APPLICATION NOTE

How to Securely Switch Atmel's LIN Transceiver ATA6662/ATA6662C to Sleep Mode

ATA6662/ATA6662C

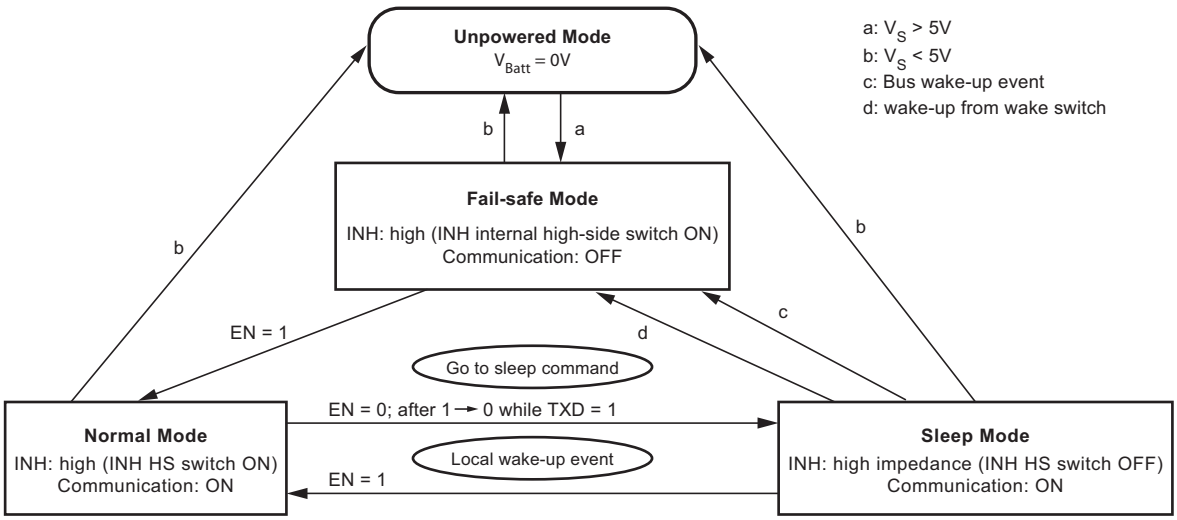
Concerning Atmel ATA6662

The goal of this document is to describe how to switch the Atmel® LIN transceiver ATA6662 into Sleep mode and how to securely handle undervoltage situations and immediate wake-ups.

This document contains simple code examples. These code examples assume that the part-specific header file is included before compilation. However, note that not all C compiler vendors include bit definitions in the header file. The code examples are dedicated to the AVR® microcontrollers and the IAR™ C compiler.

Figure 1 on page 2 shows the operating modes available in the Atmel ATA6662 as well as the transitions between them. It is only possible to switch the device to Sleep mode from Normal mode.

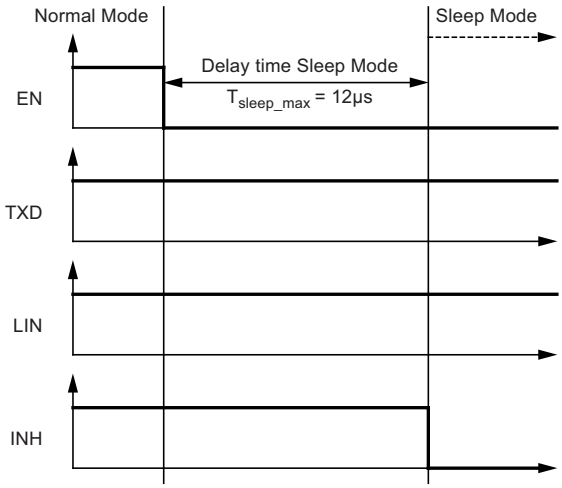
Figure 1. Atmel ATA6662 Operating Modes



1. Simple Switching of the Atmel ATA6662 to Sleep Mode

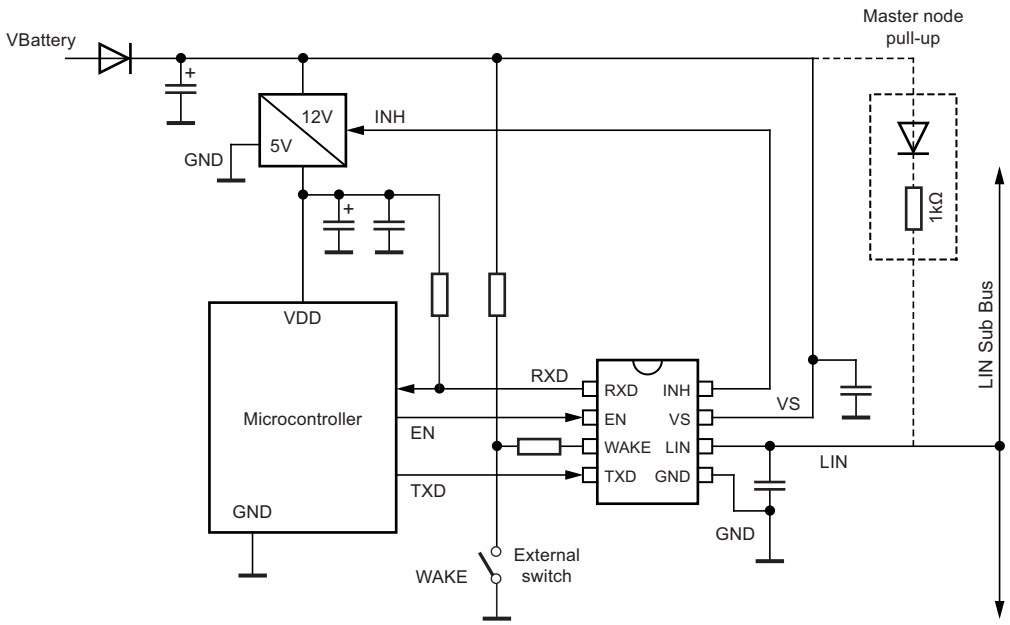
In order to switch the Atmel® ATA6662 to Sleep mode, the EN pin has to be pulled to GND and TXD has to remain at high level. After the time T_{sleep} (beginning directly at the falling edge of the Enable signal) has expired, the device enters Sleep mode and the INH output is switched off.

Figure 1-1. Switch to Sleep Mode



In a typical application, as depicted in Figure 1-2, switching off the INH-Pin switches off the voltage regulator of the module and thus the microcontroller is not longer supplied. Therefore, the complete module has very low current consumption until a wake-up occurs.

Figure 1-2. Typical Application



Either a remote wake-up via LIN or a local wake-up via the WAKE pin can be used to wake up the module. The device also wakes up with an undervoltage at the pin Vs. In all cases the LIN transceiver then enters the fail-safe mode, the INH output is switched on and the voltage regulator supplies the microcontroller.

1.1 Simple Switching C Routine

The following C function can be used to switch the Atmel® ATA6662 to Sleep mode.

```
void Switch_ATA6662_To_SleepMode (void)
{
	// Before setting the EN pin to low it has to be sure, that the TXD line
	// remains high. Therefore the pin PD1 (= TXD in at the ATmega88/168) has to be
	// set to output high here at the latest. The output high will be overwritten
	// by the UART as long as the UART transmitter is enabled. Therefore the UART
	// transmitter has to be disabled.
	PORTD I= (1<<PD1); // Set pin PD1 to output high (part 1)
	DDRD I= (1<<TXEN); // Set pin PD1 to output high (part 2)
	UCSRB &= ~(1<<TXEN); // Disable the UART transmitter.
	// Now the actual switching of the ATA6662 into the Sleep Mode takes place.
	// Assuming that the EN pin of the ATA6662 is connected to pin PD4 and that
	// this pin is already set to output.
	PORTD &= ~(1<<PD4); // Switch EN pin to low
	// From now on the proper supply of the microcontroller will be switched off
	// sooner or later. The microcontroller is waiting for this to happen.
	while (1)
	{
		// Active waiting for the power supply to be switched off
		_no_operation();
	};
}
```

2. Potential Problems Using Simple Switching Without Additional Safety Precautions

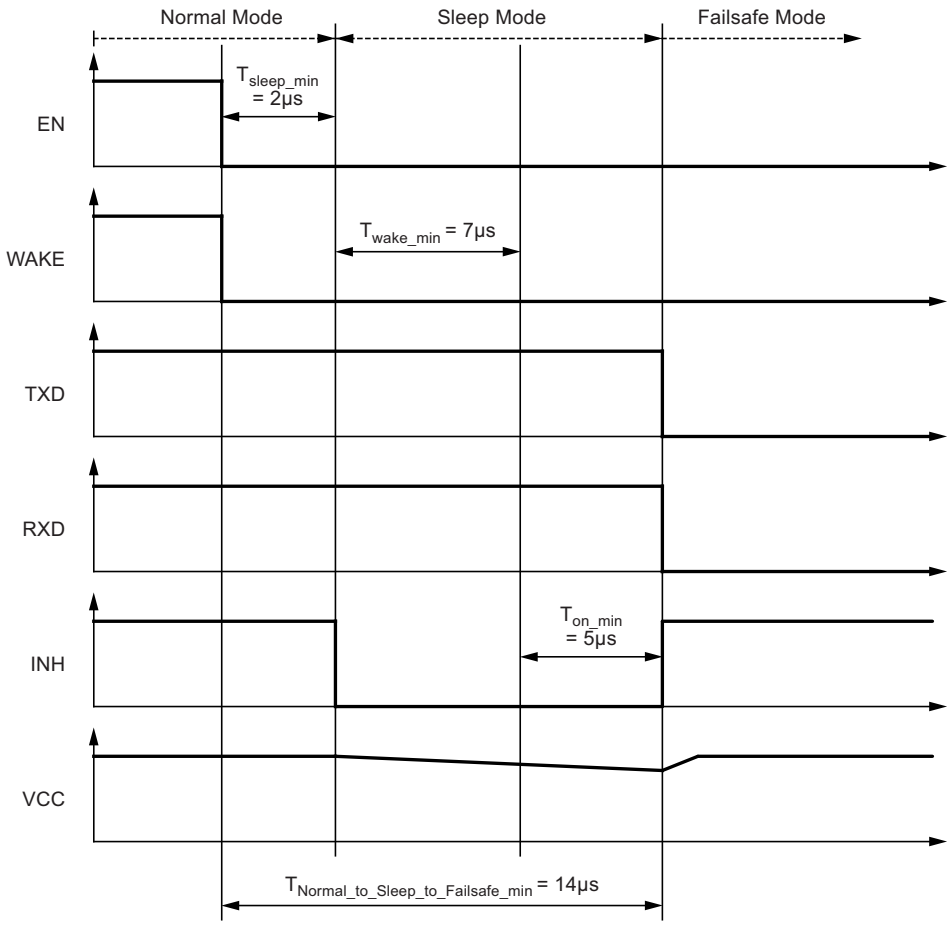
After switching the Atmel® ATA6662 to Sleep mode, there are two scenarios which can lead to the situation where the connected microcontroller remains powered. With the implementation of the Go-to-Sleep mode function as shown under the heading “Simple Switching C Routine” on page 4, the microcontroller waits for the blocking capacitors to be discharged and the supply voltage to end; however, as this situation never occurs, the complete module ends in a dead-lock.

2.1 Immediate Wake-up after Going-to-sleep

A very short low phase at the INH pin in combination with the blocking capacitors at the output of the voltage regulator is the basis for the first problem scenario.

In order for this situation to occur, one possibility is the appearance of a Wake signal at the same time or slightly later than the falling edge of the EN signal. In this case, the device will wake up again and in the worst case, the INH pin has been at low level for just 12µs. The relevant signals during this scenario are illustrated below.

Figure 2-1. Immediate Wake-up after Going-to-sleep



Depending on the capacitances of the blocking capacitors at V_{cc} and the current consumption of the microcontroller, it is possible that the microcontroller remains powered during the 12µs.

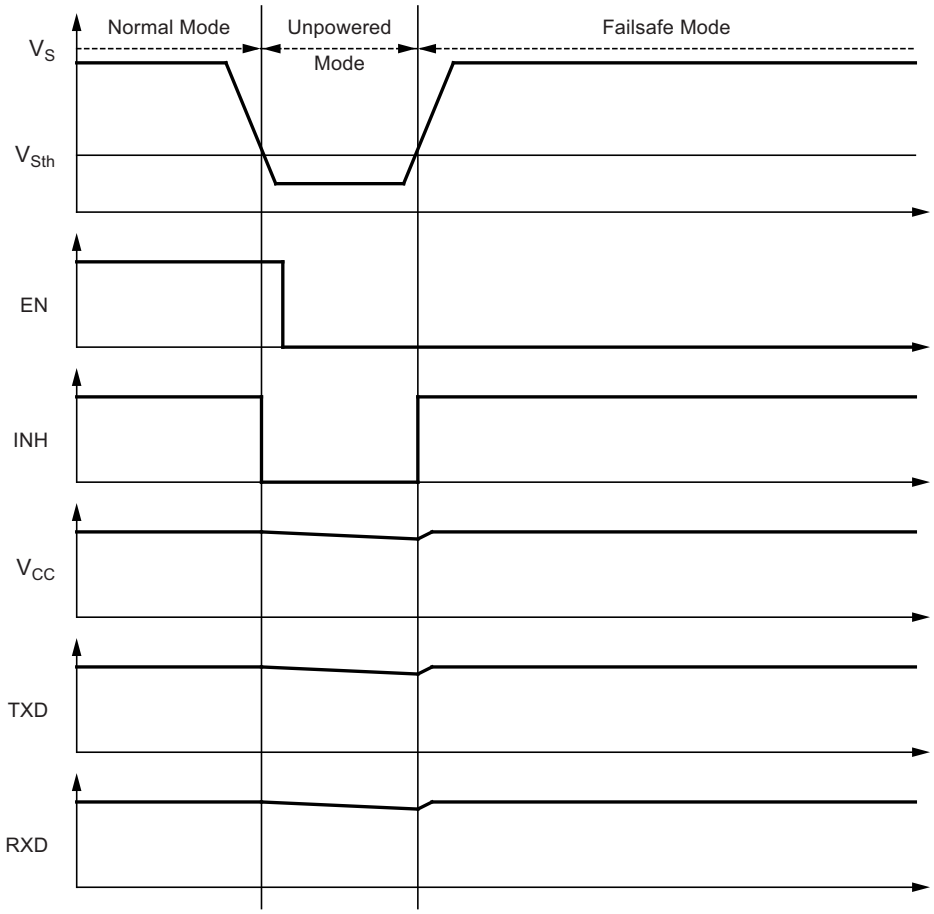
2.2 Undervoltage During Switching to Sleep Mode

The second scenario that may lead to a dead-lock of the module occurs with battery voltages below +5V. In this case, the Atmel® ATA6662 automatically switches into unpowered mode without providing information about this to the microcontroller. The unpowered mode is in general the same as the fail-safe mode; however, the Atmel ATA6662 does not change its state according to the EN pin.

If the microcontroller does not change the state of the EN pin (the EN pin remains high) during this time, then the Atmel ATA6662 re-enters Normal mode after the battery voltage has automatically recovered. During the period of undervoltage, no data transmission via LIN is possible, which is in accordance to the LIN specification.

If the Atmel ATA6662 device detects undervoltage and it has changed to unpowered mode, any further mode change via the EN pin does not lead to any action. During the unpowered mode, the INH pin switches to low level. Due to the blocking capacitors at the microcontroller's Vcc pin, the microcontroller remains active for a certain time. As the microcontroller has no information about the Atmel ATA6662 being in the unpowered mode, it is possible that the microcontroller intends to switch the device into the Sleep mode at or directly after this very point of time. This scenario is shown in Figure 2-2.

Figure 2-2. Undervoltage During Switching to Sleep Mode



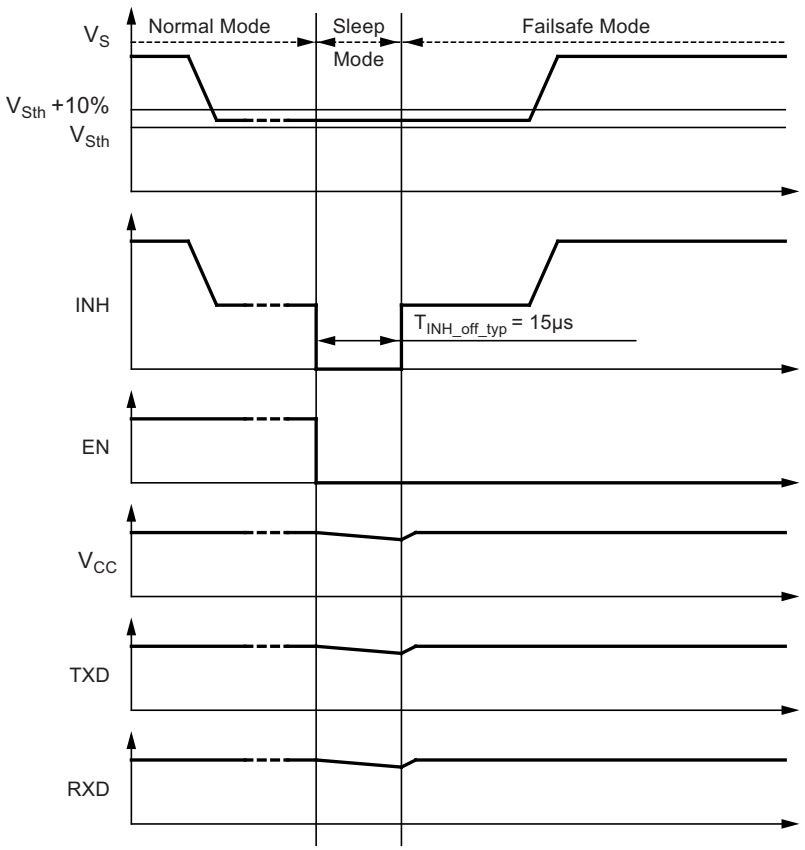
After the disappearance of the undervoltage at Vs, the Atmel ATA6662 switches from unpowered mode into fail-safe mode and therefore, the INH pin switches to high again. The Atmel ATA6662 stays in fail-safe mode as the EN pin is still low. However, the microcontroller expects the ATA6662 to have entered Sleep mode and it is caught in the active waiting loop for its power supply to be switched off. The microcontroller does not put the Atmel ATA6662 into Normal mode via the EN pin and the module remains in a dead-lock situation isolated from the LIN communication.

2.3 Behavior of the Atmel ATA6662 When Powered from a Voltage Slightly Above the Undervoltage Threshold

In the third scenario, the behavior of the Atmel ATA6662 when using the software routine detailed in Section 1.1 “Simple Switching C Routine” on page 4 is the root cause for the potential dead-lock.

Switching the Atmel ATA6662 into Sleep mode while the supply voltage of the Atmel ATA6662 is above the undervoltage voltage threshold but below the undervoltage voltage threshold +10% forces the device to enter the fail-safe mode typically 15µs after having entered the Sleep mode (10µs ≤ TINH_off ≤ 20µs). The reason for this is that the undervoltage detection of the Atmel ATA6662 is fed with a slightly increased reference voltage during the first microseconds in the Sleep mode. Therefore, the device changes into unpowered mode. In this mode the undervoltage detection is supplied with the correct reference voltage and therefore, no undervoltage is detected anymore and the Atmel ATA6662 switches directly to fail-safe mode. As the very short step through the unpowered mode is not important for the behavior itself it has been omitted in the following graph, which shows the scenario described above.

Figure 2-3. Behavior at a Supply Voltage Slightly Above the Undervoltage Threshold



3. Solving the Potential Dead-lock Problems

That the microcontroller remains powered while the Atmel® ATA6662 has entered and left Sleep mode and remains in fail-safe mode is common to all of the described dead-locked scenarios. The module's power supply has been switched off for a couple of microseconds, however this does not always lead to an undervoltage reset of the microcontroller due to the blocking capacitors. As the microcontroller is in an endless loop expecting either a reset signal or its supply voltage to disappear, both of which never occur, the module remains in a dead-lock.

This scenario can be avoided using either a hardware solution or a software solution.

3.1 Hardware Solutions

3.1.1 Using an External Watchdog

When the microcontroller remains alive and is waiting for its power supply to disappear, an external watchdog can reset the microcontroller. The precondition for this solution is, of course, that the microcontroller stops serving the watchdog before entering the waiting loop.

3.1.2 Using a Dedicated External Voltage Regulator with Reset Output

As the Atmel ATA6662 indicates all of the above-mentioned scenarios with a low phase of the INH signal, which normally leads to an undervoltage reset of the connected microcontroller, the voltage regulator used should react on this INH signal quickly and directly. The voltage regulator should be able to generate a reset signal for the microcontroller as long as the INH input is low. Switching off the output voltage might not be sufficient as the remaining output voltage stored in the blocking capacitors can be sufficient to keep the microcontroller alive.

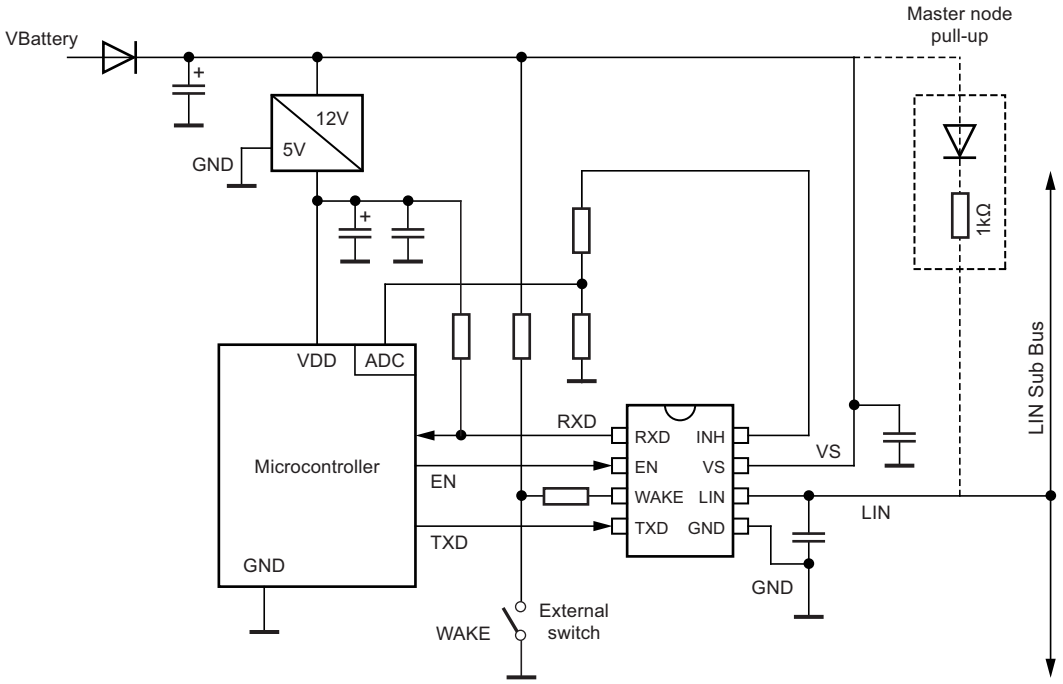
3.1.3 INH Monitoring for Resetting the Microcontroller

If the voltage regulator cannot generate an INH-dependant reset signal, then the microcontroller itself can take over this task. The easiest solution is to connect the INH pin to a dedicated I/O-Pin of the microcontroller for monitoring the logic level of the INH pin. The microcontroller can then generate an internal reset according to the INH status.

But as the INH pin is a battery-related output, special care must be taken when connecting this pin to the microcontroller. With an additional voltage divider between the INH output pin and ground the logical level of the INH output can easily be read out and the voltage level at the microcontroller input is limited to noncritical values. With this voltage divider it is also possible to monitor the battery voltage via the INH output pin. In this case the voltage divider has to be connected to an A/D converter input of the microcontroller as shown in Figure 3-1 on page 9.

The current consumption of the module is not increased in Sleep Mode as the INH will be switched off in the Sleep Mode and therefore, the voltage divider is not active in this mode.

Figure 3-1. Monitoring the Battery Voltage



3.2 Software Solution

In addition to the hardware based solutions, it is also possible to identify when the Atmel® ATA6662 is not in Sleep mode as expected via a software routine. When the microcontroller expects its power supply to disappear, an additional time-out can be implemented to cancel the waiting loop after the time-out has expired. The necessary time-out time mainly depends on the value of the blocking capacitor and on the output current of the voltage regulator. Using the following equations, the minimum time-out time necessary can be easily calculated.

$Q = C \times U$ (1-1)
$Q = I \times t$ (1-2)
$C \times U = I \times t$ (1-3) using (1-1) and (1-2)
$\Rightarrow t_{min} = C \times U/I$ (1-4)

C: Total of all blocking capacitors at the power supply of the microcontroller

I: Total output current of the voltage regulator to which the microcontroller is connected.

U: Voltage difference of the nominal output voltage to the undervoltage reset threshold (if there is one) or the brown-out detection threshold.

The resulting time is the minimum time for the blocking capacitors to be discharged. As the current output is almost never constant, the calculated time should be doubled (at least) to ensure that enough time has elapsed before leaving the endless loop.

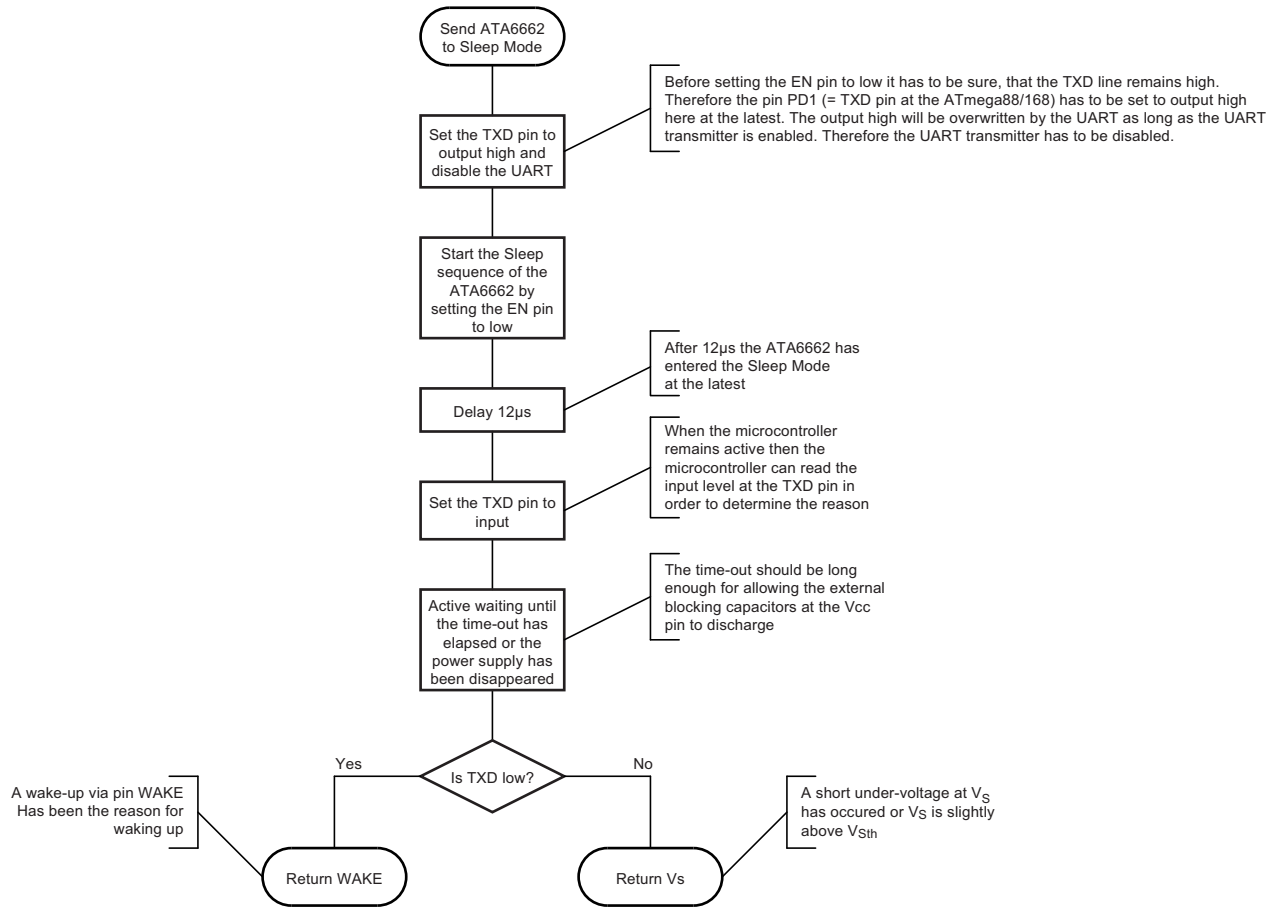
As the variable values are highly application-dependant it is not possible to state a fixed time-out time valid for all applications.

In order to discharge the blocking capacitors as fast as possible resulting in an earlier undervoltage reset, the current consumption should not be reduced by the microcontroller while waiting for the power supply to disappear by entering a Current Save mode.

3.2.1 PAP

The following programming flowchart shows a secure way of switching the Atmel® ATA6662 to Sleep mode. The flowchart also identifies the previously described situations in which the microcontroller stays alive when using the C routine from “Simple Switching C Routine” on page 4.

Figure 3-2. Flow Chart



Using this flowchart, the calling function will report the reason why the Atmel ATA6662 has left Sleep mode and why the microcontroller remains active. The calling function then has to decide how to proceed.

3.2.2 C Routine

```
unsigned char Switch_ATA6662_To_SleepMode (void)
{
      // Before setting the EN pin to low it has to be sure, that the TXD line
      // remains high. Therefore the pin PD1 (= TXD pin at the ATmega88/168) has
      // to be set to output high here at the latest. The output high will be
      // overwritten by the UART as long as the UART transmitter is enabled.
      // Therefore the UART transmitter has to be disabled.
      PORTD I= (1<<PD1);         // Set pin PD1 to output high (part 1)
      DDRD I= (1<<PD1);          // Set pin PD1 to output high (part 1)
      UCSRB &= ~(1<<TXEN);       // Disable the UART transmitter

      // Now the actual switching of the ATA6662 into the Sleep Mode takes place
      // Assuming that the EN pin of the ATA6662 is connected to pin PD4 and that
      // this pin is already set to output
      PORTD &= ~(1<<PD4);        // Switch EN pin to low
      // From now on the power supply of the microcontroller will be switched
      // off sooner or later. The microcontroller is waiting for this to happen.

      // Delay 12µs
      // Assuming the device is running on a 1MHz RC oscillator
      // Including the file intrinsics.h is required for using the delay function
      _delay_cycles(12;

      // Set the TXD pin to input. The internal pull-ups will be active then
      // because of the former PORTD setting
      DDRD &= ~(1<<PD1);         // Set pin PD1 to input

      // Delay 5ms for the power supply to be switched off. The time-out should
      // be adapted to the particular application
      _delay_cycles(5000);

      // Normally the microcontroller will never reach the following statements,
      // as the power supply should have been switched off long ago
      // If the TXD pin is high, then either a short undervoltage at VS was
      // present while switching to Sleep Mode or VS is slightly above the VS
      // undervoltage threshold and the ATA6662 has been woken because of this.
      // If the TXD pin is low, then an immediate and regular Wake-up via the
      // pin WAKE (local wake-up) was detected

      if (PIND & (1<<PD1))
            return VS;
      else
            return WAKE;
}
```

VS and WAKE have to be defined as global parameters.

4. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

Revision No.	History
9155D-AUTO-06/15	• Put document in the latest template
9155C-AUTO-05/11	• Section 4.1.3 “INH Monitoring for Resetting the Microcontroller” on page 8 updated
9155B-AUTO-03/11	• ATA6662C on page 1 added

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**