AT91 Library

Scope

This document describes the contents of the AT91 Library V2.0, explains its uses and defines the naming and coding rules used in the source files. It applies to version 2.0.

The section “Getting Started with the AT91 Library” on page 4 provides installation instructions to get up and running quickly.

What is the AT91 Library?

The AT91 Library is a set of C and assembly source code modules and project files that enable AT91 developers to quickly and easily define the architecture and the peripherals of their intended application, using any of the supported toolkits.

This AT91 Library software package supports all the AT91 ARM Thumb microcontrollers and the associated evaluation boards.

The AT91 Library V2.0 contains:

- Header files in C that define the AT91 peripherals and parts
- Assembly Include files that define the AT91 peripheral and part assembly language
- Examples of how to access the AT91 peripherals with or without interrupt handling
- C start-up files that explain how to boot an AT91 part, and how to branch on the main C function
- Project examples that show different aspects of the debug features, such as semi-hosting
- Target tools with which to evaluate the AT91 parts

The AT91 Library V2.0 also provides one software library for each of the supported parts and one common software library for the complete set of drivers. All projects and tools use these libraries by mounting a part library and, if required, the driver library.

Definition of Terms

- **Device** is used to indicate devices used by the operating systems, such as Flash memory.
- **Part (or parts)** is used for any device of the AT91 ARM Thumb family (AT91M40800, AT91M63200, etc.). It is used instead of “device” to prevent any confusion with the devices used by the operating systems.
- **Periph** is used to mean peripherals.
- **Target (or targets)** refers to any evaluation board or any customer-specific board fitting an AT91 ARM Thumb microcontroller.

AT91
ARM® Thumb® Microcontrollers

Application Note

Rev. 1385A–11/00

Warnings

Backward Compatibility

The AT91 Library V2.0 is not compatible with the previous version, which was provided as an example for developing and running code for the AT91M40400 only.

Warranty

All delivered sources are free of charge and can be copied or modified without authorization.

The software is delivered “as is”, without warranty or condition of any kind, either express, implied or statutory. This includes, without limitation, any warranty or condition with respect to merchantability or fitness for any particular purpose, or against the infringements of intellectual property rights of others.

AT91 Library Rules

These rules are used to improve readability and to obtain information on the library tools.

File Naming and Extensions

File names are written in lowercase to ensure maximum compatibility with all operating systems that the development tools may run on.

Table 1. File Names and Extensions

File Type	File Format	Comments
Header files	`<periph_name>.h` `<driver_name>.h`	For example, the USART header file is `usart.h` and the wait driver interface header file is `wait.h`.
Assembly Include files	`<periph_name>`.inc	For example, the assembly include file for the USART is `usart.inc`.
C Library files	`lib_<periph_name>.c`	Contains the library functions
Prototyping Header files	For C file `<c_file>.c`, prototyping header file is `<c_file>.h`	Each C file has a C header file that prototypes the functions of the C file. This header file also defines the descriptor of the drivers.
Assembly Source files	`.s` extension	Assembly source files are named differently from C files in order to prevent the creation of an object file name with extension `.o`, which negates both sets of files. Assembly coding is required for interrupt handling (at least interrupt handler entry and exit).
Assembly Macro files	`.mac` extension	The assembly macro file names have the extension `.mac` (and not `.s` as generally used in the ARM examples).

Coding Rules

Constants

All assembly or C constants are defined in uppercase.

The constant names have the format <periph_short>_<field>. <periph_short> is the name used in the datasheet (US for the USART, AIC for the AIC, etc.). <field> refers to the name of the register fields as described in the datasheet.

Peripheral Structure Definitions

Each AT91 peripheral is described by a C structure. This structure is composed of registers and a corresponding list of offsets.

This structure enables a modular architecture that can be used with any peripheral. The peripheral base address identifies the peripheral.

File Inclusions

All AT91 Library header files are protected against multiple inclusion – each file can be defined only once. Each Library header file can include another header file.

Function name

All functions of the AT91 Library have one of the following names:

- at91_<module>_open
- at91_<module>_close
- at91_<module>_get_status
- at91_<module>_trig_cmd
- at91_<module>_read
- at91_<module>_write
- at91_<module>_set_mode

Each of these names corresponds to a logical operation to be performed on the peripheral or on the driver. For example, the function at91_usart_open enables the clock on the peripheral, validates the I/O lines and programs the main configuration register of the peripheral.

Comments

All C and assembly constants are commented.

The comments are placed before the code lines and use the same indentation.

Getting Started with the AT91 Library

Installation

To install the AT91 Library V2.0, copy the complete hierarchy onto the hard disk of the computer, using the following path:

`C:\at91\software`

All projects provided use this path. If you change the path, you must define the new path before building the projects.

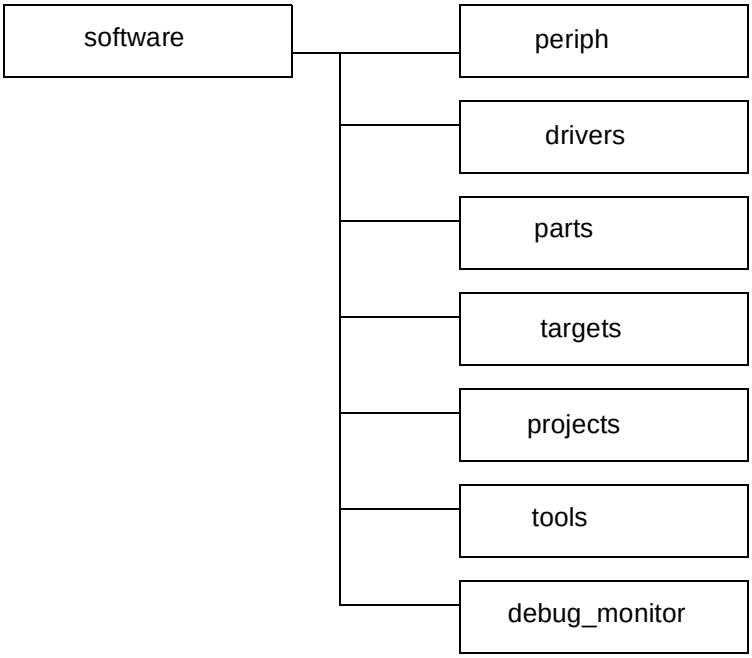
Building the Libraries

Library object files are delivered already built. However, if you want to modify the sources and rebuild the libraries, all the files are provided and you can do so by running one of the following DOS commands:

- To build the library for the part <part>:
  ```
  make -f c:\at91\parts\<parts>\makefile
  ```
- To build the driver library:
  ```
  make -f c:\at91\drivers\lib_drv\makefile
  ```

AT91 Library Source Hierarchy

Figure 1. AT91 Library Directory

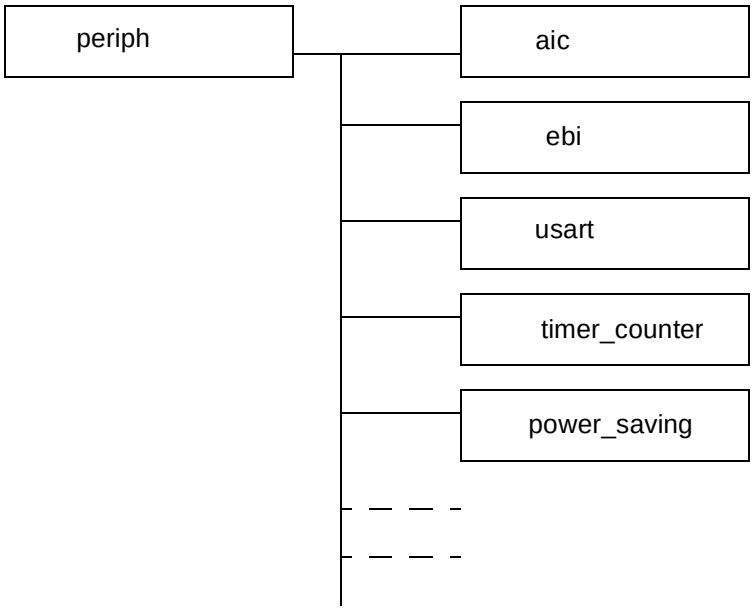


Peripheral Directories

The directory `/software/periph` contains a directory for each of the AT91 on-chip peripherals.

These directories contain C and assembly source code files that provide examples of how to use the AT91 peripherals. The peripheral access functions are written in C. They are associated with an assembly file if the peripheral needs interrupt handling.

Figure 2. AT91 Peripheral Hierarchy



Each peripheral directory contains at least:

- A peripheral header file (extension .h) that describes the peripheral structure and, when necessary, the descriptor structure of the peripheral
- A peripheral assembly Include file (extension .inc)
- A peripheral C access file (lib_<peripheral>.c) that works on a peripheral descriptor structure when it exists or directly on the peripheral from its base address
- A peripheral C header that prototypes functions of the C access file

The Power-saving Directory

The subdirectory \software\periph\power_saving contains description files of all the different power-saving modules, or power management controllers integrated in the AT91 microcontroller.

It also contains a C access peripheral file and a corresponding C header file (lib_power_save.h).

The C Standard Directory

The directory \software\periph\stdc contains the C definitions used as a standard by all the components of the AT91 Library.

It includes the file std_c.h, which defines the AT91 peripheral registers and the TRUE and FALSE labels.

It also includes the files lib_err.c and lib_err.h, which enable management of errors inside the AT91 Library.

Parts Directory

The directory \software\parts contains a folder for each part supported by the AT91 Library. This folder uses the part name, minus the AT91 prefix, in lowercase. For example, the AT91M40800 is described in the folder \software\parts\m40800.

Each part is described as follows:

- The file <part>.h includes all the peripheral definition files of the part, providing a shortcut to loading the core and each relevant peripheral. This file also defines the base addresses of the on-chip user peripherals and memories.
- The file reg_<part>.h defines a complete list of the on-chip peripheral registers, making these registers easily accessible.
- The file lib_<part>.h includes all the C peripheral access header files that define the functions (defined in lib_<part>.c) and prototyping for the part. It also defines the external reference for the on-chip peripheral descriptors.
- The file lib_<part>.c defines the peripheral descriptor of the part.

Driver Directory

The AT91 Library drivers are functional objects described in the directory software\drivers. Each driver has its own directory, which includes:

- The file <driver>.h, which defines the driver descriptor, the data driver descriptor (if necessary) and the external references to the driver functions
- The file <driver>.c, which defines the driver functions
- The file irq_<driver>.mac, which is an assembly macro file that enables the driver user layer to easily define a driver interrupt handler

Depending on the complexity of the driver, other C access and C header files may also be included.

The driver library is created in the directory lib_drv, which includes subdirectories for each compiler variant.

To use a driver library, you have to link it to the file lib_drv_16.alf in the appropriate variant directory.

Target Directory

Each AT91 evaluation board is described inside a subdirectory of the directory \software\targets.

Each of these subdirectories contains the following files:

- The <target>.h file, which defines the components of the boards in C
- The <target>.inc file, which defines the components of the boards in assembly
- One or more cstartup.s files, which define a standard boot for the boards

C Start-up Files

The C start-up files provide examples of how to boot with an AT91 part, taking into account the part-specific features, the board-specific characteristics and the debug level required.

A final production boot that works with Angel and enables semi-hosting requires a different initialization sequence to the production boot, which uses an ICE interface.

When working with Angel, the boot sequence must not re-initialize the interrupt controller (used by Angel to communicate with the debugger).

Project Directory

The directory software\projects contains project files that include examples of programming and show how to use the AT91 Library using direct register access and access using the peripheral and driver libraries. Each project is in a separate subdirectory, which contains all the relevant C assembler and project files.

These project files are provided for building the part library. Each project file has a different optimization variant (for example, Debug mode, Flash mode) in order to match build needs.

Table 1. Project Folders

Project Directory	Contents
`software/projects/boot_eb01`	Projects programmed in EB01 boot Flash devices
`software/projects/boot_eb40`	Projects programmed in EB40 boot Flash devices
`software/projects/boot_eb63`	Projects programmed in EB63 boot Flash devices

Tools Directory

The AT91 Library directory `\software\tools` contains tools that run on a target.

The directory includes:

- The Flash downloaders for the Flash devices fitted on the AT91 evaluation boards
- Power consumption measurement utilities that enable AT91 devices to operate under different conditions (Idle mode, running out of internal or external memories, memories programmed with or without wait states)
- The Dhrystone program, which enables evaluation of a complete development chain and device environment
- The DataFlash®
- A large image generator

The sources and the project files are provided for each tool.

Angel Debug Monitor Directory

This directory contains the Angel Debug Monitor sources and the project files so you can use the Angel Debug Monitor on your own board.

The Angel Debug Monitor is programmed in the program nonvolatile memory of the AT91 Evaluation Board.

The AT91 Application Group can offer only limited support on the Angel sources, as the Angel development and port on evaluation board were not developed by Atmel.

Angel Image Files

The Angel binary image files to be programmed on the AT91 evaluation boards are:

- `software\debug_monitor\angel\image\sram_eb01\angel_at91.rom` for the EB01
- `software\debug_monitor\angel\image\sram_eb40\angel_at91.rom` for the EB40
- `software\debug_monitor\angel\image\sram_eb63\angel_at91.rom` for the EB63

The corresponding ELF image files have the same name with the extension `.axf`. The ELF image files help to debug the start-up sequence by downloading it to a target. They can also be used to download symbols when debugging Angel from nonvolatile memory.

Rebuild Angel for an AT91 Evaluation Board

To rebuild Angel for one of the AT91 evaluation boards:

1. Open the project file `software\debug_monitor\angel\image\angel_at91.apj` with the ARM Software Development Toolkit V2.5 (or older).
2. Click on the required variant (sram_eb01, sram_eb40, sram_eb63, etc.) and select Force Build from the Project menu.

Using the AT91 Library

Direct Register Accesses

All the registers of a part can be easily accessed using the following C code:

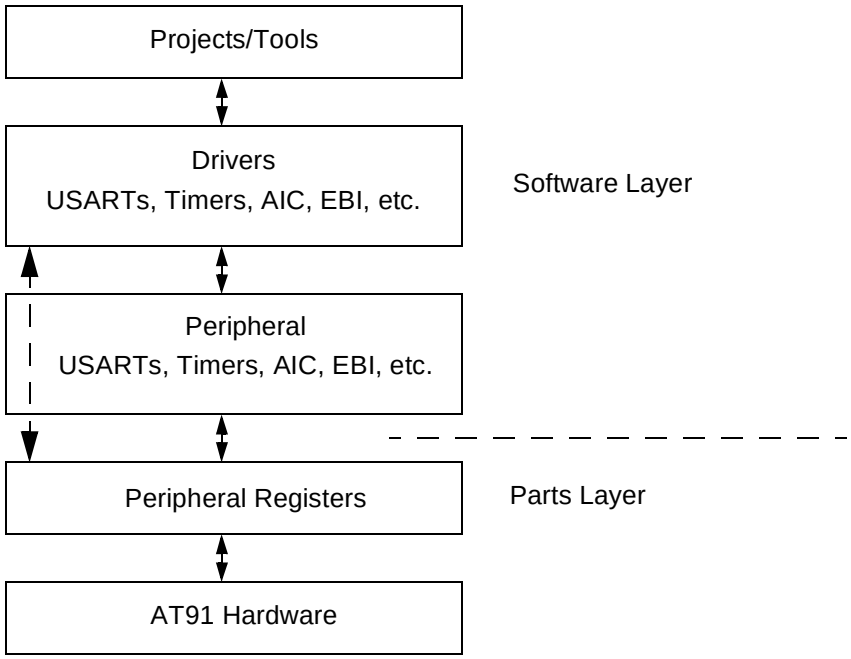
```
#include“part\<part_name>\reg_<part_name>.h
...
status = US0_SR ; \* read the USART 0 Status Register */
US1_MR = 0 ; \* Clear the USART 1 Mode Register */
```

If the register or the peripheral does not exist, an error is provided when compiling.

The syntax for the register access is the one defined in the relevant AT91 datasheet. If the part uses more than one peripheral, the peripheral index is added after the peripheral abbreviation.

Part and Driver Software Layers

Figure 3. AT91 Library Software Layers



Dynamic Referencing Considerations

In order to increase the reuse of any code provided in the AT91 Library, dynamic references are used for the peripherals, the drivers and their data spaces and for the interrupt handling definitions.

Peripheral Descriptors

A peripheral descriptor contains a data structure and specific constants that define the characteristics of the peripheral. For example, a USART descriptor defines the USART base address, the peripheral identifier (i.e., its interrupt number in the AIC and its clock number in the power-saving module), and the PIO Controller descriptor (to which its I/O lines and the number of these I/O lines are connected).

Only user peripherals require a descriptor. However, because the PIO Controller has a system role (assigning I/O lines to a peripheral) and a user role (the direct control of the I/O lines), it also needs a peripheral descriptor.

The AT91 Library functions that operate on a user peripheral accept, as a first argument, a peripheral descriptor pointer.

Driver and Data Driver Descriptors

Each driver uses a driver descriptor that includes the peripheral descriptor pointer or pointers that the driver must use. For example, the serial driver descriptor's first parameter is the USART descriptor pointer that the driver has to use. When necessary, a data driver descriptor is also defined for the drivers requiring a data space be defined. The second parameter of a driver descriptor is the pointer on a data driver descriptor.

Interrupt Handler Macros

As all the library components can be defined dynamically, the peripheral handler must be defined by the highest software layer. To make this definition easier, each driver that requires interrupt handling defines a macro named (in uppercase): AT91_<DRIVER>_ASM_HANDLER.

To create a handler, an assembly file (`.s`) must be defined that includes the driver macro file.

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-803-000
FAX (44) 1355-242-743

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1385A-11/00/0M