AVR Freaks

DESIGN NOTE #003

KEYWORDS:
SLEEP MODE, POWER-DOWN, IDLE, POWER-SAVE, POWER CONSUMPTION

AVR Sleep Modes

A short description of the AVR Sleep modes and how they are used, including code examples in C and Assembly language.

Introduction

The AVR family of microcontrollers features several sleep modes. The purpose of these modes is to provide a way of suspending program execution when necessary, thereby reducing power consumption.

Each AVR device features more or less sleep modes. This document is intended to summarize the most common ones and their use, thus enabling the reader to select one of the modes and easily implement this choice in Application code.

The three most common sleep modes are (in order from maximal to minimal power consumption):

- Idle mode

  The idle mode stops the CPU but leaves peripherals (UART, Analog Comparator etc.) running. The MCU will continue program execution immediately after waking up from Idle mode.
- Power-save mode

  This mode is only available in devices featuring a Timer Crystal Oscillator. It's identical to the Power-down mode, with one exception: The Timer Crystal Oscillator will continue to operate and the Timer can continue to count. The device can wake up from either a Timer Overflow or Output Compare event from this timer if the necessary conditions in TIMSK are set and the global interrupt enable bit is set in SREG (please see note 4 about the AT90(L)S8535).
- Power-down mode

  In this mode, all Oscillators are stopped while the External Level interrupts and the Watchdog (if enabled) continue operating. Only an External Reset, a Watchdog Reset or an External Level interrupt can wake up the MCU.

Only these devices feature the Power-save mode at this time:

- AT90S8535
- ATmega103
- ATmega163

These devices' distinction from the rest, is their ability to clock a Timer from an external crystal connected directly to the pins TOSC1/TOSC2.

Further information about each device's particular sleep modes and settings is available in the datasheets for the respective devices. Some devices feature additional sleep modes.

Using the Sleep Modes

In order to send a device into sleep mode, the following procedure must be followed:

- Select the desired sleep mode by setting the SM bit(s) in the MCU Control Register (MCUCR) (consult the datasheet of the specific part).
- Enable the interrupts that should be able to wake the MCU up from sleep.
- Set the SE (Sleep Enable) bit in MCUCR (usually done right before the SLEEP command as a security measure).
- Execute a "SLEEP" instruction.

Notes

Paying attention to the following notes should resolve many common issues regarding the AVR sleep modes:

1. When waking up from Power-down or Power-save mode, a delay from the wake-up condition occurs before the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is equal to the Reset period (see datasheet). Note that the time required to stabilize the power supply on Reset is not required on wake-up, only the time required to stabilize the clock. Thus, the wake-up time will be limited to the specified number of clock cycles in the datasheet.
2. If the wake-up condition disappears before the MCU wakes up and starts to execute, e.g., a low level is not held long enough, the interrupt causing the wake-up will not be executed.
3. When waking up from Power-save mode or Power-down mode by an external interrupt, a number of instruction cycles are executed before the interrupt flags are updated. When waking up by the asynchronous timer, a further delay in updating these flags is added.

   During these cycles, the processor executes the instructions following the SLEEP instruction, but the interrupt condition is not readable and the interrupt routine has not started yet. Consult the datasheet on your part to find how many cycles are executed before the interrupt flags become readable if this is of concern.
4. When waking up from Power-save mode by an Asynchronous Timer interrupt request, the AT90(L)S8535 part will wake up even if global interrupts are disabled.
5. If the Asynchronous Timer is not clocked asynchronously, Power-down mode is recommended instead of Power-save mode, because the contents of the registers in the asynchronous timer is to be considered undefined after wake-up in Power-save mode.
6. The Idle mode can be made more energy efficient if wake-up from the Analog Comparator interrupt is not required. The Analog Comparator can then be powered down by setting the ACD bit in ACSR.

Implementation; sample code

The following code demonstrates the use of the sleep modes:

C Code

```
//include bit definitions for our part
#define ENABLE_BIT_DEFINITIONS
#include <iom103.h>

//include intrinsic commands like _SLEEP();
#include <ina90.h>

void main(void)
{   //Example:
    MCUCR |= (1<<SM1)|(1<<SM2); //pwsave mode
    // .
    // .       (other programming code)
    // .
    MCUCR |= (1<<SE);         //sleep enable
    _SLEEP();
    MCUCR &= ~(1<<SE);        //sleep disable
}
```

Assembly

```
;include definitions for our part, like
;MCUCR and SM

.include "m103def.inc"


reset:
    ldi r16, (1<<SM1)|(1<<SM2)
    out MCUCR, r16    ;power save mode
    ; .
    ; .       (other programming code)
    ; .
    in  r16, MCUCR
    ori r16, (1<<SE)
    out MCUCR, r16        ;add sleep enable
    sleep
    in  r16, MCUCR
    andi r16, ~(1<<SE)
    out MCUCR, r16        ;sleep disable
```