AVR32000: Introduction to AVR32 header files

Features

- **Register and Bit-Name Definitions**
- **Use of Bit-field and Bit-mask**
- **Use of type definitions**
- **Use of Macros**
- **Deviance Between Hardware Registers and Header Register Naming**

1 Introduction

The purpose of this application note is to give new users a basic introduction to the header files for AVR®32 microcontrollers. The usage of I/O registers, bit-names and module type definitions. It will also cover more advanced usage of the header files like the I/O modules structures. This application note is specific for IAR Systems® AVR32 compiler and GNU GCC for AVR32 compiler.

32-bit AVR® Microcontrollers

Rev. 32005A-AVR32-05/06

2 Register and Bit-Name Definitions

The header files for AVR32 are split into many small files. One header file defines the core and one header file defines each module. This eases the portability of the code to other devices because the source code does not have to know which version of a module is used in a specific device, the only necessary include is the unique device header file.

All code, which includes a device module, must start by including the io header file for the AVR32 devices. The io header file for devices uses the following convention:

```
avr32/io.h
```

The io.h header file knows the target device by a flag passed to the compiler.

Within the io.h file the devices and their specific header file are included, which includes and defines all modules available for the target device. This eases the users need to know what the include file for a specific module is named, since all defines for a module are present by including a single general header file.

Registers are named as they are given in the datasheet, and extended with a defined prefix. Naming of registers use the following convention:

```
AVR32_<module name>_<register name>
```

Example for USART control register:

```
AVR32_USART_CR
```

Bit-field names are named as they are given in the datasheet including a prefix. Naming of bit-fields use the following convention:

```
AVR32_<module name>_<register name>_<bit-field name>
```

Example for USART transceiver enable in the control register:

```
AVR32_USART_CR_TXEN
```

The offset and mask of a bit-field name in a register is also available with a more explained define. Naming of bit-field offsets and masks use the following convention:

```
AVR32_<module name>_<register name>_<bit-field name>_OFFSET
AVR32_<module name>_<register name>_<bit-field name>_MASK
```

Example for USART transceiver enable in the control register:

```
AVR32_USART_CR_TXEN_OFFSET
AVR32_USART_CR_TXEN_MASK
```

For reducing code text size it is possible to use abbreviated bit-field names. If the bit-field name is unique and all values with that name are the same for all registers, the register name is dropped in the definition of the bit-field name.

Example for USART transceiver enable bit in the control register can be written:

```
AVR32_USART_CR_TXEN
```

Or

```
AVR32_USART_TXEN
```

3 Use of Header Files

3.1 Use of Bit-field and Bit-mask

Registers are available by using type definitions, typedef, or by direct access. All registers can be defined as pointers to a memory address, and are accessible by dereferencing the pointer.

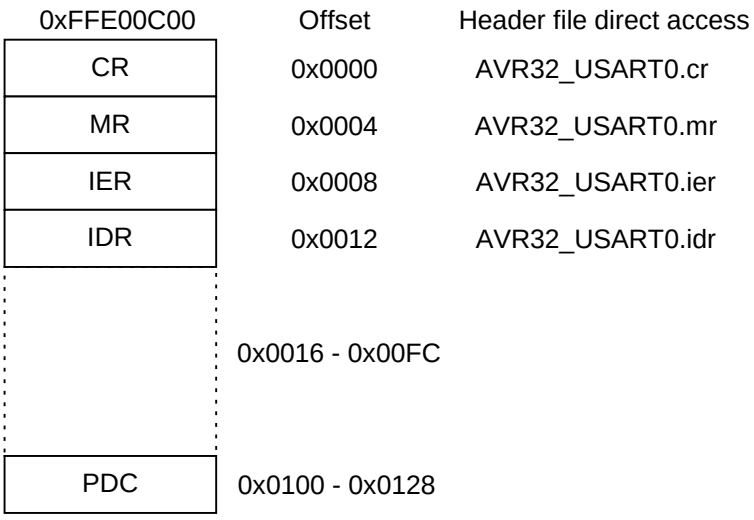
Registers are defined as an offset to the base address, for simplifying access to the registers of each module instance (see Figure 3-1). Naming of the pointer to the base address use the following convention:

```
AVR32_<module name><module instance starting at 0>
```

Example for USART module instance A pointer to base address:

```
AVR32_USART0
```

Figure 3-1. Memory mapping to registers for the AP7000 USART0 module to header files.



3.1.1 Use of type definitions

All modules have a type definition, in C and C++ known as typedef. These can be used to access the I/O memory concerning this module.

All type definitions are a volatile pointer that consists of one or several structures. Naming of type definitions use the following convention:

```
volatile avr32_<module name>_t *
```

Example using type definitions for accessing the USART0:

```
volatile avr32_usart_t * myUsart = &AVR32_USART0;
myUsart->mr = (1<<AVR32_USART_TXEN_OFFSET);
```

3.1.2 Use of pointers

The bit-fields and bit-masks are accessed by writing values to registers defined in the module structure, which is defined in the header file for the module.

Example using a pointer to USART0 struct for accessing MR in the USART0:

```
AVR32_USART0.mr = (1<<AVR32_USART_TXEN);
```

3.2 Use of Macros

There is a set of defined macros for accessing system registers. Using regular C calls cannot access these registers; they have to be accessed by the assembler functions *mtsr* and *mfsr*.

Five macros has been defined for generating the compiler independent code needed to access these registers:

- AVR32_SET_SR_BIT(sregister, bitname)
- AVR32_SET_SR_REG(sregister, regval)
- AVR32_CLEAR_SR_BIT(sregister, bitname)
- AVR32_TOGGLE_SR_BIT(sregister, bitname)
- AVR32_READ_SR_REG(sregister, return_value)

The macros are defined in the *"avr32/macro.h"* file, also included with this application note.

Example for setting the global interrupt flag in the status register:

```
AVR32_SET_SR_BIT( AVR32_SR, AVR32_SR_GM );
```

This will translate to the following code for the GCC compiler:

```
volatile long avr32_sr_set_value;
avr32_sr_set_value = __builtin_mfsr(AVR32_SR);
avr32_sr_set_value |= AVR32_SR_GM;
__builtin_mtsr(AVR32_SR, avr32_sr_set_value);
```

4 Deviance Between Hardware Registers and Header Register Naming

Some hardware registers may have names that are reserved in the C and C++ standard. The naming of these registers is renamed in the header files.

A general convention is used when a register has a reserved name; an underscore is added to the end of the register name in the header file.

Example setting the register *if* in a module *MODULE*, would according to the previous standard be written as:

```
(&AVR32_MODULE)->if = 0;
```

But due to “if” being a reserved word, the header file name of the register is altered:

```
(&AVR32_MODULE)->if_ = 0;
```

5 Macro file

Included with this application note is a header file containing macros for accessing the system register by using the *mfsr* and *mtsr*. The header file is located under src/macro.h. This file is also shipped with the header files for AVR32 devices.

5.1 Example usage

The example below set, clear, toggle and read the global interrupt flag.

```
#include <avr32/io.h>
#include “macro.h”

int main( int argc, char * argv[] )
{
     AVR32_SET_SR_BIT(AVR32_SR, AVR32_SR_GM);
     AVR32_CLEAR_SR_BIT(AVR32_SR, AVR32_SR_GM);
     AVR32_TOGGLE_SR_BIT(AVR32_SR,AVR32_SR_GM);
     volatile unsigned int readSystemRegister;
     AVR32_READ_SR_REG(AVR32_SR, readSystemRegister);
     return 0;
} /* End main */
```

5.2 Doxygen documentation

All source code is prepared for doxygen automatic documentation generation. Premade doxygen documentation is also supplied with the source to this application note, located in src/doxygen/index.html.

Doxygen is a tool for generating documentation from source code by analyzing the source code and using known keywords. For more details see http://www.stack.nl/~dimitri/doxygen/.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
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

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

32005A-AVR32-05/06