You should see a form like shown below.



Make sure you select the “SW” for the debug port and select the correct FLASH downloader on the *Flash Download* tab.

To debug your application, Select *Debug* and then *Start/Stop Debug Session* or press CTRL+F5 to start the debugger. You should then see the debugger connect to the target and stop at the main entry point as shown below.



ABOUT ACTIVE-SEMI

Active-Semi, Inc. headquartered in Dallas, TX is a leading innovative semiconductor company with proven power management, analog and mixed-signal products for end-applications that require power conversion (AC/DC, DC/DC, DC/AC, PFC, etc.), motor drivers and control and LED drivers and control along with ARM microcontroller for system development.

Active-Semi's latest family of Power Application Controller (PAC)™ ICs offer high-level of integration with 32-bit ARM Cortex M0, along with configurable power management peripherals, configurable analog front-end with high-precision, high-speed data converters, single-ended and differential PGAs, integrated low-voltage and high-voltage gate drives. PAC IC offers unprecedented flexibility and ease in the systems design of various end-applications such as Wireless Power Transmitters, Motor drives, UPS, Solar Inverters and LED lighting, etc. that require a microcontroller, power conversion, analog sensing, high-voltage gate drives, open-drain outputs, analog & digital general purpose IO, as well as support for wired and wireless communication. More information and samples can be obtained from http://www.active-semi.com or by emailing marketing@active-semi.com

Active-Semi shipped its 1 Billionth IC in 2012, and has over 120 in patents awarded and pending approval.