AVR2022: AT86RF231 - Software Programming Model

Features

- **AT86RF231 – programming reference**
- **Message Sequence Charts with code examples**

1 Introduction

The AT86RF231 Software Programming Model (SWPM) shall provide a reference for developers utilizing the radio transceiver AT86RF231 as effective as possible. The model describes the behavior of the radio transceiver and the required programming steps to use the provided hardware features, exemplary shown in Figure 1-1.

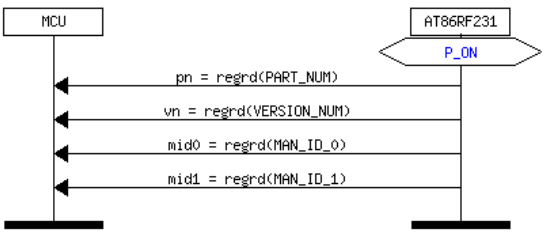
The SWPM is targeted to software developers, who are going to develop low-level drivers for network stacks and network applications, using the AT86RF231 radio transceiver directly. The SWPM describes the hardware usage together with the related timing constraints usually handled by higher layers like IEEE802.15.4™ MAC, ZigBee™, or IP/6LoWPAN network layers.

Figure 1-1. MSC and example for PHY IDENTIFICATION procedure

Identification

This section describes how the radio transceiver can be identified. The table below lists the values for AT86RF231.

RG_PART_NUM	0x03
RG_VERSION_NUM	0x02
RG_MAN_ID_0	0x1F
RG_MAN_ID_1	0x00



Code example

```
pn = trx_reg_read(RG_PART_NUM);
vn = trx_reg_read(RG_VERSION_NUM);
mid0 = trx_reg_read(RG_MAN_ID_0);
mid1 = trx_reg_read(RG_MAN_ID_1);
```

Application Note

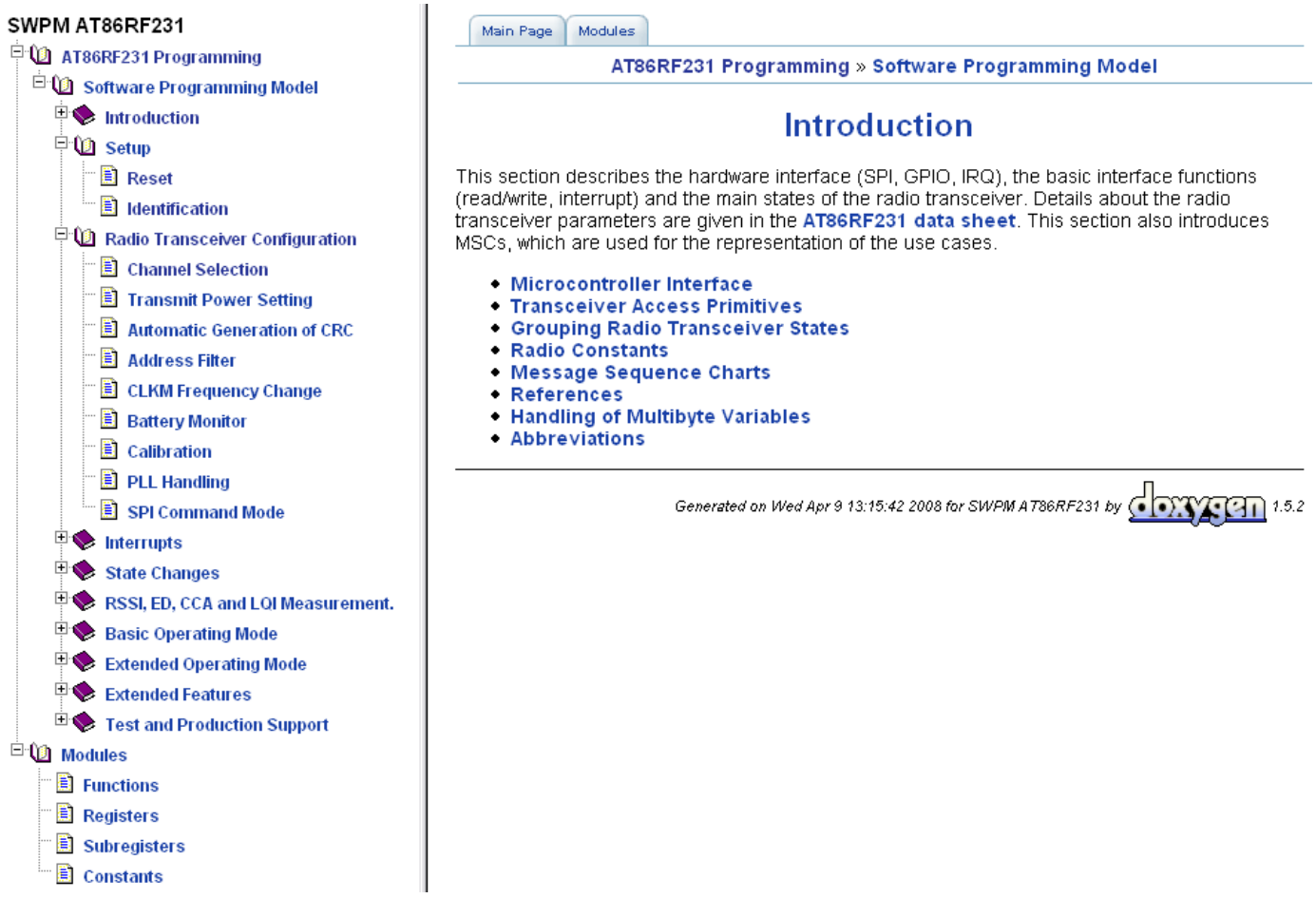
Rev. 8156A-AVR-05/08

2 Functional & Usage Description

The SWPM is primarily a collection of message sequence charts (MSC) that are used to illustrate the behavior and the usage of the AT86RF231 transceiver. MSCs are a method to describe and visualize message handling between different entities. They show events, transactions and the timing relations between them.

The SWPM can serve developers as a reference manual for the daily work with the AT86RF231 transceiver. The SWPM is delivered in HTML format. The HTML files contain a detailed verbal description as well as a graphical representation of the message sequence charts. In addition to the message sequence charts there are basic code examples as illustration of a possible implementation of this specific MSC in a real programming language. The language that is used for these examples is C. The main user interface to the SWPM is the file *index.html* as shown in Figure 2-1.

Figure 2-1. AT86RF231 SWPM – Introduction Page



The navigation is very easy by using the navigation pane on the left side. There are 10 different main categories under the Software Programming Model entry shown in Table 2-1.

Table 2-1. AT86RF231 SWPM Categories

Category Name	Category Description/Summary
Introduction	- Microcontroller interface - Register, Frame Buffer and SRAM access - Radio transceiver states - Radio transceiver constants - MSC basic description
Setup	- Reset - Radio transceiver identification
Configuration	- Channel selection - Transmit power setting - Address filter configuration - …
Interrupts	- Description of interrupts and their handling
State Changes	- Description of radio transceiver state transitions
RSSI, ED, CCA, and LQI	- Measurements of different radio relevant parameters
Basic Operation Mode	- Frame transmit procedure - Frame receive procedure
Extended Operation Mode	- Automated frame transmit procedure (TX_ARET) - Automated frame receive procedure (RX_AACK) - Example configurations
Extended Feature Set	- Hardware security support (AES) - High Data Rate Modes - Antenna Diversity - RX/TX Indicator - RX Frame Time Stamping - Frame Buffer Empty Indicator - …
Test and production support	- Continuous transmission test mode

All timings given in the SWPM are based on the typical timing values specified in the AT86RF231 datasheet [1]. Radio constants are referenced to the datasheet tables 7-1 and 12-4. Note that the actual timings partially depend on external circuitry and thus might be different for a particular hardware implementation.

The MSC concept provides a simplified programming model, which features neither loops nor branches. Therefore, all wait times are specified as a worst-case scenario, taking into account frame transmissions of the largest possible frames, exhausting the maximal number of retries etc. Usually, an actual implementation is not going to wait for a long time in many situations, but it rather will either use interrupts to get notified about the completion status of some operation, or poll the transceiver to identify whether the operation has yielded the desired effect.

Thus, the given C code examples are not meant to be an actual full implementation, they are merely provided to illustrate the case.

Further information can be found in the file *readme.html* in the top-level folder of the ZIP package.

References

[1] AT86RF231, Low Power, 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, and ISM Applications

http://atmel.com/dyn/resources/prod_documents/doc8111.pdf

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

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

8156A-AVR-05/08