Passive InfraRed Reference Design (PIRRD) for SAM3S Motion Detector Camera

1. Scope
This application note provides detailed directions and instructions to both hardware and software engineers building a low cost, yet powerful, Passive InfraRed Motion Detector Camera as found in building or home alarm and monitoring systems. The reference design is based on the SAM3S4C device.

2. Associated Documentation and Software
Before going further into this document, refer to the latest documents for the corresponding device and associated tools on the Atmel web site.

3. Terminology, Abbreviations and Typographical Conventions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR</td>
<td>Passive InfraRed</td>
</tr>
<tr>
<td>IR</td>
<td>Infra Red</td>
</tr>
<tr>
<td>PDC</td>
<td>Peripheral DMA controller</td>
</tr>
<tr>
<td>ACC</td>
<td>Analog Comparator Controller</td>
</tr>
<tr>
<td>ACP</td>
<td>Analog Comparator</td>
</tr>
<tr>
<td>PIO Controller</td>
<td>Parallel Input/Output Controller</td>
</tr>
<tr>
<td>TWI</td>
<td>Two-Wire Interface</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>BOM</td>
<td>Bill of Materials</td>
</tr>
<tr>
<td>PSRAM</td>
<td>Pseudo Static RAM</td>
</tr>
<tr>
<td>SMC</td>
<td>Static Memory Controller</td>
</tr>
<tr>
<td>FPS</td>
<td>Frame Per Second</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
</tbody>
</table>

4. Keywords
Atmel ARM® Cortex®-M3, PIR, Passive InfraRed, Camera, SAM3S, CMOS image sensor, OmniVision®, low-power, IEEE802.15.4, ZigBee®, Wireless....
5. Overview

PIR Motion Detectors have been present for a considerable time in building, home alarm and monitoring systems markets. A PIR Motion Detector predominantly converts a microwave signal, infra-red light in this case, emitted from a human body, for example, into an electrical signal. A PIR Motion Detector does not detect a body standing still, but in motion only. For further details about PIR Motion Detection, see Section 10.2 “References”.

Having a PIR Motion Detector on its own is not enough in an alarm or monitoring system. Such a system is also equipped with a CMOS imaging sensor to take a snapshot of the moving body in front of the PIR sensor. The acquired image can then be transmitted over wired or wireless medium.

The reference design implemented around Atmel's SAM3S Cortex-M3 based MCU is particularly well suited for this kind of application and provides ready to use hardware and software blocks. The SAM3S PIO Controller integrates an interface able to read data from a CMOS digital image sensor and store images in internal or external memory with the CPU being in sleep mode, thus optimizing power consumption when running from batteries. One important requirement as well, is the image compression process for transmitting images to optimize transmission time and in turn, power consumption. Compared to higher priced CMOS imaging sensor devices with embedded JPEG compression, the SAM3S device is able to perform JPEG compression through software.

Other features, such as low-power analog comparator or analog-to-digital converter for motion detection, low-power modes and PIO state preservation (even in backup mode), simplify the design by reducing the count of external components, board space and therefore, overall system cost.
6. Hardware Description

6.1 Overview

Hardware choices to reduce BOM, consumption and component count are detailed in the chapters that follow.

6.2 PIRRD Board Block Diagram

As shown in the block diagram, the component count is low, thanks to the high integration level of the SAM3S MCU.

Figure 6-1. PIRRD Block Diagram
Figure 6-2. SAM3S PIRRD Board Top Side

1: PIR sensor with Fresnel lens
2: CMOS image sensor
3: RZ600 IEEE 802.15.4 Connector
4: PSRAM Memory
5: Micro AB USB connector
6: JTAG/ICE Debug Connector
7: Pre-cut line for LCD Removal
6.3 Power Supply

The power supply needs for the PIR reference design is quite simple. The MN6 voltage regulator has an adjustable output voltage set at 3V (3V_MCU). Note that the SAM3S can have IO voltage (VDDIO) and core voltage (VDDCORE) as low as 1.62V. 3V is a minimum requirement for the analog supply of the CMOS image sensor. MN6 has a quiescent current of 55 µA typic in active mode. The MN6 power down mode is not used in this reference design. It can be replaced by a better voltage regulator to enhance quiescent current performance. The MN4 is a PolyZen protection circuit that protects against inductive voltage spikes, voltage transients, incorrect power supplies and reverse bias. MN5 is a noise suppression/EMI suppression circuit employed to avoid disturbing analog voltages provided to the CMOS sensor, PIR sensor and reference voltage when powering the system from a wall plug. When powering PIR Motion Detector Camera from batteries, the noise and EMI Suppression circuit can be dispensed with. All current measurements presented in the following paragraphs are done on JP6 jumper.

The board can be directly supplied with 3 x AAA 1.2V NiMH batteries by removing the JP5 jumper and connecting the battery pack.
6.4 Oscillators

Provisions have been made on the schematics and PCB to use an external slow clock crystal oscillator and 12 MHz crystal for the main oscillator. If no USB or accurate slow clock source is needed in the application, external crystal oscillators are not needed since the SAM3S embeds one 32.768 kHz RC oscillator and an accurate (factory trimmed) 4/8/12 MHz fast RC oscillator. The internal PLL can work from the fast RC oscillator.
6.5 PIR Sensor

The PIR sensor (RE200B from Nicera) chosen for the reference design is a general purpose dual element. This model satisfies customer's cost reduction needs, keeping most dual element type performances at reasonable levels.

PIR Sensors need a Fresnel lens to concentrate IR waves (5 µm to 14 µm) and to filter ambient light, see Section 10.2 “References”. A detailed explanation of PIR Sensor mechanical aspects is not discussed in this document. Please refer to the Mechanical section of the RE200B catalog, here: http://www.nicera.ph/images/PDF/pyro_catalog.pdf

The PIR Sensor provides a 3900 mVpp signal (with 80dB gain) when measured with a 420K µW/Cm² black body in front of it. The sensor has a frequency response in the range of 0.3Hz to 5Hz.

The PIR sensor and op-amp power supply are provided by the 3V_MCU rail.

Figure 6-6. PIR Sensor

R42 provides a current voltage converter. The signal is then passed through two active filter stages. Amplifiers provide 80dB gain in total. The two filters formed by R37/C45 - R39/C50 and two others formed by R31/C33 - R34/C39 provide a 1.59 Hz band-pass filter at -3dB with attenuation of -40dB.

Shown below in Figure 6-7, is the amplified and filtered signal. The green signal is at the first amplifier stage output (OUT2 pin) and the yellow one is at the second amplifier stage output (OUT1 pin).
The “1” pulses represent a premier movement detected in one sense, then “2” pulses indicate a second motion detected in the opposite sense. The OUT1 signal is then fed to the Analog-to-Digital Converter (ADC) AD0 (PA17) input. Since the ADC and Analog Comparator (ACP) share analog inputs, motion detection can be done with the ACP or the ADC. Using the ACP provides lower power consumption whereas using the ADC requires less external components and provides better motion detection sensitivity.
6.5.1 Using the Analog Comparator (ACP) for Motion Detection

The ACP inputs are shown below in Figure 6-8. Analog Comparator Diagram. ADC and ACP share common inputs comprised of 8 multiplexed inputs for INP (positive input) and 4 for INN (negative input). INN and INP can be swapped by software in the ACP controller (ACC).

Figure 6-8. Analog Comparator Diagram

Thanks to the swap feature, the ACP can be used as a windowed comparator with R60, R61 and R62 making a high (1.7V to PB0/AD4 pin) and low (1.3V to PB1/AD5 pin) threshold to INN input. More details are given in Section 7.3.2 “PIR Sensor”. The sensitivity of the detection can be adjusted by the threshold level.

6.5.2 Using the 12-bit ADC for Motion Detection

If reducing board space is needed (and minor cost reduction as well), a 12-bit ADC with embedded programmable gain amplifier can be used. In this case, only the first stage amplifier with a 60dB gain (1000) is needed. The internal PGA provides a gain of 4 to achieve a 72dB gain (4000) in total. Only one stage filter can be used, as well as anti-aliasing filter to then perform digital filtering if needed. Using the ADC assures better motion detection sensitivity. The counterpoint to using an ADC is an increase in power consumption. Motion detection with the ADC is not presented in detail in this document.

6.6 CMOS Image Sensor

The CMOS image sensor, OV7740 from OmniVision, used in this reference design is a low-complexity, low-cost, low-power, yet powerful sensor. It supports VGA (640x480) at up to 60 fps and QVGA (320x240) at up to 120 fps. The OV7740 has standard interface output pins such as data, horizontal/vertical synchronization signals, pixel clock output and main clock input. Interfacing between the CMOS image sensor and the SAM3S device is made easy by means of the sensor interface (parallel capture mode). The SAM3S is able to sample data from the CMOS image sensor without CPU intervention and transfer image data into internal or external memory.
As seen in the schematic above, the OV7740 requires three different supplies:

- **DVDD** for digital,
- **DOVDD** for IO,
- **AVDD** for analog

The following supplies are provided.

- Digital (core) rail, DVDD, ranging from 1.70V to 3.5V generated from 3V_OVT
  - 1.5V chosen (MN11 regulator) (1.5V internal regulator of OV7740 not used)
- Analog rail, AVDD, ranging from 3.0V to 3.6V generated from 3V_OVT
  - 3V chosen
- Input/Output buffer rail, DOVDD, ranging from 3.0V to 3.6V
  - 3V chosen

In order to manage power consumption, 3V_OVT is provided from the 3V_MCU rail, controlled by Q3 and SAM3S PC16 pin to switch ON and OFF the CMOS image sensor. The startup time when setting PC16 at level one is 0.4ms for MN11. The OV7740 can receive configuration data 1ms after its power supplies are stable. Data and command settings are taken into account after 5 ms.

The CMOS image sensor is used in 8-bit mode, and gets its clock from a Programmable Clock Output (PCKO) coming from the SAM3S, which avoids using an external oscillator to provide a clock to the CMOS image sensor. The XVCLK1 (main input clock) ranges from 6 MHz to 27 MHz, 24 MHz typical. If using internal SRAM as a memory buffer (32-bit zero wait state memory), the pixel clock from the sensor (PCLK) connected to PIODCCLK (PA23) can be equal to 2 times the PIO master clock (PCLK < MCK/2). This is the raw transfer rate of the PIO parallel capture mode. In the reference design, when taking into account the PSRAM access time of 55 ns, PCLK must be inferior or equal to MCK/4 when in color mode and inferior or equal to MCK/3 for black and white mode.
Thanks to SAM3S on-die termination (ODT) series resistors, no external ones are needed for clock signals (XVCLK1, PCLK) and data lines. The series resistor helps to reduce I/Os switching current (di/dt) thereby reducing in turn, EMI. It also decreases overshoot and undershoot (ringing) due to inductance of interconnection between devices. ODT helps diminish signal integrity issues and reduce cost and board space.

Configuration of the sensor is done via the Two-Wire Interface (I2C compatible) SIOC (PA4) and SIOD (PA3) pins.

### 6.7 Memory

To store one or more 320x240 images from the sensor, a buffer is needed. An external PSRAM memory is connected to the static memory controller of the SAM3S. This choice is due to the lower cost per bytes compared to SRAM. As shown in Figure 6-10 below, 16-bit x 512K words is used as 8-bit PSRAM, giving a 512 KBytes memory size. The choice of 16-bit PSRAM over 8-bit PSRAM is due to easier 16-bit availability.

**Figure 6-10.** External Memory

The PSRAM power supply can be switched OFF and ON by means of Q2 and PC9 pin. This is to save power consumption while there is no image to acquire.

Below, Figure 6-11 shows how to address the whole 1 Mbyte memory size.
Figure 6-11. External Memory (addressing all the memory size)

6.8 Wireless interface

Provision has been made on the board for the Atmel RZ600 IEEE802.15.4 Module. The RZ600 is a low-power module allowing wireless transmission to a PC or base station of the image taken, for example. The RZ600 can be used with proprietary protocol over 2.4 GHz ISM band link or for implementing IEEE 802.15.4 compliant wireless applications such as ZigBee and 6LoWPAN. For further details about RZ600, visit the Atmel web pages at:


The wireless interface is reserved for future use in this reference design.

Figure 6-12. Connector for Atmel RZ600 IEEE802.15.4 Module

The RZ600 module requires few pins. The main interface for setting RZ600 registers, receiving and transmitting data, is over the SPI port. Interrupt and Sleep (SLP_TR) pins are provided for efficient power management.

6.9 JTAG Connector, TFT Display/Resistive Touch Screen and USB

The JTAG/SWJ-DP interface, the 16-bit color TFT, USB and the resistive touchscreen controller are for debug and demonstration purpose only. Note that power consumption of the above devices does not affect the current values given in the following paragraphs. As seen in the board’s front side picture, Figure 6-2 on page 4, the board has a pre-cut line used to remove the LCD Side if a wireless link is available.
6.10 General Purpose I/Os Control

The table below gives a summary of the SAM3S PIOs used to control power supply pins of external devices.

Table 6-1. PIO Control Summary

<table>
<thead>
<tr>
<th>Device</th>
<th>Pin</th>
<th>Active Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS Image Sensor</td>
<td>PC16</td>
<td>0</td>
</tr>
<tr>
<td>PSRAM</td>
<td>PC9</td>
<td>0</td>
</tr>
</tbody>
</table>

6.11 Reference Design Schematics


6.12 Printed Circuit Board Considerations

The main focus for the PCB layout is around the CMOS image sensor and the PIR sensor (sensor + amplifier). No high constraints are required but general rules such as placement of decoupling capacitors, power plan distribution, etc., should be followed. As stated in Section 6.6 "CMOS Image Sensor", ODT resistors help reduce PCB design space and assure signal integrity to achieve better PIR detection and image quality.
7. Software Description

7.1 Overview

The software is intended to provide some guidance to use with SAM3S and components on the PIRRD board. It can also be used as a reference design to deliver a final product.

The software is composed of three examples and three demos. The three examples provide basic functions for certain peripherals and components on the board. The first example performs motion detection using the PIR sensor. The second one captures an image with the CMOS Image Sensor. The last one performs JPEG compression by software. The three demos provide reference designs or demonstrations for specific low-power mode available in SAM3S and function of PIR camera as well, which can help users achieve low power consumption and high performance balance for final products.

To get the power consumption of the application, it is sequentially split into several phases and power consumption is measured in each phase as the tables in Section 8. All the measured figures under each phase could be synthesized to get the total power consumption.

All the examples and demos are provided with IAR project files included in the accompanied software package.

7.2 Demonstration Applications

Described below are three demonstration applications for the reference design:

- Single Snapshot Mode
- Periodic Wake-up Mode with Motion Detection Mode
- Continuous Capture Mode

These three applications (demos) are provided for use under particular circumstances. These applications integrate several basic functions to accomplish a special task in distinct power modes.

Further, information is provided on MCU initialization, PIR sensor, CMOS image sensor, JPEG compression.

7.2.1 Single Snapshot Mode

This mode demonstrates CMOS image sensor capture through the SAM3S device backup mode. SAM3S wake-up input pins wake up the system when motion is detected. In this reference design, this is emulated by the on-board push button. This mode is based on the Backup mode as defined in the SAM3S series datasheet.

The current in backup mode can be as low as 21 µA and the wake-up time is around 856 µs. For more detailed data on power consumption and duration, see the tables in Section 8. “Power Consumption and Battery Life Time”.

The detailed procedure is provided below in Figure 7-1. Flowchart of Single Snapshot Mode.
7.2.2 Periodic Wake-up with Motion Detection Mode

In this application, to minimize power consumption, the SAM3S (which drives the PIR sensor) is configured in wait mode. In this mode, the SAM3S needs to wake itself up and perform “analog acquisition” (ACP or ADC) to detect a motion from the PIR sensor. If motion is detected, the CMOS sensor will capture an image. The demo uses wait mode as defined in the SAM3S series datasheet.

For more detailed data of power consumption and duration, see Table 8-1 and Table 8-2 in Section 8. “Power Consumption and Battery Life Time”.

The detailed procedure is provided below in Figure 7-2. Flowchart for Periodic Wake-up with Motion Detection.
Figure 7-2. Flowchart for Periodic Wake-up with Motion Detection

1. **Startup**
   - Initialize Ms RTT Alarm
     - Enter Wait Mode
       - Alarmed
         - Enter Active Mode without PLL
           - Initialize Motion Detect and N ms Time-out Interval
             - Enter Sleep Mode
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
             - Enter Active Mode without PLL
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
             - Enter Active Mode without PLL
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
             - Enter Active Mode without PLL
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
             - Enter Active Mode without PLL
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
               - Enter Sleep Mode
                 - Motion Detected or Time-out Interval Expired
                   - Enter Active Mode without PLL
                     - Is Motion Detected and Is Interval Expired
                       - Y
                         - Enter Active Mode with PLL Enabled
                           - Capture
                             - Display
                       - N
             - Enter Active Mode without PLL
               - Motion Detected or Time-out Interval Expired
                 - Enter Active Mode without PLL
                   - Is Motion Detected and Is Interval Expired
                     - Y
                       - Enter Active Mode with PLL Enabled
                         - Capture
                           - Display
                     - N
   - Enter Wait Mode
7.2.3 Continuous Capture Mode

This mode is intended to improve frame rate performance when capturing images and displaying them onto the LCD. Only the frame rate and the total average current are given in this mode. The push button is used to switch between color and B&W mode. B&W mode can achieve better frame rate performance. More details are given below in Figure 7-3. Flowchart for Continuous Capture Mode.

Figure 7-3. Flowchart for Continuous Capture Mode

7.3 Code Implementation

7.3.1 MCU Initialization

For demos, it is necessary to perform certain initialization tasks to meet various requirements, such as low-power mode, clock configuration, and initialization of some application-specific components.

7.3.1.1 PIO Configuration for Low-power Mode

For low-power consumption, the external components should be shut down while the system is in backup mode or wait mode. This can be achieved by configuring PIOs high to enable the PMOS cut-off switch. See, Section 6.10 “General Purpose I/Os Control”

To avoid leakage, configure low (with pull-up disabled) unused I/Os connected to pad from inactive external devices.

Configure unused I/Os which are connected from pad to active devices, depending on the external circuit. For example, PC12 connected to LED can be configured as an input with pull-up resistor enabled.
The following code demonstrates setting PC12 as an input with pull-up resistor enabled. All definitions such as: PIOC, PIO_ODR etc., can be found in the device header file, for example, SAM3S4.h for the SAM3S4 device.

```c
/* Disable the output on PC12 */
PIOC->PIO_ODR = (1<<12);
/* Enable PIO to control PC12 */
PIOC->PIO_PER = (1<<12);
/* Enable the pull up resistor on PC12 */
PIOC->PIO_PUER = (1<<12);
/* Disable the pull down resistor on PC12 */
PIOC->PIO_PPDDR = (1<<12);
```

### 7.3.1.2 Clock for Maximum Frequency

The main clock source is the internal fast RC Oscillator. By default, its frequency is 4 MHz. To support 12 MHz output, the 3-bit MOSCRCF field in PMC Clock Generator Main Oscillator Register can be set to enable it.

The following code configures the fast RC oscillator at 12 MHz:

```c
/* Enable Fast RC oscillator at 12Mhz. */
if( (PMC->CKGR_MOR & CKGR_MOR_MOSCRCF) != (0x2 << 4)) {
   PMC->CKGR_MOR = (0x37 << 16) | CKGR_MOR_MOSCRCEN | (0x2 << 4);
   /* Wait the Fast RC to stabilize */
   while( !(PMC->PMC_SR & PMC_SR_MOSCRCS) ) ;
}
```

To enable maximum 64 MHz for master clock and processor clock, PLLA is activated. For detailed setting, check the accompanying software for reference. A piece of code switching MCK to PLLA with pre-scaler is as follows (timeout is a predefined 32-bit integer variable). Pay attention to the sequence of setting MCKR, which differs from case to case. A detailed description is in the PMC part of the datasheet:

```c
/* Initialize PLLA */
PMC->CKGR_PLLAR = ((1 << 29) | (0x1f << CKGR_MUL_SHIFT) \ 
   | (0x1 << CKGR_PLLCOUNT_SHIFT) \ 
   | (0x3 << CKGR_DIV_SHIFT));
while( !(PMC->PMC_SR & PMC_SR_MCKRDY) ) ;

/* set pres */
PMC->PMC_MCKR = (PMC->PMC_MCKR & (uint32_t)~PMC_MCKR_PRES) | PMC_MCKR_PRES_CLK_2;
for ( timeout = 0; !(PMC->PMC_SR & PMC_SR_MCKRDY) && (timeout++ < CLOCK_TIMEOUT) ; ) ;

/* Switch to PLLA */
```
7.3.1.3 **PSRAM Initialization**

As SRAM size (48 Kbytes on SAM3S4 devices) is not sufficient to store memory for capture buffer, image frame buffer, etc., PSRAM is applied for memory usage as large as several hundred kilo bytes.

The PSRAM is driven by the SMC (Static Memory Controller), the user interface of which contains several registers to configure the timing and mode for PSRAM. See the SAM3S series datasheet for SMC registers bit-field descriptions.

To enable a clock for SMC, the interface below can be called, where ID_SMC is the peripheral identifier for SMC.

```c
/* Enable peripheral clock */
PMC_EnablePeripheral( ID_SMC ) ;
```

The SMC in SAM3S has 4 chip selects. The user needs to configure the respective one. The chip select number can be indexed in the SMC structure defined in the device header file. For example,

```c
SMC->SMC_CS_NUMBER[0] is related to chip select 0 which is used for PSRAM on SAM3S-PIRRD.
```

The full sample code is provided in the package: libboard_sam3s-pirrd library.

7.3.2 **PIR Sensor**

The PIR sensor is accompanied by an Analog Comparator to provide motion detection. A reference voltage is used to determine the sensitivity of the detector.

7.3.2.1 **Motion Detection Using the Analog Comparator and PIR Sensor**

The PIR sensor senses the energy change while a living body moves across its field of view. This change generates voltage deviation from the static output. The deviation varies from case to case, as determined by the distance from the sensor and the body’s temperature.

The analog comparator compares two input voltages and gives the result. Two signals can be applied to the inputs of the ACC (analog comparator controller). One is the output of the PIR sensor and the other is a reference voltage provided by a resistor. The reference can be adjusted depending on the need.

7.3.2.2 **ACC Initialization**

For the ACC implementation in SAM3S devices, the clock should be enabled first through the PMC interface. The two inputs are selected through the ACC mode register. Edge type for comparing output can be configured too. The sample code of ACC initialization is provided by pyrosensor_motion_detect example.

The chip library contains the interface operating on ACC. The most important one is `ACC_configure`. Its definition follows:
extern void ACC_Configure( Acc *pAcc, uint8_t idAcc, uint8_t ucSelplus,
uint8_t ucSelminus, uint16_t wAc_en, uint16_t wEdge, uint16_t wInvert )

where ucSelplus and ucSelminus are the paired inputs and wEdge is the edge type.

7.3.2.3 ACC Comparison Interrupt Handler

The ACC comparison interrupt is enabled through the ACC Interrupt Enable Register and checked through the ACC Interrupt Status Register. The flag, SCO (Synchronized Comparator Output) in the ACC Interrupt Status Register is used to check the comparison result of two inputs.

If the comparison edge interrupt is captured, the system falls into ACC_IrqHandler for interrupt handling. In the handler, SCO is checked and the comparison result is updated as well.

7.3.2.4 Software Example

A PIR Sensor motion detection example is provided in the “examples_pirrd” folder, "pyro_sensor_motion_detect".

7.3.3 CMOS Imaging Sensor

The SAM3S device is capable of capturing parallel data in synchronization with an external clock, which is used for interfacing to a CMOS image sensor.

7.3.3.1 Initialization of PIO Parallel Capture and OV7740

An 8-bit parallel capture mode is available inside the SAM3S PIO controller. It is used to interface a CMOS digital image sensor.

OV7740 is a CMOS image sensor with maximum VGA output, RAW RGB and YUV format, I2C compatible serial interface. For our application, PIO capture mode is used for the clock and data interface and TWI (I2C compatible) for serial controlling interface.

For the detailed description of the PIO parallel capture function and settings, refer to the Parallel Input/Output Controller section in the SAM3S series datasheet. Only a few functions are presented here: B&W mode and color mode setting, master clock/pixel clock and capture size.

7.3.3.2 B&W Mode and Color Mode Setting

For the YUV422 output from the sensor, Y is twice the amount than U and V and interleaved with them. That is, the output is Y: U: Y: V or U: Y: V: Y in sequence. Y is the luminance component in a color space and the only one for the black and white mode. So it’s adequate for the system to only have half-sized samples. This can be obtained by setting HALFS to 1 in PIO_PCMR. The code below enables or disables half sampling, depending on the output mode:

```c
/* B & W mode*/
if(out_mode == OUT_MONO){
  pio->PIO_PCMR &= ~(uint32_t)PIO_PCMR_HALFS;
  pio->PIO_PCMR |= PIO_PCMR_HALFS;
  pio->PIO_PCMR &= ~(uint32_t)PIO_PCMR_FRSTS);
}else if(out_mode == OUT_COLOR){
  /* COLOR mode*/
  pio->PIO_PCMR &= ~(uint32_t)PIO_PCMR_HALFS;
}
```
7.3.3.3 Master Clock/Pixel Clock

The master clock is provided through the Programmable Clock with multiplexed I/Os. The source and pre-scaler are set in PMC Programmable Clock Register. Additionally, the relevant PIO should be configured for working in respective peripheral mode. For example, PB13 can be configured as PCK0 in Peripheral B mode.

The source of PCK can be slow clock, main clock and PLL. For flexible clock output from PCK, PLL is a good choice in combination with the prescaler. In SAM3S-PIRRD, OV7740 works in a wide range of frequencies from 6 MHz to 27 MHz. To support best capture performance, the PIO parallel capture mode should work at its best capability. In the SAM3S device, the PIO controller clock frequency must be at least twice the Pixel Clock frequency. That is, the Pixel Clock should be less than half the frequency of 35 MHz (as listed in I/O /Characteristics table in the Electrical Characteristics section of the SAM3S series datasheet). For example, in the referenced application, the frequencies of Master clock and Pixel clock to and from OV7740 are configured to 16 MHz, which meets the requirements of both OV7740 and PIO capture mode.

See code below to check how it works.

```c
Pmc *pmc = (Pmc*) PMC;
const PIN pinPCK = {PIO_PB13B_PCK0, PIOB, ID_PIOB, PIO_PERIPH_B, PIO_DEFAULT};
/* configure PB13 working in peripheral B*/
PIO_Configure(&pinPCK,1);
/* PLLA is 128MHz so that PCK0 is 128MHz/8 = 16MHz*/
pmc->PMC_PCK[0] = 0x32;
/* Enable PCK0 output*/
pmc->PMC_SCER = PMC_SCER_PCK0;
while(!( pmc->PMC_SCSR & PMC_SCSR_PCK0));
```

7.3.3.4 Captured Size

The captured size is determined by the size of the captured image and mode, color or B&W. In consideration of the word width in a 32-bit machine, it is also affected by the bits in one transfer. The DSIZE bit field in PIO Parallel Capture Mode Register is used for this purpose. For example, if the original size is QVGA and the destination is QVGA in B&W, the captured size should be org_size/2/4 with DSIZE as 2. The org_size is 240*320*2 representing color data in YUV422 of QVGA. Being divided by 2 is because B&W is chosen. Being divided by 4 is because the word width is 32 bits.

7.3.3.5 Parallel Capture Event Handler

The parallel capture event is associated with PDC transfer event. It’s the same as other applications of PDC transfers and used with or without interrupt support.

- With interrupt support: Provides routines for special handling in PIOA_IrqHandler.
- Without interrupt support: Checks PDC flags after transfer size is set and the PDC is enabled as done in image sensor_capture application.

7.3.3.6 Image Quality Improvement

The Auto White Balance (AWB) and Auto Exposure (AE) settings of the OV7740 image sensor are set to auto mode by default, so the image sensor gets good quality of images under different environments. But this increases the tuning time and needs frames before it can get good image quality. In the software, because the image sensor is put into low-power mode and wakes up
from time to time, if the AWB and AE are set to auto mode, the first frame’s quality after wake up will not be perfect.

In order to improve the image quality, the following method is used to improve the image quality. After power up, the AWB and AE of the image sensor are set to auto mode. Before the first capture, the software will wait three seconds for AWB and AE tuning, then the parameters of the AWB and AE settings will be saved in the SAM3S backup registers, and AWB and AE will be set to manual mode. Afterwards, each time before capture, the AWB and AE will be configured using these saved (good) parameters from the backup register. Thus after the wake-up, and thanks to the saved parameters, the quality of the first image is improved. Note that this method will make the first capture after power up longer, because there is a three-second waiting time for the good AWB and AE parameters to settle. Also, this method only improves the image quality if the environment does not change greatly compared with the time that AWB and AE perform auto tuning.

7.3.3.7 Software Example
An image sensor example is provided in the “examples_pirrd” folder, “imagesensor_capture”. See the Section 7.4 “Software Deliveries” on page 24 for more details.

7.3.4 JPEG Compression

7.3.4.1 Introduction to JPEG Compression Example
The jpeg_compression example is designed to provide a powerful method to support JPEG compression from captured YUV data for efficient transfers. An open source code project called Independent JPEG Group (IJG) library is ported and integrated into the example to accomplish the task. After capture and compression, the data for compression time, decompression time, compression ratio, memory and LCD accessing speed would be provided to show the performance.

7.3.4.2 IJG JPEG Compression Library
IJG writes and distributes a widely used free library for JPEG image compression. Compression and decompression library, example code and documents are provided in the software package from IJG.

In order to encode an image into JPEG, the following functions are used:

JpegData_SetSource(): Source buffer pointer and length, which points to the starting of the buffer that is going to be compressed.
JpegData_SetDestination(): Destination buffer pointer and length, which points to the starting of the buffer that is going to store the results.
JpegData_SetDimensions(): Set Image dimensions
JpegData_SetParameters(): To set image quality, input format and compression method
ijg_compress(): Start JPEG encoding

In order to decode JPEG, the following functions are used:

JpegData_SetSource(): Source buffer pointer and length, which points to the starting of the buffer that is going to be decompressed.
JpegData_SetDestination(): Destination buffer pointer and length, which points to the starting of the buffer that is going to store the results.
JpegData_SetDimensions(): Set Image dimensions
JpegData_SetParameters(): To set image quality, input format and compression method
ijg_decompress(): Start JPEG decoding

Note that IJG library supports compression with the input formats, YUV420 or YUV444. Since the image sensor’s output is YUV422, a conversion from YUV422 to YUV444 is needed before using the IJG library without first changing the library.

Once the library has been changed (by Atmel) to support YUV422, it is supported in the delivered software, so conversion of YUV422 to YUV444 is not necessary.

IJG library supports several compression/decompression methods, among which are IFAST and ISLOW. IFAST is faster, but a less accurate method. ISLOW is slower, but a more accurate method. The user can choose between these methods accordingly. For detailed information about IJG library, refer to their web site http://www.ijg.org/.

7.3.4.3 JPEG Compression and its Performance

The jpeg_compression example provides a seamless interface for YUV422 data compression from the image sensor and storing the JPEG file to PSRAM. The dynamic memory for compression objects is located in SRAM to achieve better total performance. The default matrix for bus access to memory is optimized for better performance. The sample code is found in the software package.

Putting the dynamic memory for compression in PSRAM can also be done from software, which offers users another option when the SRAM is not sufficient in their application. Note that a different scatter file should be used. For details, refer to “jpeg_compression” examples contained in “source_pirrd_1.0.zip”. See, Section 7. “Software Description”.

Note that all the performance data is measured under on-board 55 ns PSRAM with the processor clock at 64 MHz.

Performance data is as follows:

<table>
<thead>
<tr>
<th></th>
<th>Compress</th>
<th>Decompress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heap in SRAM</td>
<td>218 ms</td>
<td>181 ms</td>
</tr>
<tr>
<td>Heap in PSRAM</td>
<td>317 ms</td>
<td>398 ms</td>
</tr>
</tbody>
</table>

Note that in software, the JPEG compression uses the captured image from the OV7740, so the performance data can vary due to the environment for capture changing.

Table 7-2. JPEG Memory Consumption

<table>
<thead>
<tr>
<th></th>
<th>Flash (Bytes)</th>
<th>SRAM (Bytes)</th>
<th>PSRAM (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heap in SRAM</td>
<td>81,892 (RO)</td>
<td>47,094 (RW+Stack+Heap)</td>
<td>168,960 (Buffer)</td>
</tr>
<tr>
<td>Heap in PSRAM</td>
<td>81,892 (RO)</td>
<td>5,112 (RW+Stack)</td>
<td>210944 (Buffer+Heap)</td>
</tr>
</tbody>
</table>
7.3.4.4 Using the JPEG Library

A JPEG compression example is provided in the “examples_pirrd” folder, “jpeg_compression” that shows how to use the JPEG library. See Section 7.4 “Software Deliveries” on page 24 for more details.

7.4 Software Deliveries

7.4.1 Contents Overview

The SAM3S-PIRRD software package consists of binaries and source code of 3 basic examples and 3 demos. Files provided with this user application note are:

- bin_pirrd_1.0.zip: Binary Files
- ewarm_pirrd_patch.zip: IAR EWARM 5.50 Patch.
- source_pirrd_1.0.zip: Source code files of the demos and examples

The examples provide basic function of related peripherals, such as motion detection, image sensor capture and JPEG compression.

The demos provide reference designs or demonstrations used in SAM3S low-power mode. Several peripherals are involved to complete a special demo, for example, the system is periodically woke up from wait mode and takes pictures as long as motion is detected.

Examples:

- imagesensor_capture
- jpeg_compression
- pyrosensor_motion_detect

Demos:

- Continuous_Capture_Mode
- Periodic_Motion_Detect_Mode
- Single_Snapshot_Mode

7.4.2 Using the Source Code

To use the source code for debugging or modification, unzip “source” archive. The directory structure is as follows (only folders are shown here):

- demo_pirrd
  - Continuous_Capture_Mode
  - Periodic_Motion_Detec_mode
  - Single_Snapshot_Mode
- examples_pirrd
  - imagesensor_capture
  - jpeg_compression
  - pyrosensor_motion_detect
- libraries
  - libboard_sam3s-pirrd
Before opening the IAR project, patch IAR tools for SAM3S device support, if missing. The related archive is ewarm_pirrd_patch.zip.

Each example and demo is provided with an IAR project file, found in “build\ewarm_550” under each example or demo folder. For example, to see the code for Single_Snapshot_Mode in IAR EWARM, go to: “Single_Snapshot_Mode\build\ewarm_550” then open “single_snapshot_mode.eww”.

The supported version of IAR is 5.50.

7.4.3 Compiling Options

The following macros need to be customized for different purposes, especially for demos such as Single Snapshot Mode and Periodic Wake-up with Motion Detection Mode. These definitions are found at the very beginning of the “local definition” part of the main.c file.

/*LCD displaying and debugging support*/
#define LCD_DEBUG_OUT
/*State of each phase for current measurement*/
#define STATE_STAYED
/*Enable measuring the average current for reenterable code*/
#define DEBUG_CUR_CAPTURE

If LCD displaying or debugging is necessary, LCD_DEBUG_OUT should be enabled.

If time or power consumption measurement is needed for the application, STATE_STAYED should be enabled.

To get the average current during capture, DEBUG_CUR_CAPTURE should be enabled.

For example, to simply see the regular routine of the application and pictures displayed on the LCD, enable LCD_DEBUG_OUT and comment or remove the other two.

7.4.4 Debug/Release Options

For the three demos and jpeg_compression example, there are two options: “Debug” and “Release”.

- “Debug” is for users to debug conveniently, with low optimization, and debug easily step by step.
- “Release” is for better performance, so it is configured as high optimization (Balanced optimization between Speed and Size).

For the other 2 examples, there are 2 release targets “flash” and “sram”. Clearly, the names indicate that the user can run the application in the selected memory.
8. Power Consumption and Battery Life Time

8.1 Single Snapshot Mode

Table 8-1 below gives current consumption values versus duration of each main software task related to the flow chart shown in Figure 7-1 on page 15. JP6 is used for current measurements.

**Table 8-1. Power Consumption in Single Snapshot Mode**

<table>
<thead>
<tr>
<th>Function</th>
<th>Duration (ms)</th>
<th>Active Current (mA)</th>
<th>Charge (Current*Time)</th>
<th>Comments</th>
</tr>
</thead>
</table>
| MCU Wake-up and Initialization               | 0.856         | 2.1                 | 1.8E-06               | MCK = RC Osc @ 4 MHz  
Peripherals: PIOA  
Components: PIR |
| Switch MCK to PLL                            | 0.237         | 33.4                | 8E-06                 | MCK = 64 MHz from PLLA  
Peripherals: PIOA  
Components: PIR |
| SAM3S Active (transfer setup)                | 23.6          | 58.3                | 1.4E-03               | MCK = 64 MHz from PLLA  
Peripherals: PIOA, SMC, TWI  
Components: PIR, PSRAM, Image Sensor |
| CMOS image sensor (Transfer into PSRAM) + Core in Sleep Mode | 127           | 49.6                | 6.3E-03               | MCK = 64 MHz from PLLA  
Peripherals: PIOA, SMC, TWI  
Components: PIR, PSRAM, Image |
| SAM3S in Backup Mode                         | –             | 0.021               | –                     | MCK: stopped  
Peripherals: NONE  
Components: PIR |
| TOTAL                                         |               |                     | 7.73E-03              |

If, for example, 100 pictures are taken in one day, then the average charge (Amp-Hour) is 21E-06 + ((7.73E-03x100)/(24x3600)) = 30 µA.

With 2800 mAh capacitance on a 3V minimum battery supply, a battery life time of 93000 Hours is furnished, thus around 10 years of run time.
8.2 Periodic Wake-Up with Motion Detection Mode

Table 8-2 below gives current consumption number versus duration of each main software tasks related to the flowchart from Figure 7-2 on page 16. JP6 is used for current measurements.

<table>
<thead>
<tr>
<th>Function</th>
<th>Duration (mS)</th>
<th>Active Current (mA)</th>
<th>Charge (Current*Time)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU Wake-up from Wait mode and Initialization</td>
<td>0.165</td>
<td>2.234</td>
<td>0.36E-06</td>
<td>MCK = RC Osc @ 4MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: ACC, RTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, LED</td>
</tr>
<tr>
<td>Sleep Mode with PIR, ACC and TC</td>
<td>Fixed maximal: 2s (for test)</td>
<td>1.820</td>
<td>3.64E-03</td>
<td>MCK = RC Osc @ 4MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Core in sleep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: ACC, RTT, TC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, LED</td>
</tr>
<tr>
<td>Wake-up from sleep mode</td>
<td>0.030</td>
<td>2.260</td>
<td>0.06E-06</td>
<td>MCK = RC Osc @ 4MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Core in sleep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: RTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, LED</td>
</tr>
<tr>
<td>Switch MCK to PLL</td>
<td>0.220</td>
<td>35.29</td>
<td>7.7E-06</td>
<td>MCK = 64 MHz from PLLA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: RTT, LED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR</td>
</tr>
<tr>
<td>SAM3S Active (transfer setup)</td>
<td>20.3</td>
<td>59</td>
<td>1.20E-03</td>
<td>MCK = 64 MHz from PLLA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: RTT, LED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, PSRAM, Image Sensor</td>
</tr>
<tr>
<td>CMOS image sensor (Transfer into PSRAM) + Core in Sleep Mode</td>
<td>117</td>
<td>49.25(1)</td>
<td>5.76E-03</td>
<td>MCK = 64 MHz from PLLA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: RTT, LED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, PSRAM, Image Sensor</td>
</tr>
<tr>
<td>SAM3S in Wait Mode</td>
<td>-</td>
<td>33uA</td>
<td>-</td>
<td>MCK stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peripherals: RTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Components: PIR, LED</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>10.6E-03</td>
<td></td>
</tr>
</tbody>
</table>

Note: 1. Current during capturing is measured by enabling macro DEBUG_CUR_CAPTURE.

If, for example, 100 pictures are taken in one day, then the average charge (Amp-Hour) is 33E-06 + ((10.6E-03x100) / (24x3600)) = 45.33µA.

With 2800 mAh capacitance on a 3V minimum battery supply, a battery life time of 62000 Hours is furnished, thus around 7 years of run time.

8.3 Continuous Capture Mode

Details about current consumption for each software task are not given here for this mode. This application demonstrates the capability of the SAM3S to perform “live video capture”. In black and white mode the frame rate is 7fps and 5fps in color mode. Source code for this mode is provided in the software package.
9. Using the Reference Design Board

A 5V DC power supply is needed to power the board (same as provided on the SAM3S-EK).

The default application programmed on the board is the periodic wake-up with motion detection demo. After power up, the screen will be black, moving in front of the board causes a picture to be taken which is displayed on the LCD. Then, each time a movement is detected, the blue LED will turn on quickly, a picture is taken and displayed.

If the Single Snapshot Demo is programmed into the board, after power-up, pressing the BP1 push button will take a picture.

If the Continuous Capture Mode is programmed into the board, the board acts as a video camera. Pushing BP1 button will switch between B&W and color mode.

9.1 Downloading and Running the Binaries

To use the binaries, unzip “bin_pirrd.zip”. The directory structure is as follows:

-demos
  |- Continuous_Capture_Mode_Rel.bin
  |- Periodic_Motion_Detect_Mode_Rel.bin
  |- Single_Snapshot_Mode_Rel.bin

-examples
  |- imagesensor_capture_flash.bin
  |- jpeg_compression_rel.bin
  |- pyrosensor_motion_detect_flash.bin

The binary files can be downloaded directly into the Flash memory using SAM-BA® v2.10 either with Atmel SAM-ICE or via USB port. On Windows® Vista® computer, SAM-BA v2.10 CDC version must be used for USB port.

Start SAM-BA

Choose the connection and the SAM3S4-EK board, and click Connect.

On the next window shown below, from the Scripts drop-down list choose:

- Enable Flash access script, click “Execute”
- Then Boot from Flash (GPNVM1) script, click “Execute”
- Then click the icon folder close to the “Send File” button, and browse to the desired binary file. “Click Send” File button
- When asked “Do you want to lock the involved region...” choose No
- Close SAM-BA and power down then power up the board. The new application starts.
The same procedure can be done using the SAM-ICE JTAG/ICE box. Choose jlink\ARM0 connection instead.
10. Appendix

10.1 Hardware files

Board design files are provided with the reference design. See the "Hardware" folder.

Board design and board manufacturing files are provided:

Board Design files:
- SAM3S-PIRRD_RevB.pdf: Schematics, PDF format
- SAM3S-PIRRD_REVB.DSN: Schematics, Cadence® OrCAD® Capture format
- SAM3S-PIRRD_REVB.brd: PCB project, Allegro PCB Design
- SAM3S-PIRRD_REVB.xls: Bill of Materials

Board manufacturing files:
- GERBER files format.

10.2 References


Revision History

<table>
<thead>
<tr>
<th>Doc. Rev</th>
<th>Comments</th>
<th>Change Request Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>11091A</td>
<td>First issue</td>
<td></td>
</tr>
</tbody>
</table>
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.

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM®, and the ARMPowered® logo and others are registered trademarks or trademarks ARM Ltd. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be trademarks of others.