Atmel AVR126: ADC of megaAVR in Single Ended Mode

Features

- Up to 10-bit resolution
- Up to 76.9kSPS for Atmel® ATmega88
- Up to 15kSPS at maximum resolution
- Auto triggered and single conversion mode
- Optional left adjustment for ADC result read out
- Sleep mode noise canceller
- Driver source code included for ATmega88
  - ATmega88 ADC in single conversion mode
  - ATmega88 ADC in free running mode by using interrupt
  - ATmega88 ADC measurement driver using ‘Timer0’ as auto trigger source
  - ATmega88 ADC for band gap measurement

1 Introduction

Atmel ATmega devices have a successive approximation Analog-to-Digital Converter (ADC) capable of conversion rates up to 15kSPS with a (at maximum) resolution of 10 bits. It features a flexible multiplexer, which allows the ADC to measure the voltage at multiple single ended input pins and an internal channel from band gap reference in the device. Single ended input channels are referred to ground.

This application note describes the basic functionality of the ADC in Atmel megaAVR® devices in Single ended mode with code examples on ATmega88 to get started. The code examples are written in ‘C’ language and have been tested on the Atmel STK®600 starter kit for functionality.

8-bit Atmel Microcontrollers

Application Note

Rev. 8444A-AVR-10/11

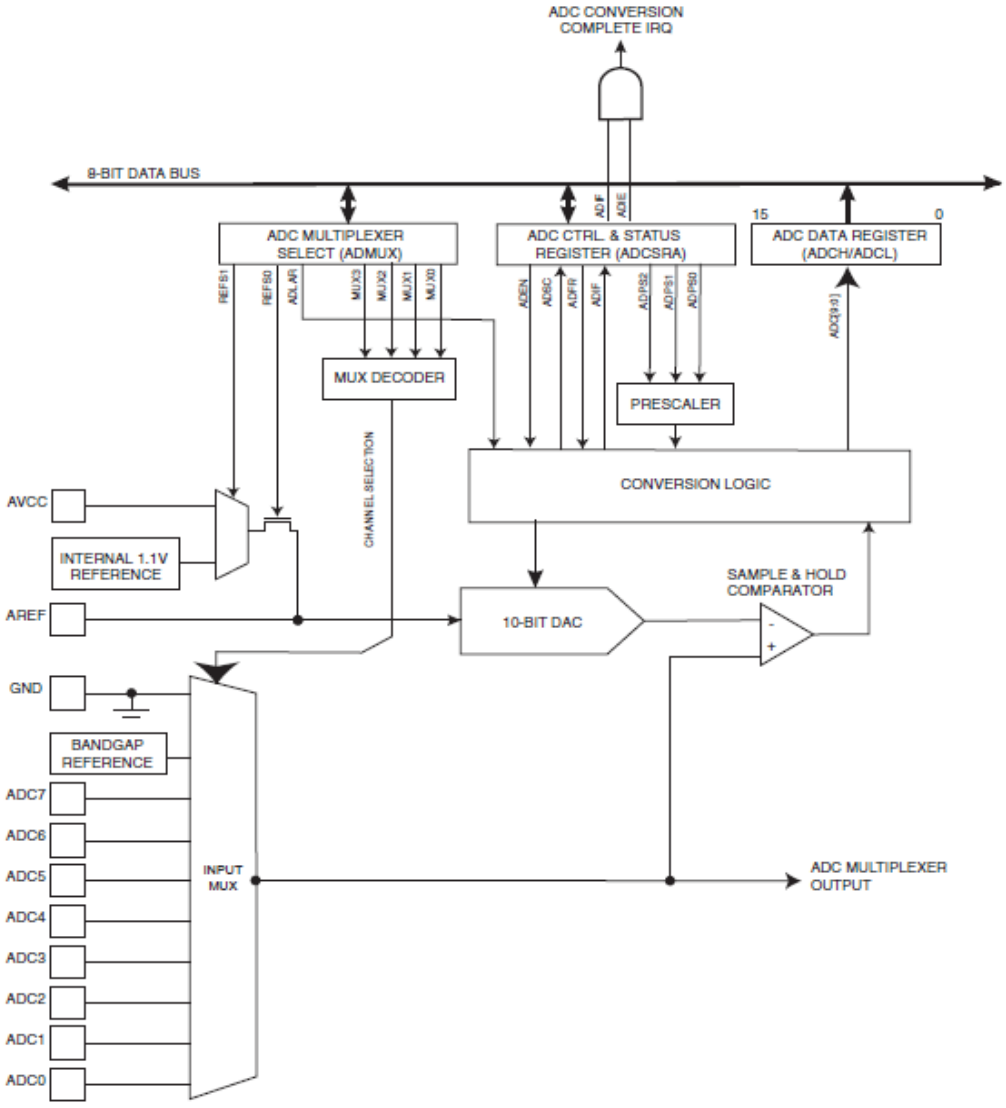
2 Module overview

This section provides an overview of the functionality and basic configuration options of the ADC where as the next section describes the basic steps to configure and run the ADC module with register description details.

2.1 ADC operation

The ADC module converts the analog input voltage to a 10-bit digital value. The minimum value represents GND and maximum value denotes the reference voltage used.

Figure 2-1. Analog to digital converter block schematic operation.



To make use of the ADC, the Power reduction bit (PRADC) in the Power Reduction Register (PRR) must be disabled. This is done by clearing the PRADC bit. To enable

the ADC, ADEN bit in the ADCSRA register must be set. The ADC module (ADEN bit in ADCSRA register) must be disabled before disabling in PRR.

The reference voltage is chosen by the REFS1:0 bits in the ADMUX register. The possible reference voltages are internal 1.1V or internal AV_{CC} or external reference (AREF) pin.

The analog input channel for conversion is selected by the MUXn bits in the ADMUX register. This includes the ADC input pins, internal voltage from fixed band gap reference voltage, ground. The channels selected for conversion will not go into effect until this ADEN bit is set.

Before entering Sleep mode, the ADC module can be disabled by clearing ADEN bit. This reduces the power consumption caused by ADC.

The ADC's 10-bit digital value after conversion is stored in ADCH and ADCL result registers. The ADCH holds the higher byte and ADCL holds the lower byte. Optionally left adjustment of the result can be done by setting the ADLAR bit in the ADMUX register if required.

If ADLAR is enabled and the application needs only eight bit accuracy then it is sufficient to read the ADCH. Otherwise ADCL must be read first followed by ADCH, to ensure that the content of the data registers belong to the same conversion. Access to ADC is blocked, once ADCL is read. It is re-enabled only after ADCH is read.

The ADC module has one interrupt source which can be triggered when conversion completes. If an interrupt occurs between reading ADCL and ADCH, it will get triggered and the result will be lost.

2.2 Input sources

The input sources for the ADC are the analog voltage inputs that the ADC can measure and convert. Two types of measurements can be selected.

- Single ended input
- Internal input

2.2.1 Single ended input

For single ended measurements all analog input pins can be used as inputs. All single ended channels are referred to GND. The analog input voltages cannot be more than the reference voltage selected for ADC.

2.2.2 Internal inputs

The internal band gap analog voltage or ground signal can be selected as input and measured by the ADC. This band gap voltage is an accurate voltage reference inside the microcontroller which is the source for internal voltage reference.

2.3 Starting a conversion

In single conversion mode the ADSC bit in the ADCSRA register must be written a logical one to start the ADC conversion. This bit remains at high logic while conversion is in progress and is cleared by the hardware, once the conversion is complete.

In auto triggered mode, conversion is triggered automatically by various sources. To enable auto triggering, the ADATE bit in the ADCSRA register must be set. The

source of trigger can be selected with the help of ADC Trigger Select bits (ADTS2:0) in ADCSRB register. The auto triggering mode provides a method of starting conversions at fixed intervals which is configurable based on the triggering source.

The interrupt flag will be set even if the specific interrupt or global interrupts are disabled. Thus a conversion can be triggered using ADIF flag without causing an interrupt. Note that ADIF must be cleared manually in order to trigger at next interrupt event in auto triggered mode.

If the ADC runs in free running mode, in which next conversion is triggered once the previous conversion completes. For free running mode, the ADC will perform successive conversions independent of whether the ADIF is cleared or not. The first conversion must be started by setting ADSC bit.

2.4 ADC clock and conversion timing

The ADC can prescale the system clock to provide an ADC clock that is between 50kHz and 200kHz to get maximum resolution. If ADC resolution of less than 10 bits required, then the ADC clock frequency can be higher than 200kHz. At 1MHz it is possible to achieve eight bits of resolution maximum.

The pre-scalar value is selected with ADPS bits (ADPS2:0) in ADCSRA. When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry. When band gap voltage is used an input to ADC then it will take certain time for the voltage to stabilize.

2.5 Changing channel or reference selection

Bits MUXn and REFS1:0 bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. If auto triggering is used, then ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared
- During conversion, minimum one ADC clock cycle after the trigger event
- After a conversion, before the Interrupt Flag used as trigger source is cleared

In these ways, the new settings will affect the next ADC conversion.

In single conversion mode, the channel must be selected before starting the conversion. The channel can be changed one clock cycle after setting ADSC bit, but it is better to wait for the conversion to complete before changing the channel.

In Free Running mode, select the channel before starting the first conversion. It can be changed one clock cycle after setting the ADSC bit but it is better to wait for the conversion to complete while changing the channel. Since the next conversion has already started automatically, the changes will be reflected in the subsequent conversion.

2.6 ADC noise canceller

The ATmega ADC has a noise canceller that enables conversion during sleep mode which reduces the noise induced from CPU core and other I/O peripherals. This feature is available in ADC noise reduction and idle mode. To use this feature the following procedure has to be followed:

- Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC conversion complete interrupt must be enabled
- Enter ADC noise reduction mode (or idle mode). The ADC will start a conversion once the CPU has been halted
- If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC conversion complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC conversion complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed

2.7 Conversion result

Once the ADC completes conversion (ADIF flag will be set) then the 10-bit result will be available in the ADCH and ADCL registers.

For single conversion, the result is:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} represents analog input voltage and V_{REF} represents the selected reference voltage. 0x000 represents GND and 0x3FF represents the reference voltage minus one LSB.

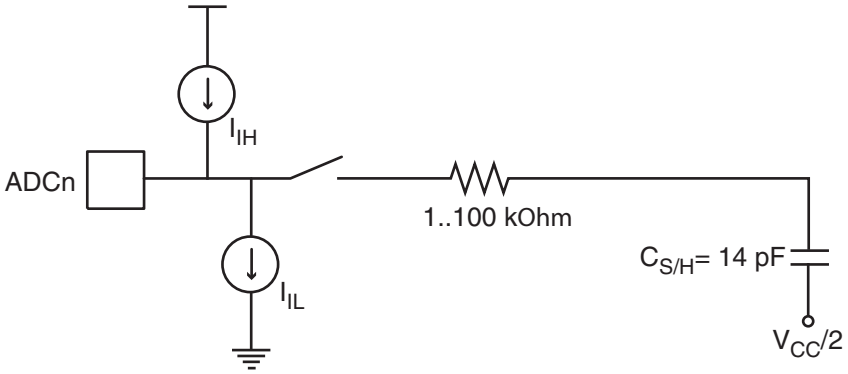
2.8 Analog input circuitry

The analog input circuitry for single ended channels is shown in Figure 2-2. An analog source applied to ADC input pin is subjected to pin capacitance and input leakage of that pin even if is not selected as input for ADC. When particular channel is selected, the source must drive the S/H capacitor through the series (combined) resistance in the input path.

The ADC module is optimized for analog signals with an output impedance of 10kΩ or less. If such a source is used then the sampling time will be negligible.

If the source impedance is higher than 10kΩ then the time taken to charge the capacitor will increase (which can vary widely) and may produce inaccurate results. For example if we use a voltage divider at the ADC input using a resistor network make sure that the source impedance is less than 10kΩ. The low impedance must be used for slowly varying signals since this minimizes the time for charge transfer. It is recommended to remove the frequency components higher than Nyquist frequency ($f_{ADC}/2$) with a low pass filter (to avoid distortion from unpredictable signal convolution).

Figure 2-2. Analog input circuitry.



2.9 Best practices for improving accuracy

The accuracy of ADC depends on the quality of the input signals and power supplies. The following points should be taken into consideration to improve the accuracy of the ADC measurements.

- Understand the ADC, its features and how they are intended to be used
- Understand the application requirements
- It is important to take great care when designing the analog signal paths like analog reference (V_{REF}) and analog power supply (AV_{CC}). The AV_{CC} is supply voltage to analog modules such as comparator, ADC
- Filtering should be used if the analog power supply (AV_{CC}) is connected to digital power supply. That is, the AV_{CC} pin on the device should be connected to the digital V_{CC} voltage via an LC network. For more information on this LC network connection and the L, C values (10µH, 100nF) please refer the respective mega device datasheet. Please note that the AV_{CC} and V_{CC} are internally connected by high impedance networks
- Please note that the AV_{CC} must not differ more than ±0.3V from V_{CC} voltage for mega devices
- The reference voltage can be made more immune to noise by connecting a capacitor between AREF pin and ground
- Keep analog signal paths as short as possible. It is also important that the impedance on the PCB tracks for the ADC channels are not big which will result in too long charge period of the capacitor for an ADC measurement
- Make sure analog tracks run over the analog ground plane
- Avoid having the analog signal path close to a digital signal path with high switching noise (that is, communication lines and clock signals)
- Consider decoupling of the analog signal between signal input and ground for single-ended inputs
- For differential signals the decoupling has to be between the positive and negative input. The decoupling capacitor value depends on the input signal. If the signals are switching fast, the decoupling capacitor has to be lower
- Please ensure that the source impedance is not too high when compared to the sampling rate. If source impedance is too high, the internal sampling capacitor will not be charged to the correct level and the result will not be accurate
- Try to toggle as few pins as possible while the ADC is converting, to avoid switching noise internally and on the power supply. The ADC is especially more

sensitive to switching the I/O pins that are powered by the analog power supply (**PORTC**)

- Disable the digital input on the corresponding ADC channel to minimize the power consumption
- Switch off the unused peripherals by setting the respective bit in PRR register to eliminate any noise from unused peripherals
- Apply offset and gain calibration to the measurement to improve the accuracy
- Put the device in “ADC Noise Reduction” mode to get more accurate results with ADC
- Wait until the ADC, reference or sources are stabilized before sampling, as some sources (for example, band gap) need time to stabilize when they are selected as ADC input channel
- Use over-sampling to increase resolution and eliminate random noise
- In free running mode we have to select the channel before starting the first conversion
- It is recommended to discard the first conversion result (like whenever there is a change in ADC configuration like voltage reference / ADC channel change)
- Please note when switching to a differential channel (with gain settings), the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. So it is better to discard the first sample result
- The linear interpolation methods such as one point (offset) calibration and two point (offset and gain) calibration method can be used based on the application needs

3 Getting started

This section walks you through the basic steps for getting up and running with simple conversion and experimenting with MUX settings. The necessary registers are described along with relevant bit settings.

3.1 Single conversion mode

Task: Single Conversion on ADC channel 0.

In this program the' initialize ()' routine is used to initialize the ADC module. The 'convert ()' routine has to be called whenever the application needs an ADC conversion.

1. Set the MUX bit fields (MUX3:0) in ADC's MUX register (ADMUX) equal to 0000 to select ADC Channel 0.
2. Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module.
3. Set the ADC Pre-scalar bit fields (ADPS2:0) in ADCSRA equal to 100 to prescale the system clock by 16.
4. Set the Voltage Reference bit fields (REFS1:0) in ADMUX equal to 11 to select Internal 1.1V reference.
5. Set the Start Conversion bit (ADSC) in ADCSRA to start a single conversion.
6. Poll (wait) for the Interrupt Flag (ADIF) bit in the ADCSRA register to be set, indicating that a new conversion is completed.
7. Once the conversion is over (ADIF bit becomes high) then read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value.

3.2 Free running mode by using interrupt

Task: Free running conversion on ADC channel 0 by using interrupt.

In this program the' initialize ()' routine is used to initialize the ADC module. The 'ADC ISR' will read the result as soon as conversion is completed. Please note that the time between two consecutive ADC samples depends on the ADC conversion time (continuous sampling and conversion in free running mode).

1. Repeat the steps 1-4 from Section 3.1.
2. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC interrupt.
3. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode. By default, the Auto Trigger Source bit fields (ADTS2:0) in ADC Control and Status Register B (ADCSRB) is set to 000, which represents Free-running mode.
4. Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion.
5. Once the conversion is over (ADIF bit becomes high) then (program control will enter into ADC interrupt service routine where we) read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value (continuously).

3.3 Auto trigged mode (for periodic ADC sample measurements)

Task: Auto triggered conversion on ADC channel 0 by using Timer as trigger source.

In this program the 'initialize_ADC ()' routine is used to initialize the ADC module. The; 'initialize_Timer ()' routine configures the Timer 0 for compare A match event. The 'ADC ISR' will read the result as soon as conversion is completed. The ADC

module will start (ADSC bit) the conversion whenever the timer reaches its compare match value (that is, every 10 milliseconds in this example).

Please note that by changing the 'OCR0A' compare register and timer0 clock prescalar bits in 'TCCR0B' register we can change the conversion interval (from 10 milliseconds) as per our application needs.

1. Repeat the steps 1-4 from Section 3.1.
2. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC interrupt.
3. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode.
4. Set the Auto Trigger Source bit fields (ADTS2:0) in ADC Control and Status Register B (ADCSRB) to 011 to use the Timer0 Compare Match A event as ADC start flag trigger (set) event.
5. Set the Waveform Generation Mode bit fields (WGM1:0) in TCCR0A to '10' (Clear Timer on Compare Match Mode).
6. Set the Timer0 Clock Select bit fields (CS2:0) in TCCR0B to '011" (ClkIO/64 -> Prescalar).
7. Set the Output Compare register A to the desired value (in this example 156) so that ADC will convert the analog value at every 10 milliseconds (compare match event) interval.
8. Once the conversion is over (ADIF bit becomes high) then (program control will enter into ADC interrupt service routine where we) read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value (whenever the Timer0 reaches the Compare A register value).

3.4 Band gap measurement

Task: Measure the band gap reference from internal band gap reference channel.

The 'initialize ()' routine is used to initialize the ADC module. The 'convert ()' routine has to be called whenever the application needs an ADC conversion. The 'MeasureGND ()' routine measures the ground value. The 'MeasureCurrentBGsingle ()' routine measures the internal band gap reference voltage.

1. Repeat the steps 1-3 from Section 3.1.
2. Set the Voltage Reference bit fields (REFS1:0) in ADMUX equal to 01 to select AV_{CC} as ADC voltage reference.
3. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC interrupt.
4. Keep measuring input on ADC Channel 0, until the switch is pressed. If the switch is pressed, configure 0V (GND) in ADC by setting MUX bit fields (MUX3:0) equal to 1111. This is done to discharge the capacitor of ADC.
5. While measuring the ground the ADC interrupt and auto trigger (free running) mode is disabled.
6. Set the Start Conversion bit (ADSC) in ADCSRA to start the conversion.
7. Once the conversion is over (ADIF bit becomes high) then read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value (for ground).
8. After 70µs, measure the band gap reference value by configuring MUX bit field (MUX3:0) equal to 1110. It should be measured after 70µs delay, because of the time taken to charge the capacitor of ADC.
9. While measuring the band gap voltage the ADC interrupt and auto trigger free running is disabled.
10. Set the Start Conversion bit (ADSC) in ADCSRA to start the conversion.
11. Once the conversion is over (ADIF bit becomes high) then read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value (band gap voltage).

4 Device datasheet reference

Please note that though this application notes describes the Atmel ATmega88's ADC module the block wise operation of on chip ADC on all mega devices is almost similar.

Please refer the respective device datasheet for any additional (different) feature wise support. For example some mega device variants (like Atmel ATmega48P) will have on-chip temperature sensor as one of the ADC channel measurement to measure the temperature, some devices also feature differential ADC channels with configurable gains such as 10,200 (like Atmel ATmega2560), some devices also have 2.56 internal voltage reference (like ATmega2560), some devices support high ADC speed such as 125Ksps (like Atmel AT90PWM1) etc.

Please refer to Section "Analog to Digital Converter" for detailed information such as ADC conversion timing diagrams etc.

Please refer to Section "Electrical Characteristics ->ADC Characteristics", "Electrical Characteristics -> System and Reset Characteristics" for ADC related characterization data's such as INL, DNL, absolute accuracy, conversion time, offset error, gain error, input resistance, band gap reference start up time etc.

Please refer to Section "Errata" to find out whether any known ADC related errata is there for that device. If any then please ensure that the work around has been implemented.

5 Driver implementation

This application note includes a source code package with a basic ADC driver implemented in C. It is written for AVR Studio® 5 IDE with AVR® tool chain. Please note that this ADC driver is not intended for use with high-performance code. It is designed as a library to get started with the ADC.

1. Atmel ATmega88 ADC in single conversion mode.
2. ATmega88 ADC in free running mode by using interrupt.
3. ATmega88 ADC measurement driver using ‘Timer0’ as auto trigger source.
4. ATmega88 ADC for band gap measurement.

6 Recommended reading

It is recommended to read the following application notes to get an overall idea on ADC.

- AVR042: AVR Hardware Design Considerations – This application note covers most of the problems encountered with power supply design and other physical design problems
- AVR120: Characterization and Calibration of the ADC on an AVR – This application note explains various ADC characterization parameters which will affect on ADC measurements and also describes how to measure and perform run-time compensation for these parameters such as offset error, gain error
- AVR121: Enhancing ADC resolution by over sampling - This application note explains the method called "Over sampling and Decimation" and which conditions need to be fulfilled to make this method work properly to achieve a higher resolution without using an external ADC
- AVR122: Calibration of the tinyAVR's internal temperature reference – This application note describes how to calibrate and compensate the temperature measurements from the Atmel ATtiny25/45/85. It can also be used on other Atmel AVR microcontrollers which have on chip internal temperature sensors
- AVR125: ADC of tinyAVR in single ended mode – This application note describes the basic functionality of the ADC in Atmel tinyAVR® devices in single ended mode with code examples on Atmel ATtiny88 to get started
- AVR127: Understanding ADC parameters – This application note describes the basic concepts of ADC and the various ADC parameters (such as INL, DNL etc) which will determine the characteristics (accuracy) of the ADC

7 Table of contents

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chou-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3523-3551
Fax: (+81) 3523-7581

8444A-AVR-10/11