

---

# AVR2007: IEEE802.15.4 MAC power consumptions for AT86RF230 and ATmega1281

## Features

- General considerations
- MAC protocol usage
- PHY mode usage
- Scilab

## 1 Introduction

This Application Note describes two ways of estimating the current consumption of the AT86RF230 radio and the ATmega1281 microcontroller as a transceiver system for the IEEE802.15.4™ standard.

Both the transceiver IC and the microcontroller are characterized for their current consumption and scenarios are shown to find a common denominator for applying algorithms to.

Based on a standardized scenario the physical layer operation (PHY) is evaluated and later adapted to the IEEE802.15.4 Media Access Control layer (MAC). Atmel's webpage provides a Scilab implementation for the described scenarios to evaluate more complex applications. A second approach is shown to estimate the system current consumption for battery lifetime, based on only a few system parameters. A calculation procedure using a trivial BASIC interpreter will be shown in this document.



## Application Note

Rev. 8100A-AVR-08/07

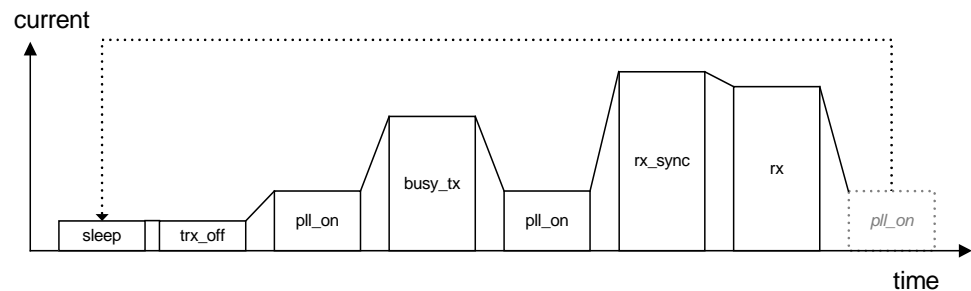


## 2 Transmission Scenario for Network Devices

### 2.1 General

Instead of just dealing with ideal cases, this calculation is based on the AT86RF230's main state machine and a real world scenario. Multiplying on-air time and the transmission current of the AT86RF230 (radio) and ATmega1281 may not be accurate enough for some cases. If this simplified scenario is accurate enough, go to chapter 4.5.

**Figure 2-1.** Standardized transmission scenario for a PHY



The used scenario calculates the energy for one device node of the network. One goal is it to find the lifetime of a battery, when the capacity is given. Therefore it is assumed, that the device has only one major task, which is transmitting sensor data to its router or coordinator. The device will transmit a given amount of data bytes in a given time period. I.e. if 100 bytes are to be transmitted per 5 seconds the device needs more energy than for an application transmitting 10 bytes per 10 minutes. The standard scenario takes into account that every data frame is acknowledged, but not retransmitted if the first attempt fails, even if IEEE802.15.4 defines retries.

So basically two values are necessary to know:

- How many bytes are to be transmitted and acknowledged?
- In which period the device transmits again?

See Figure 2-1 for a schematic drawing of the device's current consumption over time. Figure 2-1 shows the states of the main state machine of the AT86RF230 and weights their current consumption.

### 2.2 PHY

The PHY transmission scheme follows the standardized procedure as shown in Figure 2-1. After initialization is done, the AT86RF230 starts from SLEEP mode and switches to IDLE mode (or as stated in the datasheet, to TRX\_OFF mode). Since data should be transmitted, the PLL\_ON mode is entered shortly to end up in BUSY\_TX mode. Here the actual payload is transmitted, so the AT86RF230 stays in BUSY\_TX mode for the IEEE802.15.4 preamble (inclusive SFD field) and the data bytes.

After the transmission is finished, the acknowledge frame should be received, so the AT86RF230 leaves the BUSY\_TX state, going through PLL\_ON and ends up in the RX\_ON state, where it tries to synchronize to an IEEE802.15.4 acknowledge frame. If the preamble of the acknowledge frame is detected, the receiving device switches to BUSY\_RX mode as soon as the preamble inclusive the SFD byte is received. It stays there for the length of 5 bytes. If the acknowledge frame is received, the AT86RF230 leaves the BUSY\_RX mode back to the PLL\_ON state. The microcontroller (MCU) switches the radio to SLEEP mode as fast as possible. The radio enters sleep mode and stays there as long as the MCU does not initiate a new transmission.

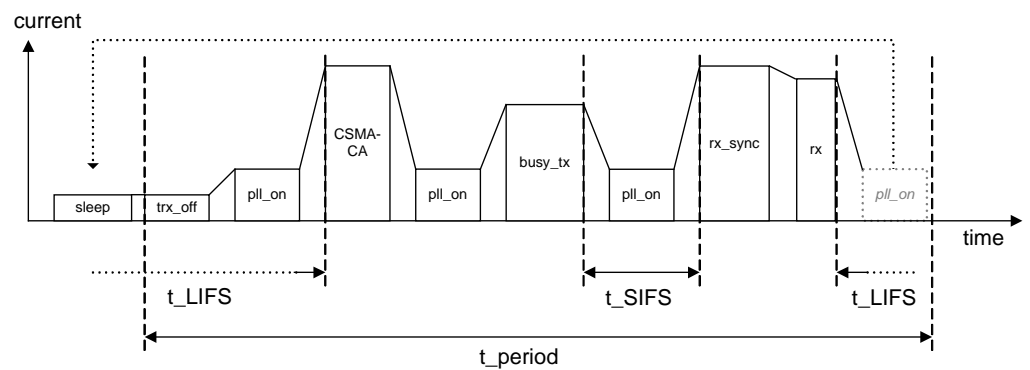
## 2.3 MAC

The MAC transmission scheme follows the PHY procedure, but the IEEE802.15.4 standard adds some rules here.

- Before transmission is initiated, the device has to check for a free channel via Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA)
- After a frame was sent the next transmission in the network may start after a time called Short Inter Frame Spacing (SIFS), when the next transmission is an acknowledge frame (ACK)
- After an acknowledge frame was transmitted, the next data or command frame in the coordinator's range may be sent after a time called Long Inter Frame Spacing (LIFS)

So knowing this, the standardized procedure must be adopted to these rules as Figure 2-2 shows. The CSMA-CA period, the SIFS and LIFS periods are introduced here.

**Figure 2-2.** Standardized transmission scenario for MAC





## 3 Current Consumption Modes

### 3.1 Microcontroller ATmega1281

Atmel offers an open-source MAC-layer for the AT86RF230 radio. The used MCU is the ATmega1281, which has several sleep modes (see Table 3-1). The MAC release (version 1.5) supports the sleep mode of the AT86RF230 radio. This and other MAC layers have to provide support functions, which must be available in the radios sleeping mode too. So the Atmel MAC uses the IDLE mode to save power.

**Table 3-1.** ATmega1281 Sleep Modes

Mode	Running MCU parts
Idle Mode	SRAM, timer, SPI, IRQs
Power Down Mode	None
Power Save Mode	Async. Timer
ADC Noise Reduction Mode	Async. Timer, ADC
Extended Standby Mode	Main oscillator, async. Timer

Using two of ATAVRRZ502+STK<sup>®</sup>500 systems with Atmel's open-source MAC to execute an IEEE802.15.4<sup>™</sup> association procedure on the device side, an oscilloscope connected via a current tongs to the VTARGET jumper on STK500 shows Figure 3-1. The same procedure, when the current tongs is connected to the jumper on ATARVRZ502, looks like Figure 3-2. From these screenshots two values can be taken. The maximal drawn current for the radio - MCU combination is  $I_{\text{combi}} = 36\text{mA}$ , the current of the radio is, during transmission periods,  $I_{\text{RF230}} = 15.4\text{mA}$ . So the MCU needs  $I_{\text{MCU}} = 20.6\text{mA}$  therefore. Now the MCU current depends on the operating conditions. MCU clock is 8MHz, MCU supply voltage is 3V and running parts are SPI, UART, Timer 0 and Timer 1. The MCU's datasheet states on pages 372ff the current consumption of the single MCU circuits. Solving Equation 3-1 will lead to a current consumption for the running circuits of 1.8mA when applying Equation 3-1 if the core is stopped.

**Equation 3-1.** Current calculation for MCU

$$I_{CC} = I_{Active} \left( 1 + \frac{I_{UART}}{100} + \frac{I_{Timer1}}{100} + \frac{I_{Timer0}}{100} + \frac{I_{SPI}}{100} \right)$$

$$I_{UART} = 0.03 \cdot I_{Active}$$

$$I_{Timer1} = 0.018 \cdot I_{Active}$$

$$I_{Timer0} = 0.015 \cdot I_{Active}$$

$$I_{SPI} = 0.033 \cdot I_{Active}$$

Figure 3-1. Current flowing through jumper VTARGET on STK500

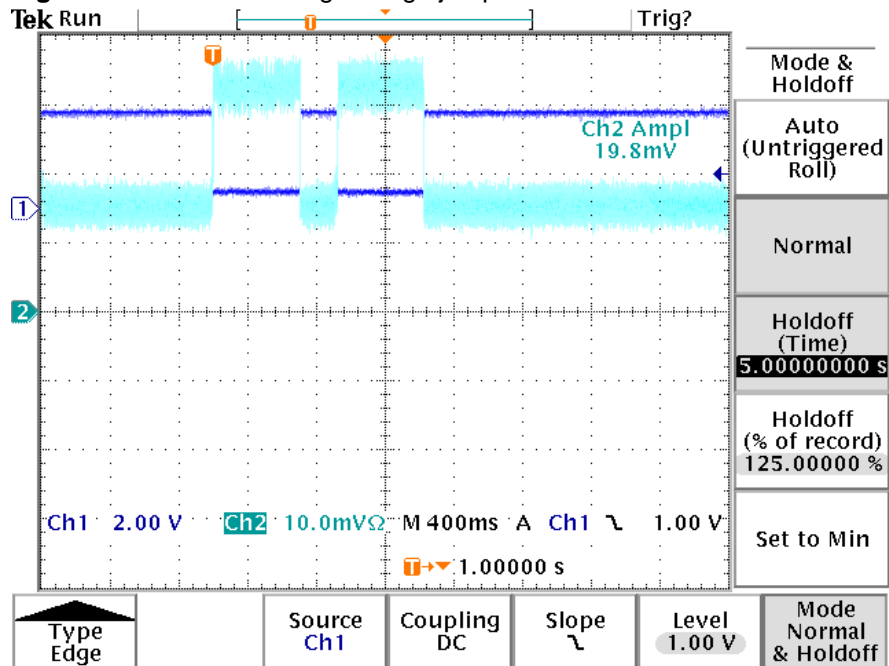
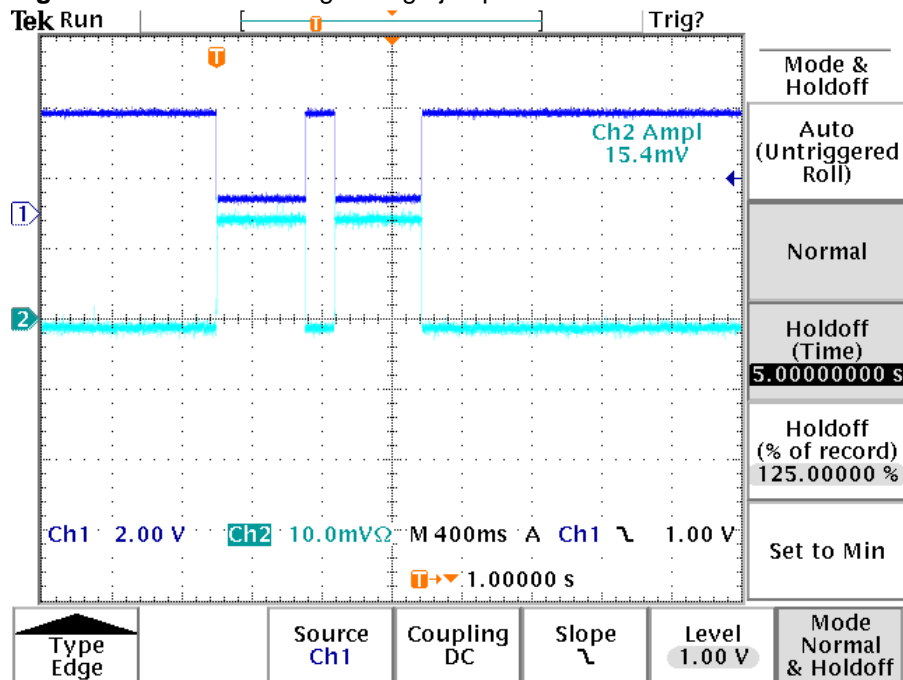


Figure 3-2. Current flowing through jumper on ATAVRRZ502





## 3.2 Transceiver IC AT86RF230

Atmel offers an open-source MAC-layer for the AT86RF230 radio. The used radio is the AT86RF230, which supports a sleep mode. Table 3-2 contains the currents for all modes, which are used for current consumption calculation.

**Table 3-2.** AT86RF230 modes

Mode	Measured current at max. output power
TRX_OFF Mode	1.7 mA
BUSY_TX Mode	17 mA
BUSY_RX Mode	15.4 mA
RX_ON Mode	15.7 mA
PLL_ON Mode	8.5 mA

## 4 Scilab Scripts

### 4.1 General

Scilab is a matrix oriented calculation package (see [2]) and is base for the two scripts "*mac\_FrameEnergyCalc.sce*" implementing the MAC transmission procedure and "*phy\_rf230energy.sce*" implementing the PHY transmission procedure. These scripts may be used to find the RMS current consumption or battery lifetime of the system consisting of the AT86RF230 and a microcontroller, especially the ATmega1281.

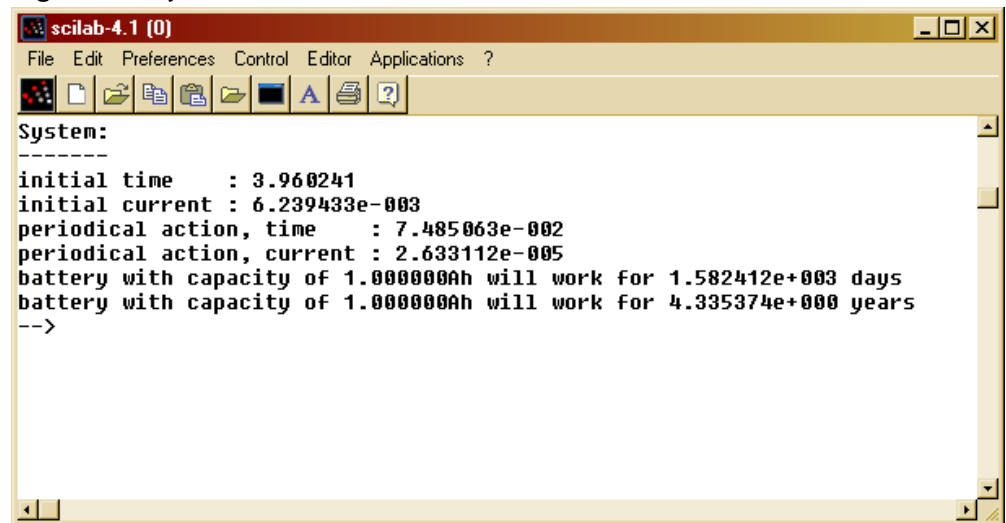
Both scripts contain a section called "*application specific parameters*" for adjusting the following values to describe the desired scenario.

- *nb\_payload\_bytes* contains the transmitted MAC payload
- *t\_period* describes the repeating period of the application
- *battery\_capacity* keeps the capacity of the battery
- *EU\_US\_HIGH* contains the setting for the frequency band
- *BE* is the MAC attribute minMacBE as the minimal backoff exponent
- *i\_mcu* is the current of the MCU in active mode
- *i\_mcu\_sleep* is the current of the MCU in sleep mode
- *i\_radio\_sleep* is the current of the AT86RF230 in sleep mode

The result of the script "*mac\_FrameEnergyCalc.sce*" is separated into three parts. Figure 4-1 shows the first part, MAC and PHY layer's transmission and reception actions. Figure 4-2 shows the second part, the application events including time, current and state. The last part is shown in Figure 4-3 and gives the system results like battery lifetime and RMS current consumption of AT86RF230 and ATmega1281.



Figure 4-3. system results



## 4.2 Simulation Comments

Since a simulation or a calculation is always based on assumptions, the results show only the output of an algorithm including those assumptions.

The here used algorithm incorporates the named devices and no other system elements. In the case a sensor is used to produce the transmitted data, the current of this sensor device is not used by the algorithm and is therefore not included in the results.

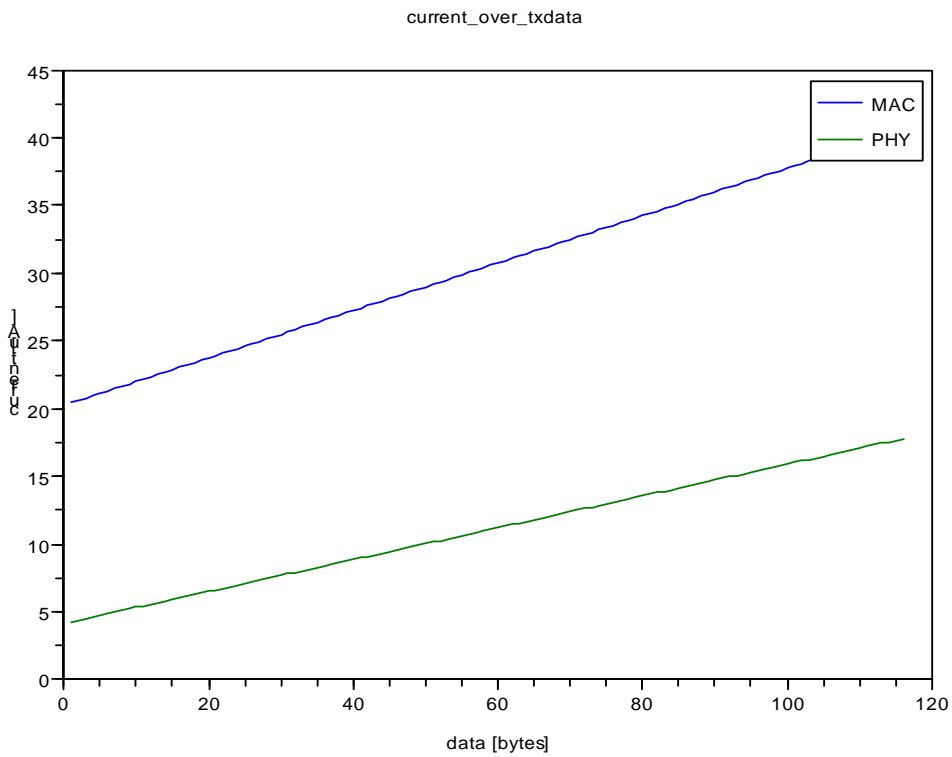
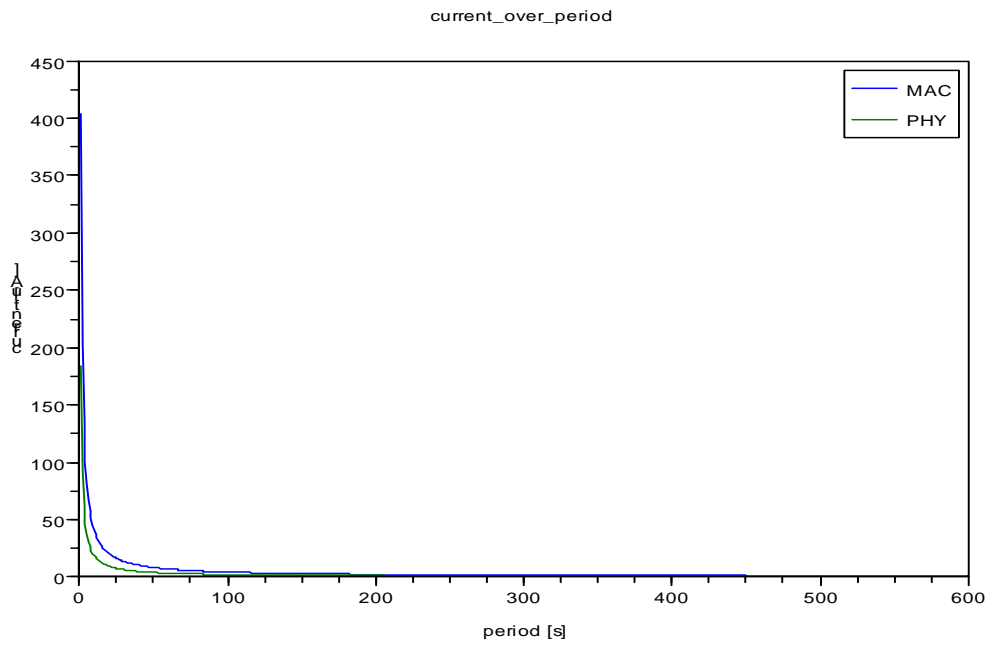
Even if pull resistors are used, their drawn current is not taken into account. The algorithm uses current and time values, given in the datasheets of the AT86RF230 and ATmega1281. Additional energy consuming elements may be added to the variables *i\_mcu* and *i\_mcu\_sleep*.

The both scripts "*mac\_FrameEnergyCalc.sce*" and "*phy\_rf230energy.sce*" are used as a base to sweep the input values of the algorithm. The resulting diagrams are the more interesting things here.

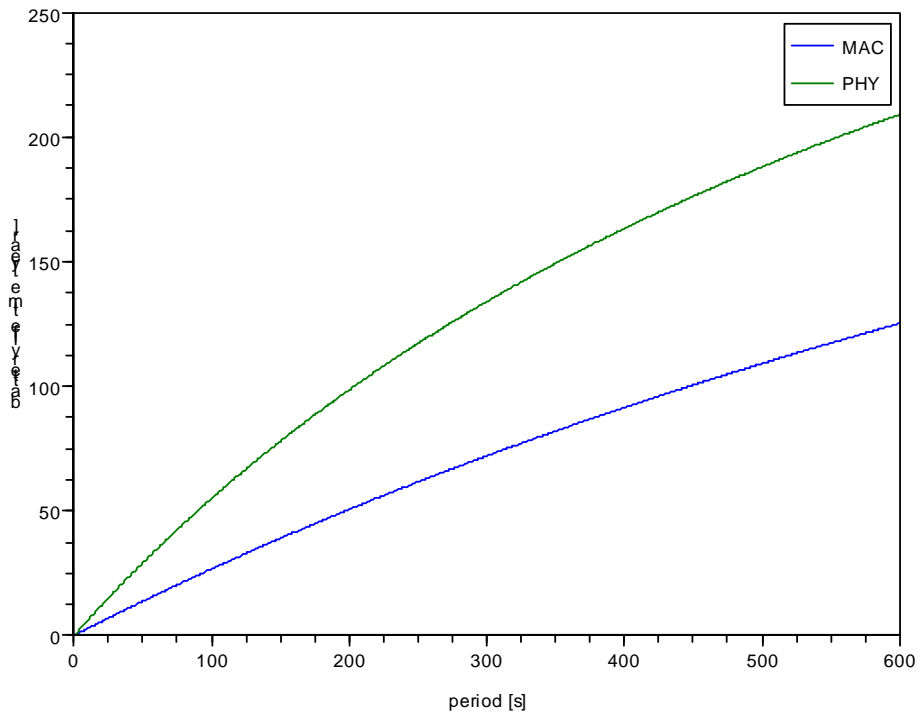
As it is seen here, the MAC layer is overhead for the whole system concerning current consumption. So battery life time is shorter and more current is necessary using the MAC layer, explainable of course due to the enlarged protocol and the MAC header in the payload (9 to 25 bytes, depending on the addressing scheme). The curves in figure 2-4 for battery lifetime assume the payload to be fixed at 127 bytes for PHY and 116 bytes for MAC data (automatic added header of 11 bytes). The curves for current over payload assume the application period to be fixed at 10 seconds. So the battery lifetime curves follow the common behavior, that with increasing period, the lifetime increases too, following the reciprocal model. Enlarging the transmitted payload enlarges the RMS current of the system in the linear way. Figure 4-4 shows comparing results for MAC and PHY.

4.3 Calculation Results

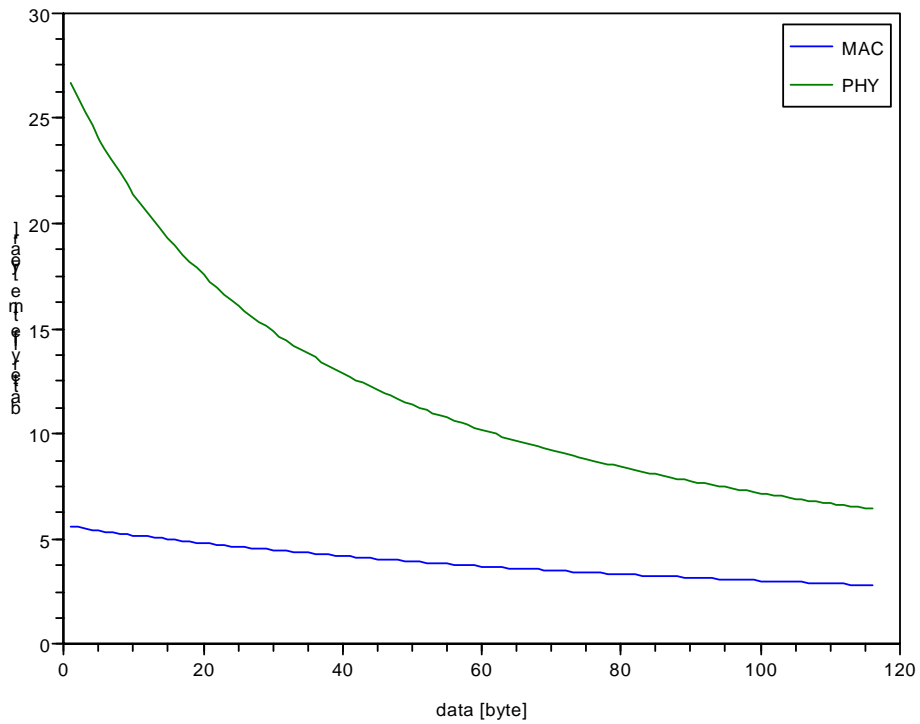
Figure 4-4. Algorithm results over complete input range at constant battery capacity of 1000mAh



runtime\_over\_period



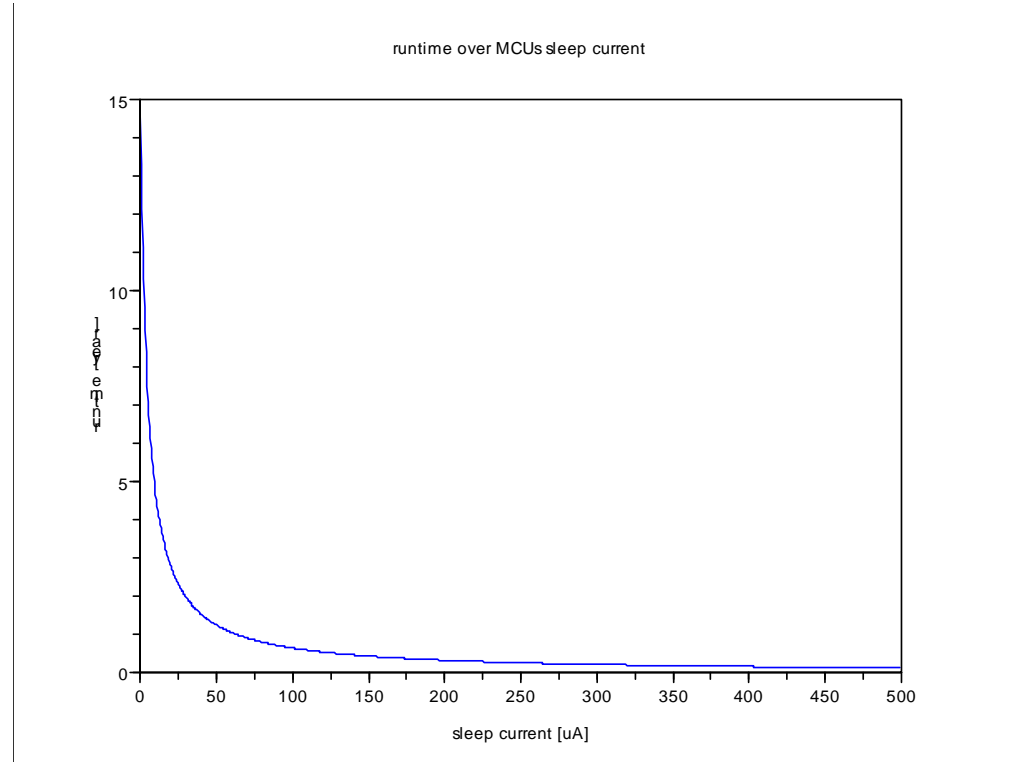
runtime\_over\_txdata



With a constant operation period and constant amount of transmitted data, the battery runtime (capacity = 1000mAh) will follow the curve shown in Figure 4-5. So creating a system with maximized runtime, the system designer may optimize the sleep current and the application's period with higher attention.

#### 4.4 Sleep Current

**Figure 4-5.** Sleep current influences the system runtime ( $t_{\text{period}} = 60\text{s}$ ,  $\text{MAC}_{\text{payload}} = 116\text{ byte}$ )



#### 4.5 Simple System Estimation

The Scilab scripts are based on the "Bottom-Up" view. A system developer may not want to step into the topic in the initial phase of a specification. So here a "Top-Down" approach is shown.

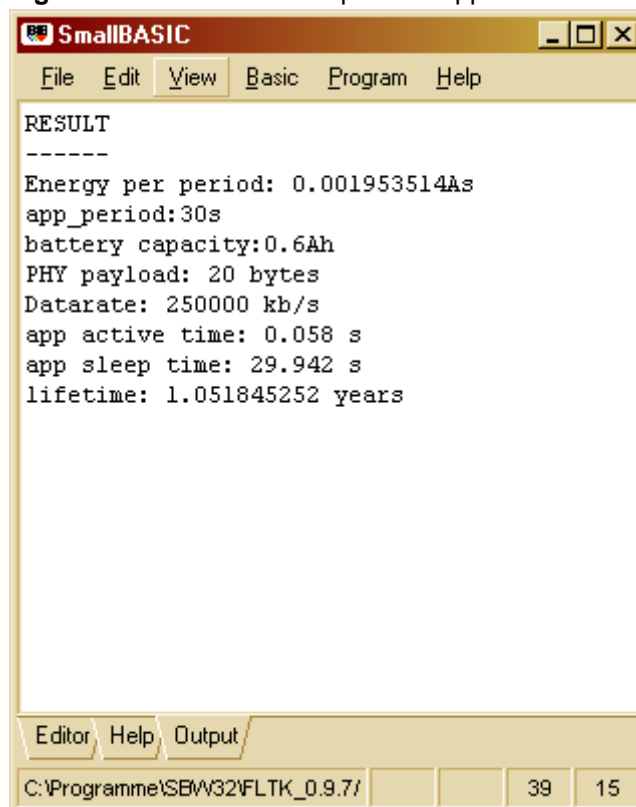
For most tasks a device node can be seen as a repetitive transmission node. So the application will transmit a constant amount of bytes in a specified period. I.e. the device transmits 20 bytes each minute to its coordinator. The rest of the period the device is sleeping. For this simple calculation, only the static currents and the duration of the four phases Transmission, Reception, MCU running and Sleeping have to be known. A simple BASIC implementation (i.e. SmallBasic, see [3]) is shown in Table 4-1. The results are shown in Figure 4-6.



**Table 4-1.** Exemplary implementation of the Top-Down approach

```
'currents
tx_mcu           = 18e-3
tx_trx           = 17.8e-3
tx_peripheral    = 10e-3
rx_mcu           = 18e-3
rx_trx           = 15.7e-3
rx_peripheral    = 10e-3
run_mcu          = 18e-3
run_trx          = 140e-9
run_peripheral   = 10e-3
sleep_mcu        = 10e-6
sleep_trx        = 100e-9
sleep_peripheral = 0
'application parameter
system_capacity  = 0.6
application_period = 30.0
payload          = 20
datarate         = 250e3
application_rxtime = 1e-3
application_active_time = 58e-3
'calc
air_time = payload*8/datarate
i_tx     = tx_mcu+tx_trx+tx_peripheral
i_rx     = rx_mcu+rx_trx+rx_peripheral
i_run    = run_mcu+run_trx+run_peripheral
i_sleep  = sleep_mcu+sleep_trx+sleep_peripheral
t_tx     = air_time
t_rx     = application_rxtime
t_run    = application_active_time-application_rxtime-air_time
t_sleep  = application_period-application_active_time
e_tx     = t_tx*i_tx
e_rx     = t_rx*i_rx
e_run    = t_run*i_run
e_sleep  = t_sleep*i_sleep
e_sum    = e_tx+e_rx+e_run+e_sleep
print "RESULT"+chr(13)+"-----"
print "Energy per period: ";e_sum;"As"
print "app_period: ";application_period;"s"
print "battery capacity: ";system_capacity;"Ah"
print "PHY payload: ";payload;" bytes"
print "Datarate: ";datarate;" kb/s"
print "app active time: ";application_active_time;" s"
print "app sleep time: ";t_sleep;" s"
lifetime_years=1/8760*system_capacity*application_period/(e_sum)
print "lifetime: ";lifetime_years;" years"
```

Figure 4-6. Result of the Top-Down approach for Table 4-1



The screenshot shows the SmallBASIC application window. The title bar reads "SmallBASIC". The menu bar includes "File", "Edit", "View", "Basic", "Program", and "Help". The main text area displays the following output:

```
RESULT
-----
Energy per period: 0.001953514As
app_period:30s
battery capacity:0.6Ah
PHY payload: 20 bytes
Datarate: 250000 kb/s
app active time: 0.058 s
app sleep time: 29.942 s
lifetime: 1.051845252 years
```

At the bottom of the window, there are tabs for "Editor", "Help", and "Output". The status bar at the very bottom shows the file path "C:\Programme\SEW32\FLTK\_0.9.7/" and two small boxes containing the numbers "39" and "15".

It implements the four phases of one period:

- Transmission of the data
- Reception of the Acknowledge, inclusive some timing uncertainties
- MCU calculation without using the radio
- Sleeping for the rest of the application's period

It does not take specialties into account like CSMA-CA, retries or others. It is a PHY-only estimation with no build-in intelligence.

## 5 References

- [1] [www.atmel.com](http://www.atmel.com)
- [2] [www.scilab.org](http://www.scilab.org)
- [3] <http://smallbasic.sourceforge.net>



## Headquarters

---

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

## International

---

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

---

**Atmel Europe**  
Le Krebs  
8, Rue Jean-Pierre Timbaud  
BP 309  
78054 Saint-Quentin-en-  
Yvelines Cedex  
France  
Tel: (33) 1-30-60-70-00  
Fax: (33) 1-30-60-71-11

---

**Atmel 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

## Product Contact

---

**Web Site**  
[www.atmel.com](http://www.atmel.com)

**Technical Support**  
[avr@atmel.com](mailto:avr@atmel.com)

**Sales Contact**  
[www.atmel.com/contacts](http://www.atmel.com/contacts)

**Literature Request**  
[www.atmel.com/literature](http://www.atmel.com/literature)

**Disclaimer:** The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2007 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR®, STK® and others, are the registered trademarks, Z-Link™ and others are trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.