APPLICATION NOTE

AT03468: Wireless Passive Infrared Motion Detection Reference Design for SAM4S

Atmel 32-bit Microcontroller

Description

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, 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 Appendix B.

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 the Atmel® SAM4S ARM® Cortex®-M4 based MCU is particularly well suited for this kind of application and provides ready to use hardware and software blocks. The SAM4S 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 SAM4S 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.

Features

- IAR Embedded Workbench®
- ASF Board support with Peripheral drivers and examples
- SAM4S-WPIR-RD board
- Motion Detector Camera
- Wireless Image Transfer using ZigBee® PRO

Terminology and abbreviations

• PIR	Passive Infra Red
• IR	Infra Red
• PDC	Peripheral DMA controller
• ACC	Analog Comparator Controller
• ACP	Analog Comparator
• PIO	Controller Parallel Input/output Controller
• TWI	Two-Wire Interface
• GPIO	General Purpose Input/output
• BOM	Bill of Materials
• SRAM	Static RAM
• SMC	Static Memory Controller
• FPS	Frame Per Second
• JTAG	Joint Test Action Group

Table of Contents

1. Hardware Description

1.1 Overview

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

1.2 WPIR-RD Board Block Diagram

Figure 1-1. WPIR-RD Block Diagram

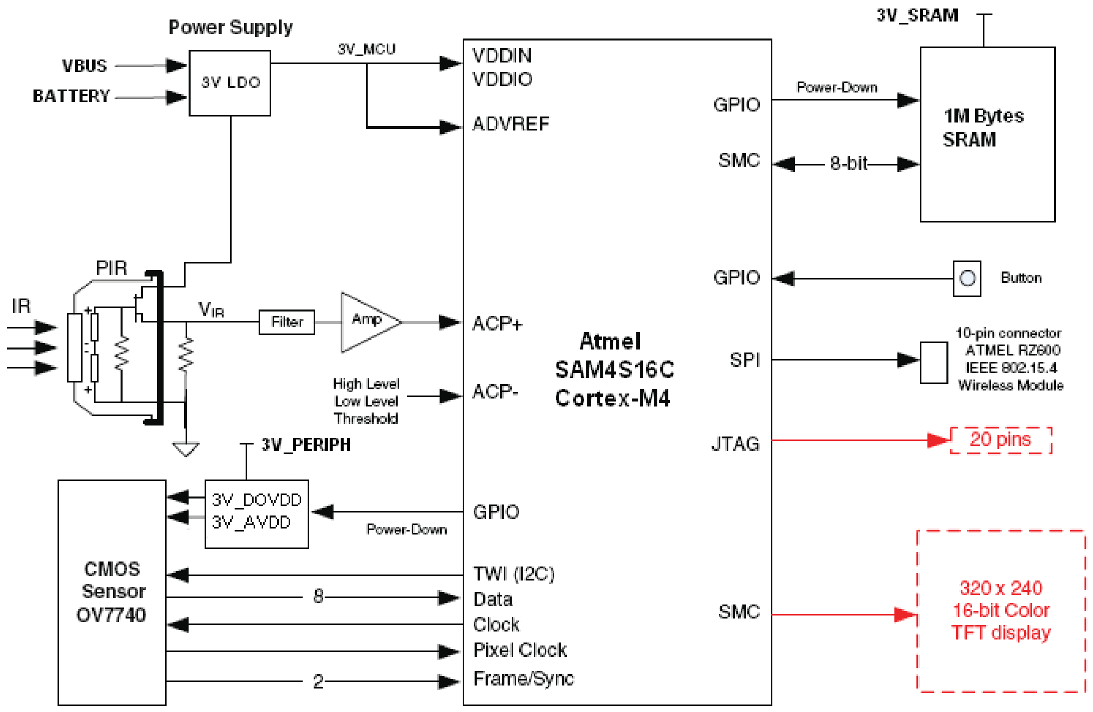
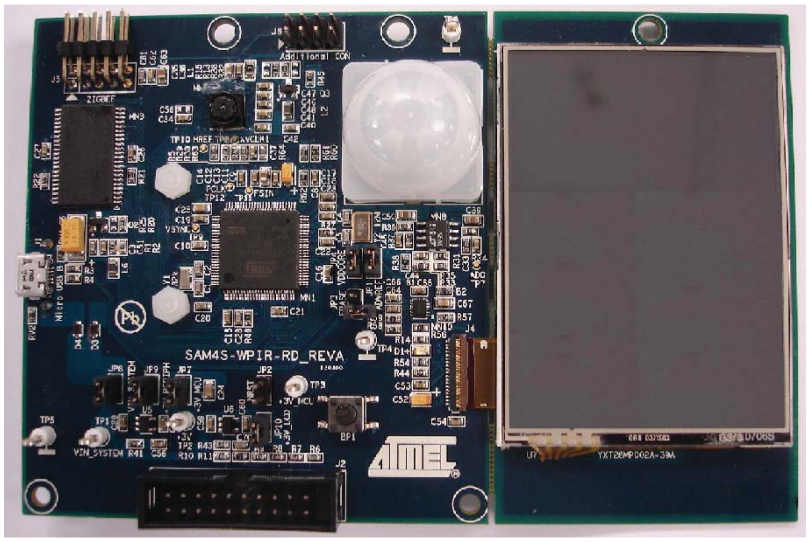


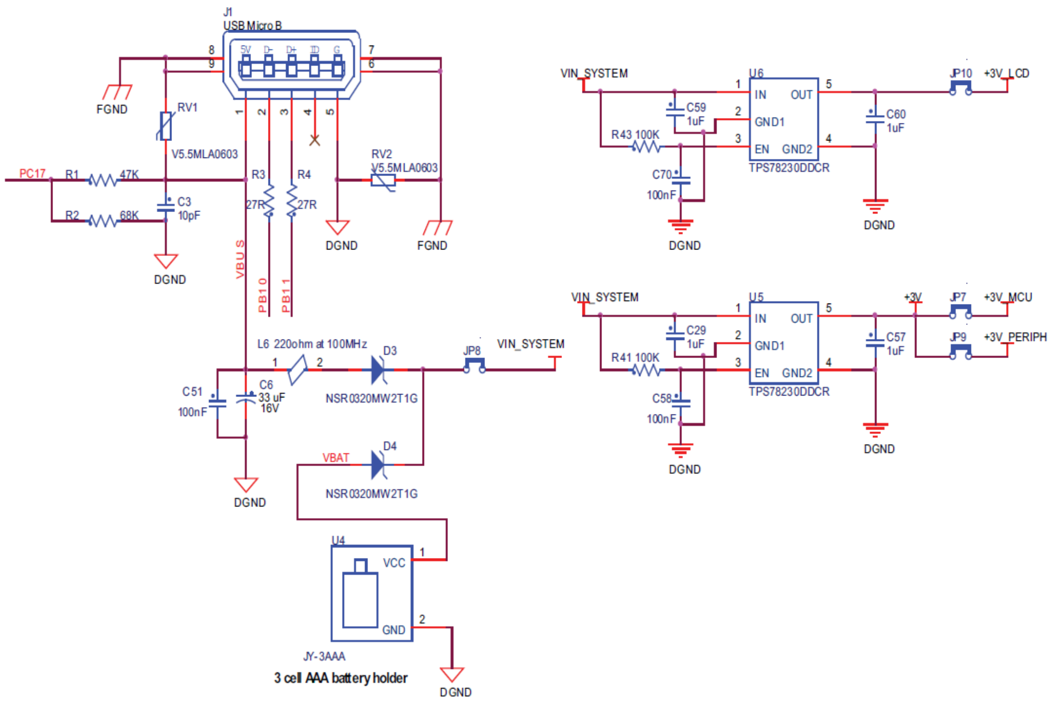
Figure 1-2. SAM4S WPIR-RD Board



1.3 Power Supply

The power supply needed for the PIR reference design is quite simple. The TPS78230 (U5) is a fixed linear voltage regulator, low-dropout and low quiescent current providing 3V for MCU and external components. Note that the SAM4S 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. U5 has a quiescent current of typical 500nA in active mode. The TPS78230 power down mode is not used in this reference design. The board can be directly supplied with 3 x AAA 1.5V Cell or can be USB powered.

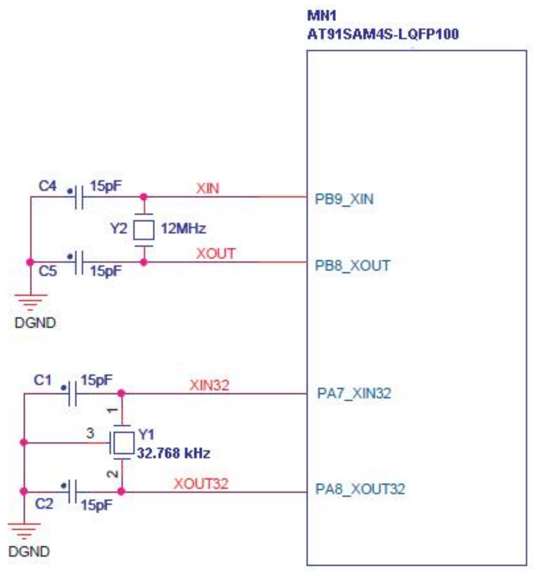
Figure 1-3. Power Supply



1.4 Oscillators

Provisions have been made on the schematics and PCB in order to use an external slow clock crystal oscillator and 12MHz crystal for the main oscillator. In our application no USB or accurate slow clock source is needed, so external crystal oscillators are not used since the SAM4S embeds one 32.768kHz RC oscillator and an accurate (factory trimmed) 4/8/12MHz fast RC oscillator. The internal PLL can work from the fast RC oscillator.

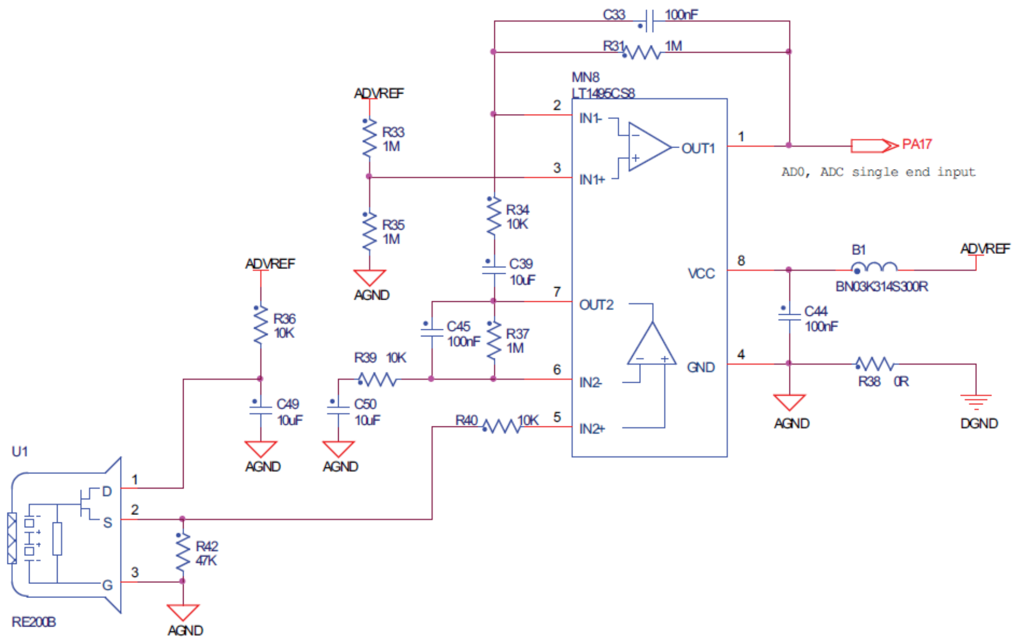
Figure 1-4. Slow Clock and Main Clock Oscillators



1.5 PIR Sensor

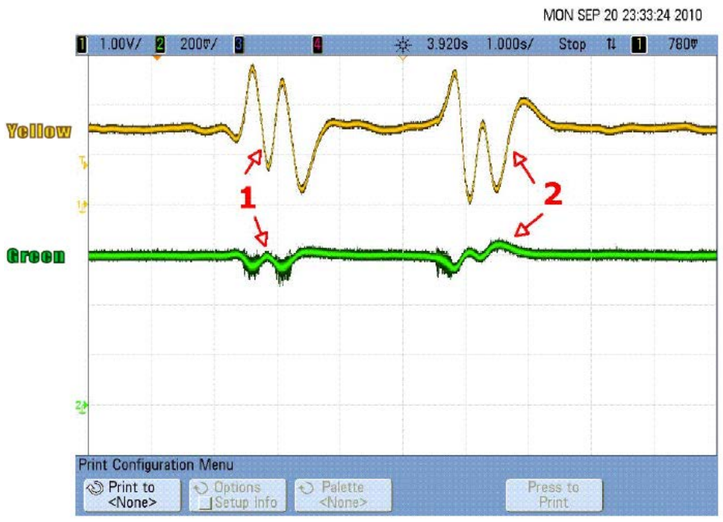
The PIR sensor (RE200B from Nicera) chosen for the reference design is a general purpose dual element. This model satisfies customer's cost reduction needs, keeping most dual element type performances at reasonable levels. PIR Sensors need a Fresnel lens to concentrate IR waves (5μm to 14μm) and to filter ambient light, see Appendix B. A detailed explanation of PIR Sensor mechanical aspects is not discussed in this document. Refer to the Mechanical section of the RE200B catalog, here: http://www.nicera.ph/images/PDF/pyro_catalog.pdf. The PIR Sensor provides a 3900mVpp signal (with 80dB gain) when measured with a 420K μW/cm^2 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 1-5. 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 1.59Hz band-pass filter at -3dB with attenuation of -40dB. The amplified and filtered signal is shown in Figure 1-6. 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).

Figure 1-6. Amplified and Filtered Signal

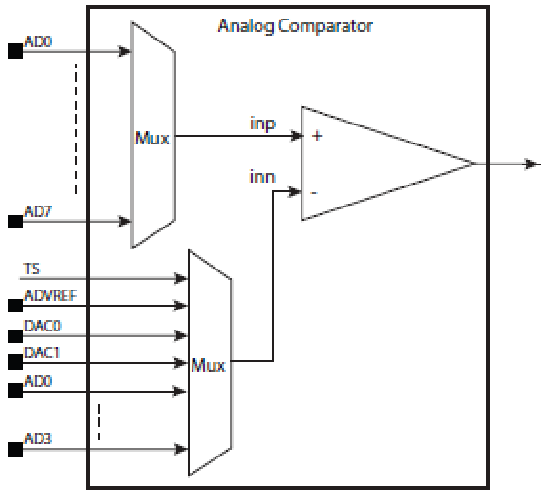


The “1” pulses represent a first movement detected in one sense, then “2” pulses indicate a second motion detected in the opposite sense. The OUT1 signal is then fed to the Analog-to-Digital Converter (ADC) AD0 (PA17) input. Since the ADC and Analog Comparator (ACP) share analog inputs, motion detection 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.

1.5.1 Using the Analog Comparator (ACP) for Motion Detection

The ACP inputs are shown in Figure 1-7. ADC and ACP share common inputs comprised of eight multiplexed inputs for INP (positive input) and four for INN (negative input). INN and INP can be swapped by software in the ACP controller (ACC).

Figure 1-7. 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 2.3.2 - PIR Sensor. The sensitivity of the detection can be adjusted by the threshold level.

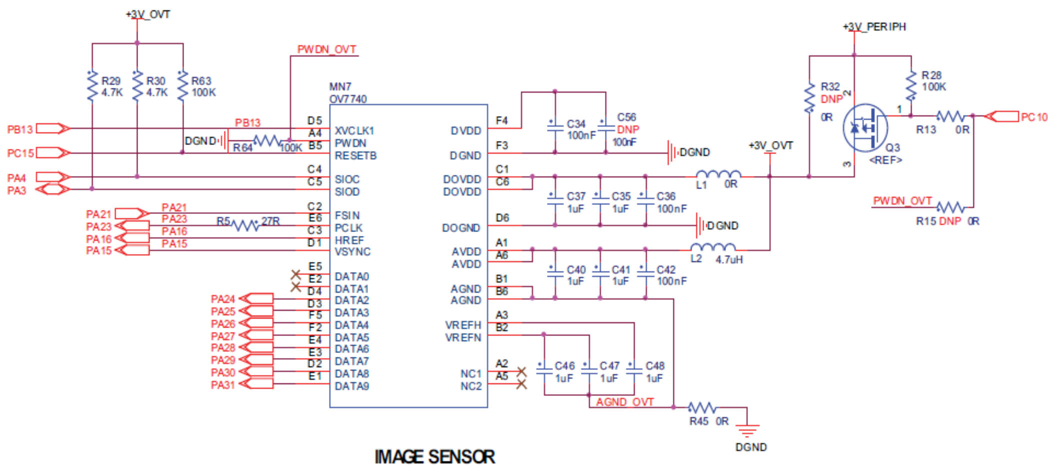
1.5.2 Using the 12-bit ADC for Motion Detection

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.

1.6 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 60fps and QVGA (320x240) at up to 120fps. 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 SAM4S device is made easy by means of the sensor interface. The SAM4S is able to sample data from the CMOS image sensor without CPU intervention using the PIO parallel capture mode and transfer image data into internal or external memory using the PIO Peripheral DMA Controller.

Figure 1-8. Omnivision OV7740 Sensor



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, generated from 1.5V internal regulator of OV7740
- 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

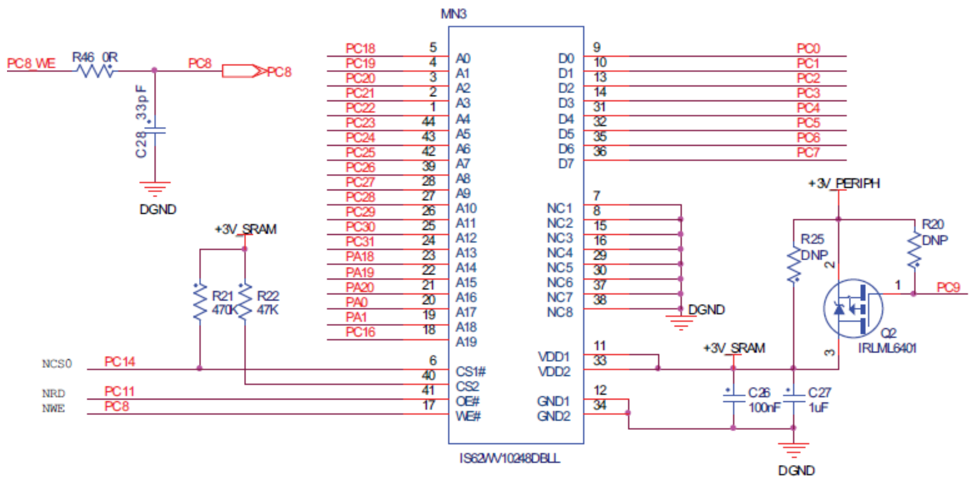
In order to manage power consumption, 3V_OVT is provided from the 3V_MCU rail, controlled by Q3 and SAM4S PC10 pin to switch ON and OFF the CMOS image sensor. For MN11, the start-up time is 0.4ms when setting PC10 at level 0. The OV7740 can receive configuration data 1ms after its power supplies are stable. Data and command settings are taken into account after 5ms. The CMOS image sensor is used in 8-bit mode, and gets its clock from a Programmable Clock Output (PCK0) coming from the SAM4S, which avoids using an external oscillator to provide a clock to the CMOS image sensor. The XVCLK1 (main input clock) ranges from 6MHz to 27MHz, 24MHz 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 two 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 SRAM access time of 55ns, 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 SAM4S 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 (I^{2}C compatible) SIOC (PA4) and SIOD (PA3) pins.

1.7 Memory

To store one or more 320x240 images from the sensor, a buffer is needed. An external low cost, low power SRAM memory from ISSI is connected to the static memory controller of the SAM4S. As shown in Figure 1-9, 8-bit x 1M bytes is used.

Figure 1-9. External Memory



1.8 JTAG Connector, TFT Display, and USB

The JTAG/SWJ-DP interface, the 16-bit color TFT and USB are for debug and demonstration purpose. Note that power consumption of the above devices does not affect the current consumption values given in the following sections. As seen in the board's front side picture, Figure 1-2, the board has a pre-cut line used to remove the LCD Side if a wireless link is to be used only.

1.9 ZigBee Connector

Provision has been made on the board for connecting a ZigBee transceiver Module. The RZ600 [5] radio module and the REB231 [7] are referred here, as examples. A user-designed IEEE®802.15.4 Atmel Transceiver based module can also be connected to this connector with the pin mapping mentioned below.

Table 1-1 gives a summary of the pin connectivity between the ZigBee connector J5 on the SAM4S-PIRRD board and the RZ600 radio stick. A pin mapping for the REB231 radio extender board [7] is also provided for reference.

Table 1-1. ZigBee Connector Pin Mapping

Connector Pin	I/O Line on SAM4S-PIRRD	RZ600	REB231
1	PA5	1 (RSTN)	25 (RSTN)
2	PB3	2 (MISC)	N.A
3	PB2	3 (Interrupt)	38 (IRQ)
4	PA6	4 (Sleep Transmit)	26 (SLPTR)
5	PB14	5 (Chip Select)	30 (SEL)
6	PA13	6 (MOSI)	28 (MOSI)
7	PA12	7 (MISO)	27 (MISO)
8	PA14	8 (SCK)	29 (SCLK)
9	Ground	9 (GND)	21 (GND)
10	+3V	10 (1.8 to 3.6V)	19 (V_{CC})

1.10 General Purpose I/Os Control

Table 1-2 gives a summary of the SAM4S PIOs used to control power supply pins of external devices.

Table 1-2. PIO Control Summary

Device	Pin	Active Level
CMOS Image Sensor	PC10	0
External SRAM	PC9	0

An additional connector J6 providing GPIOs, UART, and TWI (I^2C) connections are also available for user's needs.

Figure 1-10. J6 Connector

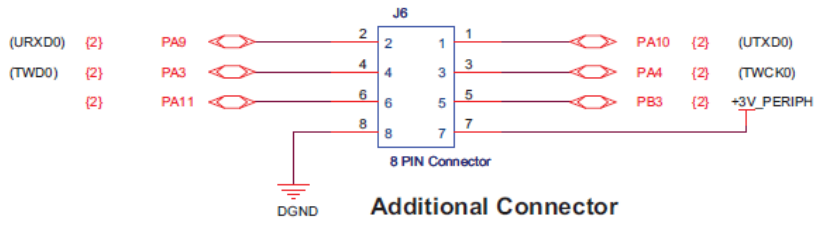


Table 1-3. General Purpose I/O Description

Connector Pin	I/O Line	Peripheral A	Peripheral B	Peripheral C	Extra Function	Comments
1	PA10	UTXD0	NPCS2			UART 0 (1)Transmit
2	PA9	URXD0	NPCS1	PWMFI0	WKUP6	UART 0 (1) Receive
3	PA4	TWCK0	TCLK0		WKUP3	TWI Clock (2)
4	PA3	TWD0	NPCS3			TWI Data (2)

5	PB3	UTXD1	PCK2		AD7	Digital Input/output Clock Output Analog Input
6	PA11	NPCS0	PWMH0		WKUP7	Digital Input/output
7	+3V					
8	Ground					

Notes: 1. No RS232 Transceiver.

2. External Pull-up in the range of 4.7kΩ must be added when connecting the TWI Clock and TWI Data to a TWI device.

1.11 Reference Design Schematics

See Appendix B, for schematics and bill of materials details.

1.12 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 1.6 - CMOS Image Sensor, ODT resistors help reduce PCB design space and assure signal integrity to achieve better PIR detection and image quality.

2. Software Description

2.1 Overview

The software is intended to provide some guidance to use with SAM4S and components on the SAM4S-WPIR-RD board. It can also be used as a reference design to deliver a final product. The software is composed of three examples and four 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 four demos provide reference designs or demonstrations for specific low-power mode available in SAM4S 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 Chapter 3 - Power Consumption and Frame Rate.

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.

2.2 Demonstration Applications

Described below are four demonstration applications for the reference design:

- Single snapshot
- Periodic motion detection
- Continuous capture with external SRAM
- Continuous capture without external SRAM

These four applications 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, and JPEG compression.

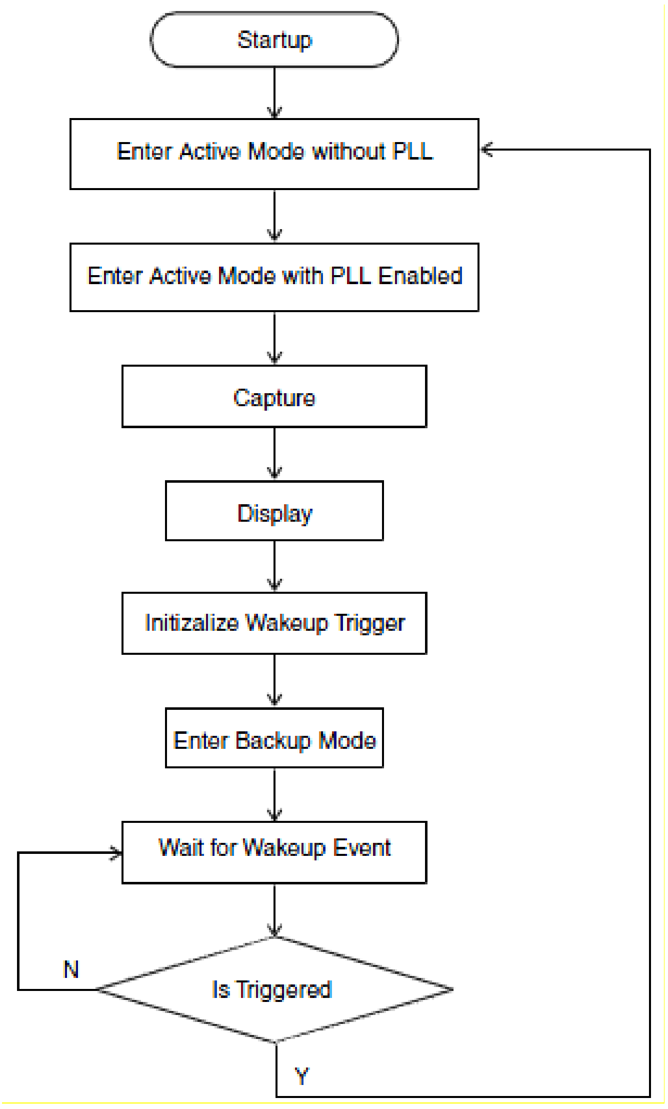
Along with these demo applications, a Wireless Image Transfer Demo using ZigBee protocol is described in Section 2.5.

2.2.1 Single Snapshot

This application demonstrates CMOS image sensor capture through the SAM4S device backup mode. SAM4S 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 demonstration use the Backup mode as defined in the SAM4S series datasheet. For detailed data on power consumption and duration, see Chapter 3 - Power Consumption and Frame Rate.

The detailed procedure is provided in Figure 2-1.

Figure 2-1. Flowchart of Single Snapshot

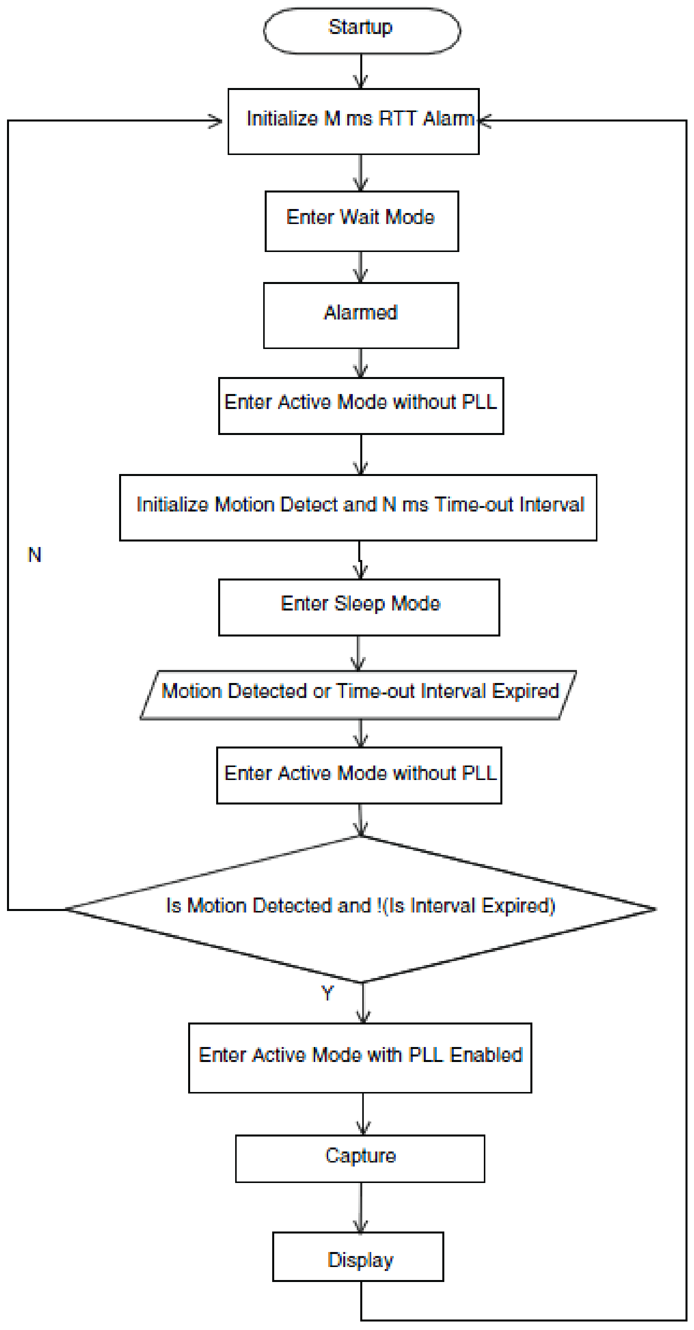

Startup
Enter Active Mode without PLL
Enter Active Mode with PLL Enabled
Capture
Display
Initizalize Wakeup Trigger
Enter Backup Mode
Wait for Wakeup Event
N
Is Triggered
Y

2.2.2 Periodic Motion Detection

In this application, to minimize power consumption, the SAM4S (which drives the PIR sensor) use two low power modes. At the beginning, the SAM4S enter in wait mode and stay in it during 200ms (wake up from RTT Alarm). After that, microcontroller initialize Motion detect and RTT alarm before entering into sleep mode. There are two possibilities to wake up from this sleep mode. The first one is the RTT Alarm (fixed to 500ms), so SAM4S will go back at the previous state (Wait mode).The second one is a Motion detected, so a picture will be taken by CMOS sensor and display on LCD, then microcontroller will return in wait mode state. For more detailed data of power consumption and duration, see Table 3-1 and Table 3-2 in Chapter 3 - Power Consumption and Frame Rate.

The detailed procedure is provided in Figure 2-2.

Figure 2-2. Flowchart of Periodic Motion Detection

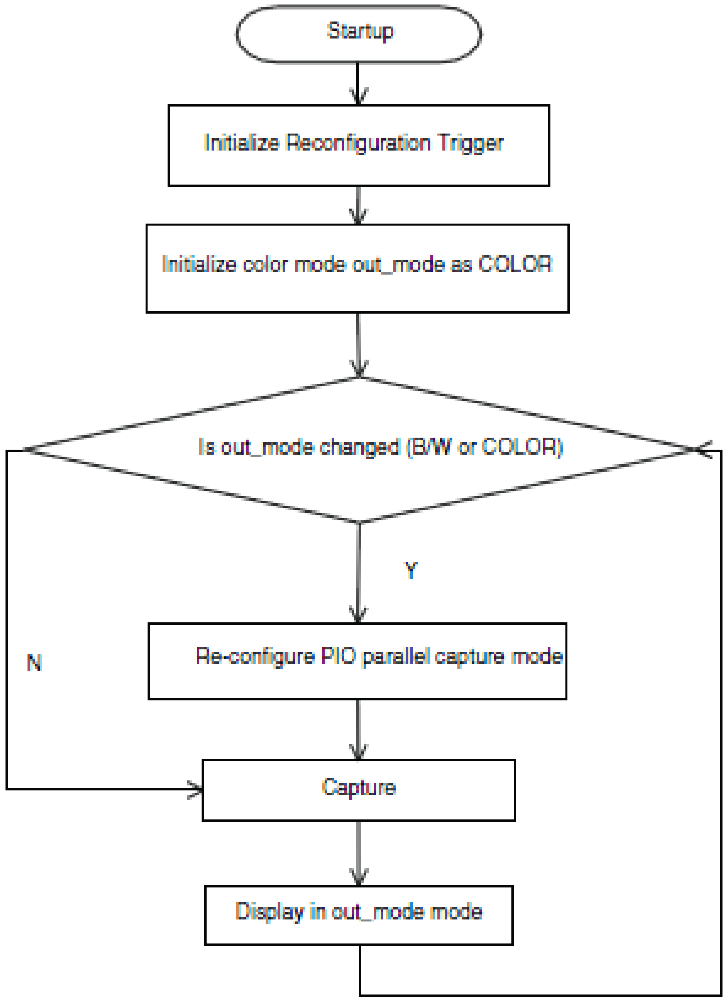

Startup
Initialize M ms RTT Alarm
Enter Wait Mode
Alarmed
Enter Active Mode without PLL
Initialize Motion Detect and N ms Time-out Interval
N
Enter Sleep Mode
Motion Detected or Time-out Interval Expired
Enter Active Mode without PLL
Is Motion Detected and !(Is Interval Expired)
Y
Enter Active Mode with PLL Enabled
Capture
Display

2.2.3 Continuous Capture with External SRAM

This application is intended to improve frame rate performance when capturing images and displaying them onto the LCD. Only the frame rate measurements are given for this demonstration. The push button is used to switch between color and Black and White mode. B&W mode can achieve better frame rate performance because a B&W picture needs less data than a color picture. Picture capturing by CMOS image sensor are saved directly in external SRAM, then SAM4S device load and decode this picture and display it on LCD.

More details are given in Figure 2-3.

Figure 2-3. Flowchart of Continuous Capture with External SRAM

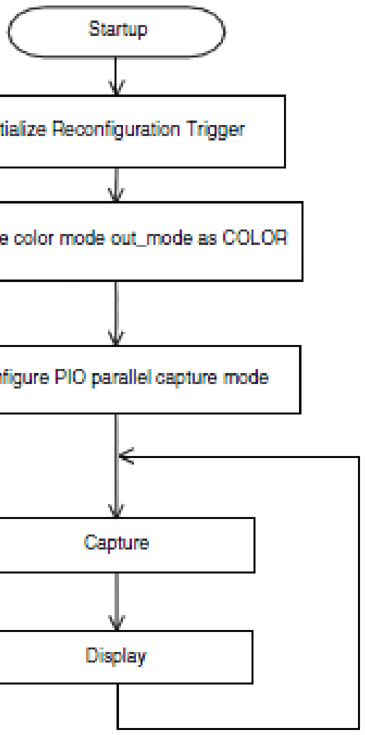

Startup
Initialize Reconfiguration Trigger
Initialize color mode out_mode as COLOR
Is out_mode changed (B/W or COLOR)
Y
N
Re-configure PIO parallel capture mode
Capture
Display in out_mode mode

2.2.4 Continuous Capture without External SRAM

This application is similar to the previous demonstration but it doesn't use the external SRAM. When a picture is taken, this picture is sliced into three parts which are saved in three different buffers. SAM4S device decodes and displays these buffers one by one on the LCD. Notes that B&W mode is not implemented on this demonstration.

The detailed procedure is provided in Figure 2-4.

Figure 2-4. Flowchart for Continuous Capture without External SRAM


Startup
Initialize Reconfiguration Trigger
Initialize color mode out_mode as COLOR
configure PIO parallel capture mode
Capture
Display

2.3 Code Implementation

The following section show common function or piece of software using in the four demonstration and the three examples.

2.3.1 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.

2.3.1.1 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 1.10 - 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_PULLUP etc., can be found in the device header file, for example, sam4s16c.h for the SAM4S16 device.

```
pio_set_input(PIOC, PIO_PC12, PIO_PULLUP);
```

2.3.1.2 Clock for Maximum Frequency

The main clock source is the internal fast RC oscillator. By default, the main clock source is the internal fast RC oscillator at 4MHz. To work at maximum frequency, main clock source have to be the external crystal at 12MHz and PLLB have to be used as mck source. To configure microcontroller with these settings, enable the correct define in conf_board headers and call the board_init(); function:

```
#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_PLLBCK //PLLB enable
#define CONFIG_SYSCLK_PRES       SYSCLK_PRES_2   //divide by 2
#define CONFIG_PLL1_SOURCE       PLL_SRC_MAINCK_XTAL //use main crystal at 12Mhz
#define CONFIG_PLL1_MUL          20
#define CONFIG_PLL1_DIV          1

board_init();//call it to configure microcontroller at (20/1)*12/2= 120 Mhz
```

2.3.1.3 Clock for Maximum Frequency

As SRAM size (128KB on SAM4S16C devices) is not sufficient to store capture buffer, image frame buffer, etc., 1MB external SRAM is provided on the board. The SRAM is driven by the SMC (Static Memory Controller), the user interface of which contains several registers to configure the timing and mode for SRAM. See the SAM4S 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.

```
/* Enable peripheral clock */
              pmc_enable_periph_clk( ID_SMC ) ;

              /* Configure SMC interface for SRAM */
              smc_set_setup_timing(SMC,SRAM_CS,SMC_SETUP_NWE_SETUP(2)
                     | SMC_SETUP_NCS_WR_SETUP(0)
                     | SMC_SETUP_NRD_SETUP(3)
                     | SMC_SETUP_NCS_RD_SETUP(0));

              smc_set_pulse_timing(SMC, SRAM_CS , SMC_PULSE_NWE_PULSE(4)
                     | SMC_PULSE_NCS_WR_PULSE(5)
                     | SMC_PULSE_NRD_PULSE(4)
                     | SMC_PULSE_NCS_RD_PULSE(6));

              smc_set_cycle_timing(SMC, SRAM_CS, SMC_CYCLE_NWE_CYCLE(6)
                     | SMC_CYCLE_NRD_CYCLE(7));

              smc_set_mode(SMC, SRAM_CS, SMC_MODE_READ_MODE
                     | SMC_MODE_WRITE_MODE
                     | SMC_MODE_DBW_8_BIT);
```

The SMC in SAM4S has four chip selects. The user needs to configure the respective one by define SRAM_CS in conf_board.h headers:

```
#define SRAM_CS      0
```

Others definition can be found in component_smc.h.

2.3.2 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.

2.3.2.1 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.

2.3.2.2 ACC Initialization

For the ACC implementation in SAM4S 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 `pir_motion_detect_example`. The chip library contains the interface operating on ACC. The most important one is `acc_init`. Its definition follows:

```
void acc_init(Acc *p_acc, uint32_t ul_select_plus, uint32_t, ul_select_minus,
                    uint32_t ul_edge_type, uint32_t ul_invert)
```

where `ul_select_plus` and `ul_select_minus` are the paired inputs and `ul_edge_type` is the edge type.

2.3.2.3 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.

2.3.2.4 Software Example

A PIR Sensor motion detection example is provided in ASF (Atmel Studio->New->Example Project->ASF->RE200B PIR Sensor Motion Detection Example) [8] and in the zip package provided with this application note.

See the section 2.4 - Software Deliveries for more details.

2.3.3 CMOS Imaging Sensor

The SAM4S device is capable of capturing parallel data in synchronization with an external clock, which is used for interfacing to a CMOS image sensor.

2.3.3.1 Initialization of PIO Parallel Capture and OV7740

An 8-bit parallel capture mode is available inside the SAM4S 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, I^2C compatible serial interface. For our application, PIO capture mode is used for the clock and data interface and TWI (I^2C 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 SAM4S series datasheet. Only a few functions are presented here: B&W mode and color mode setting, master clock/pixel clock and capture size.

2.3.3.2 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:

```
/* Only HSYNC and VSYNC enabled*/
p_pio->PIO_PCMR &= ~((uint32_t)PIO_PCMR_ALWYS);
p_pio->PIO_PCMR &= ~((uint32_t)PIO_PCMR_HALFS);

#if !defined(DEFAULT_MODE_COLORED)//in black and white mode
```

```
/* Samples only data with even index*/
p_pio->PIO_PCMR |= PIO_PCMR_HALFS;
p_pio->PIO_PCMR &= ~((uint32_t)PIO_PCMR_FRSTS);
#endif
```

2.3.3.3 Master Clock/Pixel Clock

The master clock is provided through the Programmable Clock with multiplexed I/Os. The source and prescaler 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 SAM4S-WPIR-RD, OV7740 works in a wide range of frequencies from 6MHz to 27MHz. To support best capture performance, the PIO parallel capture mode should work at its best capability. In the SAM4S 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 35MHz (as listed in I/O /Characteristics table in the Electrical Characteristics section of the SAM4S series datasheet). For example, in the referenced application, the frequencies of Master clock and Pixel clock to and from OV7740 are configured to 24MHz thanks to PLLA, which meets the requirements of both OV7740 and PIO capture mode. See code below to check how it works.

```
pmc_enable_pllack (7, 0x1, 1);//configure PLLA to provide (8/1)*12= 96Mhz

/* Init PCK0 to work at 24 Mhz*/
PMC->PMC_PCK[0] = (PMC_PCK_PRES_CLK_4|PMC_PCK_CSS_PLLA_CLK);// 96/4=24 Mhz

/* Enable PCK0 output*/
PMC->PMC_SCER = PMC_SCER_PCK0;
while(!( PMC->PMC_SCSR & PMC_SCSR_PCK0));
```

2.3.3.4 Captured Size

The captured size is determined by the size of the captured image and mode, color or Black and White. 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.

2.3.3.5 Parallel Capture Event Handler

The parallel capture event is associated with PDC transfer event. It is 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_Handler`
- Without interrupt support: Checks the PDC flags after the transfer size is set and the PDC is enabled as done in image_sensor_capture application

2.3.3.6 Parallel Capture Event Handler

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. 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 SAM4S 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.

2.3.3.7 Software Example

An image sensor example is provided in ASF (Atmel Studio->New->Example Project->ASF->OV7740 CMOS Image Sensor Example) [8] and in the zip package provided with this application note.

See the section 2.4 - Software Deliveries for more details.

2.3.4 JPEG Compression

2.3.4.1 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.

2.3.4.2 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 encode JPEG, the following functions are used:

```
ijg_compress (): 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/.

2.3.4.3 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 external SRAM. The dynamic memory for compression objects is located in internal 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. See Section 2.4 - Software Deliveries for details.

Table 2-1. JPEG Compression/Decompression Performance (with IAR Toolchain)

	Compress	Decompress
Heap in internal SRAM	129ms	119ms

Table 2-2. JPEG Memory Consumption (with IAR Toolchain)

	Flash [Bytes]	SRAM [Bytes]
Heap in internal SRAM	119,428 (RO)	53,050 (RW+ Stack + Heap)

2.3.4.4 Using the JPEG Library

A JPEG compression example is provided in the in the zip package provided with this application note. These examples show how to use the JPEG library See Section 2.4 - Software Deliveries for more details.

2.3.5 Wireless Communication

A demo application that uses ZigBee PRO for the transfer of the captured image over a ZigBee PRO network is provided along with the application note. A PC application called the `WPIRMonitor` shall be used for monitoring the network state. See Section 2.5 - Wireless PIR Image Transfer Demo for more details.

2.4 Software Deliveries

A zip package `SAM4S_WPIR_RD_V1_1.zip` is provided with this application note document.

2.4.1 Contents Overview

This application notes refers to four demonstrations and three examples, along with the Wireless Image transfer Demo.

The corresponding binaries files are provided in the zip file that comes along with this document:

- `bin_wpir_rd_1.1`: Binary Files
  - Demos:
    - Wireless Image Transfer – set of images
    - single_snapshot_rel.bin
    - Periodic_Motion_Detect_rel.bin
    - Continuous_Capture_With_Ext_SRAM_rel.bin
    - Continuous_Capture_Without_Ext_SRAM_rel.bin

- Examples:
  - imagesensor_capture_rel.bin
  - jpeg_compression_rel.bin
  - pyrosensor_Motion_Detect_rel.bin

The corresponding sources and project files are provided in the zip file that comes along with this document:

- `source_wpir_rd_1.1`: Sources and project Files

Apart from this, board support and peripheral driver examples for the SAM4S PIRRD board are available in Atmel Software Framework (ASF) [9]. This includes the image sensor capture driver with example and the PIR sensor driver with example project.

2.4.2 Compiling Options

The following macros need to be customized for different purposes, especially for demos such as single_snapshot and periodic_motion_detection. These definitions are found at the very beginning of the "local definition" part of the main file.

```
/** macro to control the staying in each state permanently, comment for
continuous transition */
#define STATE_STAYED

/** display information on LCD during State by State mode (uncomment
STATE_STAYED). Uncomment it for current measure*/
#define LCD_DISPLAY_INFORMATION

/*macro to measure Frame rate, comment if it's not used*/
#define FRAME_RATE_MEASURE
```

STATE_STAYED allow running the software state by state (for time or power consumption measurement).

When STATE_STAYED is enabled, LCD_DISPLAY_INFORMATION will allow the LCD to display information about current state. Uncomment this macro to perform better time and power consumption measurement.

To measure LCD Frame rate enable FRAME_RATE_MEASURE macro (only for continuous capture_without external SRAM and continuous capture with external SRAM demo).

2.4.3 Debug/Release Options

For all examples and demonstrations (except continuous capture without external SRAM and JPEG compression example), there are four configurations: "flash_debug", "flash_release", "sram_debug", and "sram_release":

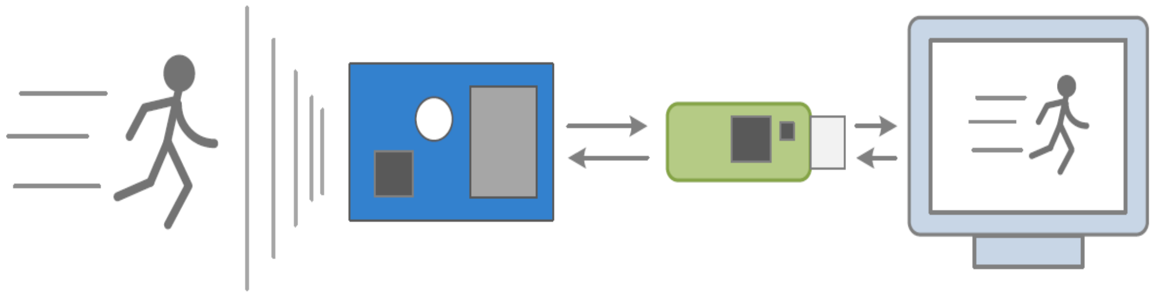
- "Debug" is for users to debug conveniently, with low optimization, and debug easily step by step
- "Release" is for better performance, so it is configured as high optimization for size
- "flash" for load software in Flash memories
- "sram" for load software in SRAM memories

Only the "flash_debug" and "flash_release" configuration are provided for continuous capture without external SRAM and JPEG compression example.

2.5 Wireless PIR Image Transfer Demo

A demo application that shows the transfer of captured image data over a ZigBee PRO network is available in the zip package that comes with this application note. This reference application involves at least two devices: one RF231USB-RD stick acting as a display (through connection with a PC) and one or more SAM4S PIRRD boards acting as camera devices. A basic understanding of ZigBee networking is required to evaluate this demo.

Figure 2-5. WPIR Demo Illustration



The RF231USB-RD stick is the ZigBee coordinator, while cameras may be either ZigBee routers or ZigBee sleeping end-devices. Each SAM4S-PIRRD boards along with the RZ600-RF231 [5] radio stick plugged into the ZigBee connector, J5, can be programmed as a ZigBee router or a ZigBee end-device.

End devices, when inactive, are put in a special state with reduced power consumption level, waking up only to exchange data with its parent device.

The PC application may be used to send a command to a particular camera to wake it up (if it is an end device), make an instant snapshot, and transmit the image to the coordinator.

The PC application also shows the network topology, indicating direct transmission links between devices. To maintain the network topology up to date in the PC application, all devices periodically report their network status to the coordinator.

2.5.1 Hardware Setup

1. ZigBee Coordinator: RF231USB-RD [6]. This tool has on board the AT91SAM3S4BA device and a 2.4GHz RF transceiver, the AT86RF231.
2. ZigBee Router/End-device: SAM4S-PIRRD board with the RZ600-RF231 [5] radio stick plugged into the ZigBee connector, J5, on the SAM4S-PIRRD.

A ZigBee network is typically made up of one coordinator and multiple routers and end-devices, forming a mesh network.

2.5.2 Package Structure

The package structure is shown in Table 2-3.

Table 2-3. SAM4S WPIR-RD Wireless Image Transfer Demo Firmware Package Structure

Location	Comment
`\bin_wpir_rd_1.1\Wireless_image_transfer`	Firmware image files of the reference application; see Table 2-5
`\source_wpir_rd_1.1\Applications\WPIRDemo`	Reference application's folder
`\source_wpir_rd_1.1\Applications\WPIRDemo\iar_projects`	IAR project files for the reference application
`\source_wpir_rd_1.1\Applications\WPIRDemo\libraries`	Auxiliary libraries from the SAM4S software package; given in source code and as compiled binaries
`\source_wpir_rd_1.1\BitCloud\lib`	Pre-compiled BitCloud® libraries
`\source_wpir_rd_1.1\BitCloud\Components\HAL`	Implementation of the Hardware Abstraction Layer of the BitCloud stack for SAM4 PIRRD and RF231USB-RD boards

2.5.3 Pre-compiled Firmware Image Files

A set of firmware image files are available in the `\bin_wpir_rd_1.1\Wireless_image_transfer\WPIRDemo` directory that can be used for getting started quickly with the Wireless PIR image transfer demo.

The types of images are tabulated in Table 2-4, Table 2-5, and Table 2-6.

Table 2-4. Pre-compiled Firmware Image Files – Configured ZigBee Channels

Image file directory	Firmware image files for 2.4GHz ZigBee channel
`\WPIRDemo\ch_0x0f`	`0x0F`
`\WPIRDemo\ch_0x10`	`0x10`
`\WPIRDemo\ch_0x15`	`0x15`
`\WPIRDemo\ch_0x16`	`0x16`

Each directory `\WPIRDemo\ch_0xXX` contains 28 firmware image files as given in Table 2-5. One set of images is available for creating a ZigBee network with no security and another set of images for creating a ZigBee network with standard network layer security.

Table 2-5. Pre-compiled Firmware Image Files – No Security

Image file name	ZigBee Device Role	Hardware
`WPIRDemo_Rf231UsbRd.bin`(1) `WPIRDemo_Rf231UsbRd.srec` (2)	Coordinator	RF231USB-RD
`WPIRDemo_WPIRRD_B_Sender_Enddevice_x.bin`(1)(3) `WPIRDemo_WPIRRD_B_Sender_Enddevice_x.srec` (2)	End-device	SAM4S PIRRD
`WPIRDemo_WPIRRD_B_Sender_Router.bin` (1) `WPIRDemo_WPIRRD_B_Sender_Router.srec` (2)	Router	SAM4S PIRRD

Table 2-6. Pre-compiled Firmware Image Files – Standard Security

Image file name	ZigBee Device Role	Hardware
`WPIRDemo_Rf231UsbRd_Sec.bin` (1) `WPIRDemo_Rf231UsbRd_Sec.srec` (2)	Coordinator	RF231USB-RD
`WPIRDemo_WPIRRD_B_Sender_Sec_Enddevice_x.bin` (1) `WPIRDemo_WPIRRD_B_Sender_Sec_Enddevice_x.srec` (2)	End-device	SAM4S PIRRD
`WPIRDemo_WPIRRD_B_Sender_Sec_Router.bin` (1) `WPIRDemo_WPIRRD_B_Sender_Sec_Router.srec` (2)	Router	SAM4S PIRRD

Notes:
1. *.bin files shall be programmed using the AT91SAM-ICE™ JTAG emulator and SAM Boot Assistant [3].
2. *.srec files shall be programmed using the PC Bootloader tool from the AVR® 2054 Serial Bootloader package [4].
3. `WPIRDemo_WPIRRD_B_Sender_Enddevice_x.bin`: The `_x` convention indicates presence of several files numbered from `_1` to `_5`. Each end-device file is unique with respect to the 64-bit MAC address used.

2.5.4 Application Parameters

Application parameters are set in the `configuration.h` file of the reference application:

- The definition for `APP_SENDER_DEVICE or APP_RECEIVER_DEVICE` determines the type of device it is: camera or display, respectively

Example:

If `APP_SENDER_DEVICE` is defined, assigning `DEV_TYPE_ROUTER` or `DEV_TYPE_ENDDEVICE` to `APP_SENDER_DEVICE_TYPE` specifies whether the camera will be an end-device (sleeping device) or a router, respectively

- `APP_MAX_SENDER_DEVICES_AMOUNT` specifies the maximum possible number of cameras in the network

Example:

#define APP_MAX_SENDER_DEVICES_AMOUNT 5

- `APP_NETWORK_INFO_SENDING_PERIOD` sets the interval in milliseconds for reporting network status to the coordinator
- The `CS_UID` parameter is used to specify the 64-bit extended address (MAC) of the device. Extended addresses on all devices in the network must be unique
- The `CS_CHANNEL_MASK` parameter sets the possible radio channels (2.4GHz) that may be used by devices. A particular channel is selected by the coordinator when the network is formed. Other devices, while joining the network, will scan all specified channels to discover the coordinator
- The `CS_END_DEVICE_SLEEP_PERIOD` parameter specifies the duration of the end device's sleep period. The reference application uses the default value (10 seconds) set in the `csDefaults.h` file. To assign a custom value to the parameter, define it in the application's `configuration.h` file:

```
#define CS_END_DEVICE_SLEEP_PERIOD <custom_value>
```

Table 2-7. Application Parameters Used

Parameter	RF231USB	Sender – end device	Sender – router
`APP_MAX_SENDER_DEVICES_AMOUNT`	5	Not applicable	Not applicable
`APP_SENDER_DEVICE_TYPE`	Not applicable	DEV_TYPE_ENDDEVICE	DEV_TYPE_ROUTER
`CS_EXT_PANID`	0xCB66F1C3302F778DLL	0xCB66F1C3302F778DLL	0xCB66F1C3302F778DLL

2.5.5 WPIR Monitor

The WPIRMonitor is the PC counterpart to the WPIR Image transfer embedded application. The installer is available in the zip package in the directory, `\bin_wpir_rd_1.1\Wireless_image_transfer\WPIRMonitor.`

The WPIRMonitor can be installed in the system (check that JRE 1.6 or greater is installed) and the RF231USB-RD connected to the PC with the coordinator firmware flashed into it. The RF231USB-RD stick will be installed as a Virtual COM port (VCP) and will interact with the WPIRMonitor.

The WPIRMonitor will display the ZigBee network topology once several nodes (SAM4S-PIRRD boards) are connected to the ZigBee network. This can be viewed in the `Net View` tab in the WPIRMonitor.

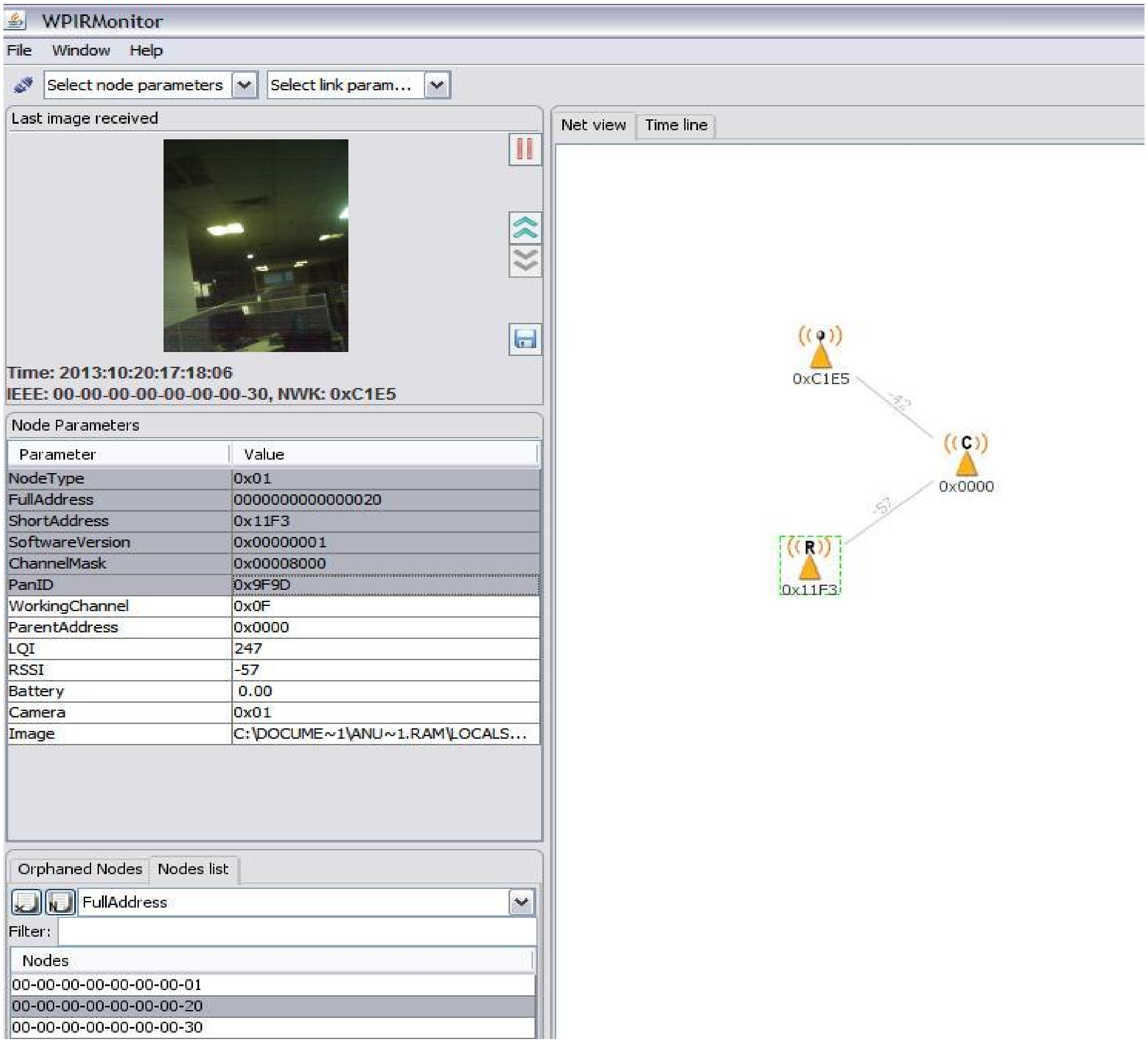
The `Time Line` tab shows snapshots of all the images captured so far by the SAM4S-PIRRD boards and sent to the coordinator wirelessly.

The `last image received` window shows the latest image received by the coordinator along with the IEEE address and network address of the SAM4S-PIRRD board which captured and sent the image wirelessly. It is also possible to pause the display of the latest image received and restart the display in the WPIRMonitor using the Play/Pause button in the `last image received` pane.

A `Nodes list` pane which shows the IEEE/network addresses of all the devices in the network and a `Node Parameters` pane that shows the link parameters of a selected node are also available in the WPIRMonitor.

Figure 2-6 shows the `Net view` of the WPIRMonitor. A ZigBee network comprising of the coordinator, a router and an end-device is shown.

Figure 2-6. WPIRMonitor Net View



Time: 2013:10:20:17:18:06
IEEE: 00-00-00-00-00-00-00-30, NWK: 0xC1E5

Node Parameters

Parameter	Value
NodeType	0x01
FullAddress	0000000000000020
ShortAddress	0x11F3
SoftwareVersion	0x00000001
ChannelMask	0x00008000
PanID	0x9F9D
WorkingChannel	0x0F
ParentAddress	0x0000
LQI	247
RSSI	-57
Battery	0.00
Camera	0x01
Image	C:\DOCUME~1\ANU~1.RAM\LOCALS...

Nodes
00-00-00-00-00-00-00-01
00-00-00-00-00-00-00-20
00-00-00-00-00-00-00-30

2.5.6 Quick Start Guide

To run the demo using the pre-compiled images:

1. Program one RF231USB-RD stick device with the coordinator's firmware, `WPIRDemo_Rf231UsbRd.bin`.
2. Program a SAM4S PIRRD board with:
   a. `WPIRDemo_WPIRRD_Sender_Enddevice.bin` to make it an end device.
   b. `WPIRDemo_WPIRRD_Sender_Router.bin` to make it a router.

   Note: All devices in a demo network shall be programmed with different firmware files, because each firmware file sets a particular extended address to the device, and all devices in the network should have different extended addresses.

   Note: To employ security in the demo network, use firmware files with `_Sec` in the names. In this case all devices shall be programmed with such files.
3. Attach the RF231USB-RD stick to a USB port on a PC.

4. Attach RZ600-RF231 radio extenders to SAM4S PIRRD boards and power on the boards (by connecting them with a micro-USB cable to a PC or by batteries).

   Launch `WPIRMonitor` on the PC to which the RF231USB-RD stick is attached.

5. Specify the connection settings for the stick by selecting the `File > Connect` menu item or clicking the top-left button on the toolbar:

   a. Choose serial as the protocol.

   b. Select the COM port.

      To find out the port number assigned to the RF231USB-RD stick, open Device Manager and look for `Virtual COM port for BitCloud` under the `Ports(COM & LPT)` heading.

      Note: The Virtual COM port driver is available in the directory: `source_wpir_rd_1.1\demo_wpir_rd\Wireless_image_transfer\ThirdPartySoftware`

6. Click `OK` and after a while, observe the network topology in `Net view` and the snapshot images received from cameras in `Time line`. Each image can be saved in JPEG format to the PC's hard drive. The `WPIRMonitor` will also show the last received image in the left hand side of the window.

2.5.7 Display/Cameras Interaction

When the occupancy sensor on a camera (a router or an end device) detects motion, the device makes a snapshot and transfers it to the display (the coordinator), applying the scheme (illustrated in Figure 2-7):

- If an end device is going to send a snapshot, it wakes up the BitCloud stack, restoring its functions, and does not fall asleep until the end of transaction
- The device announces the coordinator that the new snapshot is ready to be transferred by sending the `Data Ready` command
- The coordinator replies with the `Start Sending` command when it is ready to receive the snapshot. The coordinator is able to receive a snapshot from one device at a time
- The snapshot is divided into smaller blocks that are sent in separate data frames. When all blocks are sent the device notify the coordinator that all the transmission is finished with the `Finished Sending` command
- The coordinator may ask to resend blocks via the `Resend Blocks` command if not all blocks have reached the coordinator in the first attempt
- When all blocks are received the coordinator sends the confirmation to the device via the `Stop Sending` command
- If the device is an end device it goes into the sleeping state
- Besides, end devices and routers periodically send `Network Info` commands to the coordinator, providing the coordinator with the updated information about their network parameters, types, software version etc.

Figure 2-7. Camera and Display Interaction During Image Transfer

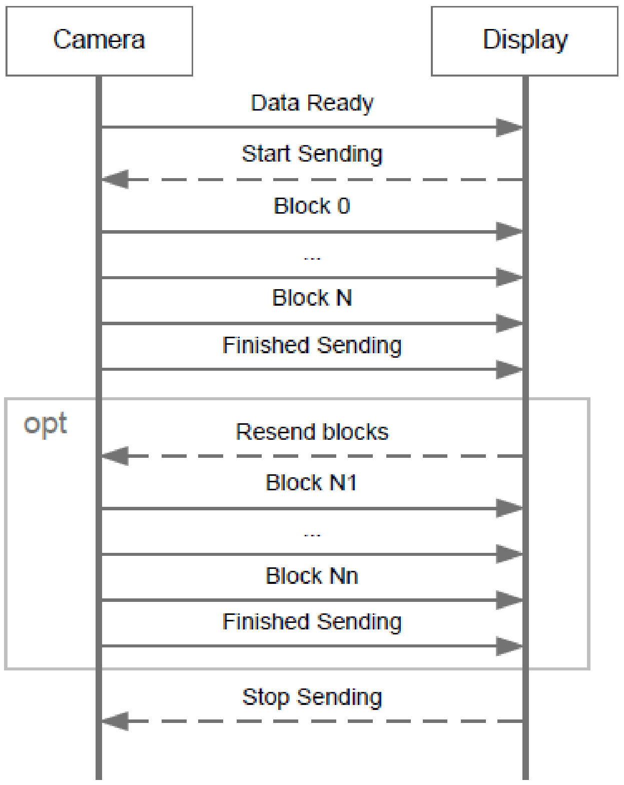


Table 2-8. Display and Camera Communication Protocol

Command	ID	Direction	Bytes
`Network Info`	0x01	Camera to display	See _AppNwkInfoCmdPayload_t definition in WPIRDemo.h
`Data Ready`	0x02	Camera to display	1 – command ID 2 – size 2 – blocks 1 – block size
`Start Sending`	0x03	Display to camera	1 – command ID 1 – response spacing [(1)]
`Sending Finished`	0x04	Camera to display	1 – command ID
`Resend Blocks`	0x05	Display to camera	1 – command ID 1 – response spacing [(1)] 1 – number of blocks 2 – first block index … N-th block index
`Stop Sending`	0x06	Display to camera	1 – command ID
`Single block (data frame)`	N/A	Camera to display	2 – block index variable – block data

Notes: 1. The interval between consecutive data frames (in milliseconds).

2.5.8 Display/PC Interaction

The display (the coordinator of the network) exchanges data with the `WPIRMonitor` PC application through the USB connection.

- The coordinator notifies `WPIRMonitor` about the received image via the `Image Received` command. Note that loading of another image to the coordinator will not start until the current image is transferred to the PC
- The user may initiate the capturing of an image on a particular device, using `WPIRMonitor`. In this case the coordinator receives the `Force Image Capture` command from the PC

Payload formats for the described commands are defined in the `receiver.c` file located in the `/Applications/WPIRDemo/src` directory.

Table 2-9. Display and PC Communication Protocol

Command	ID	Direction
Image Received	0x07	Display to PC
Forced Image Capture	0x08	PC to Display

3. Power Consumption and Frame Rate

All following values were obtained with IAR Toolchain and with the flash_release configuration.

3.1 Single Snapshot

Table 3-1 gives current consumption values versus duration of each main software task. STATE_STAYED macro was enabled, LCD_DISPLAY_INFORMATION (see Section 2.4.2 - Compiling Options) and current measure was done with current probe on JP4 (VDDCORE).

Table 3-1. Power Consumption in Single Snapshot

Function	Duration [ms]	Active current [mA]	Average Current (Current*Time)	Comments
System clock and board initialization	5.58	3.950	22.041 E-06	MCK = from RC @ 4Mhz to PLLB @120MHz with ext XTAL Peripherals: PIOA Components: None
LCD initialization	81	32	2.592 E-03	MCK = 120MHz from PLLB Peripherals: SMC, PIOA Components: LCD
External SRAM initialization	0.0017	29.5	0.05 E-06	MCK = 120MHz from PLLB Peripherals: SMC, PIOA Components: LCD, Ext. SRAM
Capture Initialization	20.7	27	558.9 E-06	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
Capture	67.2	15.4	1.034 E-03	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
Display	45.4	28.4	1.289 E-03	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
Mode backup	N/A	0.001	N/A	MCK: stopped Peripherals: PIOA Components: None
TOTAL			5.495 E-03	

If, for example, 100 pictures are taken in one day, then the average current is
1E-06 + ((5.495 E-03x100)/(24x3600)) = 7.34µA.

3.2 Periodic Motion Detection

Table 3-2 gives current consumption values versus duration of each main software task. LCD_DEBUG and STATE_STAYED macro was enabled (see Section 2.4.2 - Compiling Options) and current measure was done with current probe on JP4 (VDDCORE).

Table 3-2. Power Consumption in Periodic Motion Detection

Function	Duration [ms]	Active current [mA]	Average Current (Current*Time)	Comments
Wait mode	200	0.0225	4.5 E-06	MCK stopped Peripherals: RTT, SMC, PIOA Components: LCD
Sleep mode with PIR, ACC for motion detect	500	0.366	183 E-06	MCK = RC Osc @ 4MHz Core in sleep Peripherals: ACC, RTT, SMC, PIOA Components: PIR, LCD
Switch MCK to PLL	2.36	5.9	13.9 E-06	MCK = from RC @ 4MHz to PLLB @120MHz with ext XTAL Peripherals: SMC, PIOA Components: LCD
External SRAM initialization	0.00169	29.8	0.05 E-06	MCK = 120MHz from PLLB Peripherals: SMC, PIOA Components: LCD, Ext. SRAM
Capture Initialization	20.2	27.8	561.5 E-06	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
Capture	73.5	15.3	1.12 E-03	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
Display	44.4	27.8	1.234 E-03	MCK = 120MHz from PLLB Peripherals: PIOA, SMC, TWI Components: LCD, Ext. SRAM, Image Sensor
TOTAL			3.117 E-03	

If, for example, 100 pictures are taken in one day, then the average current is
22.5 E-06 + ((3.117 E-03x100) / (24x3600)) = 26.1µA.

3.3 Continuous Capture with External SRAM

Details about current consumption for each software task are not given here for this mode. This application demonstrates the capability of the SAM4S to perform “live video capture” when external SRAM is used. In black and white and in color mode the frame rate is 10fps. Source code for Frame rate measurement is provided in the software package.

3.4 Continuous Capture without External SRAM

Details about current consumption for each software task are not given here for this demo. This application demonstrates the capability of the SAM4S to perform “live video capture” when external SRAM isn’t used. The frame rate is 15fps in color mode. Source code for Frame rate measurement is provided in the software package.

3.5 Wireless PIR Image Transfer

The radio transceiver power consumption is measured for various states of the end-device. The measurements were made for the RZ600 module attached to the SAM4S PIRRD board:

- The RZ600 module consumption including the voltage regulator was measured by inserting an ammeter into the circuit at the tenth pin of the J101 connector
- The AT86RF231 transceiver current (ignoring the voltage regulator) was measured by removing the R105 resistor on the RZ600 module and connecting the J103 pins via an ammeter, as instructed in the RZ600 schematic [5]

The results are given in Table 3-3. The second column in the table shows results for the RZ600 module, while the results in the third column are for the AT86RF231 transceiver only. The results in the two columns differ by a fixed value, about 0.4mA, which is the power consumed by the frequency regulator hosted on the RZ600 module.

Table 3-3. Transceiver Power Consumption in an End-device

State	Maximum current consumption (RZ600) [mA]	Maximum current consumption (sole RF chip – AT86RF231) [mA]
Sleeping	0.443	0.000029
Polling for data and network info transmission	9.207	8.721
Image transmission	14.773	14.339

4. Using the Reference Design Board

The default application programmed on the board is the periodic motion detection demonstration. After power up, the screen will display information about demo, 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; a picture is taken and displayed.

If the single snapshot demonstration is programmed into the board, after power-up, pressing the BP1 push button will take a picture.

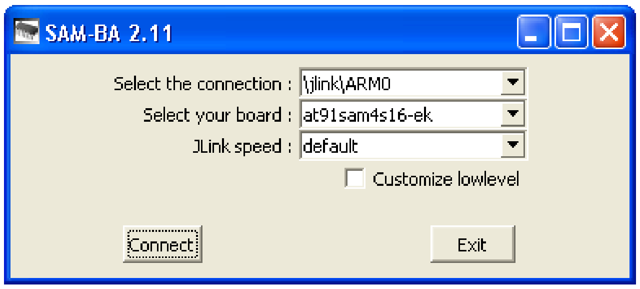
If the continuous capture with external SRAM is programmed into the board, the board acts as a video camera. Pushing BP1 button will switch between Black and White and color mode.

4.1 Downloading and Running the Binaries

All provided binaries files were compiled with IAR Toolchain and with flash release configuration in `bin_wpir_rd_1.1` in the zip package as mentioned in Section 2.4 - Software Deliveries

The binary files can be downloaded directly into the Flash memory using SAM-BA® v2.11 either with Atmel SAM-ICE™ or via USB port (Virtual serial port on device manager). SAM-BA v2.11 CDC version must be used for USB port.

Figure 4-1. Start SAM-BA

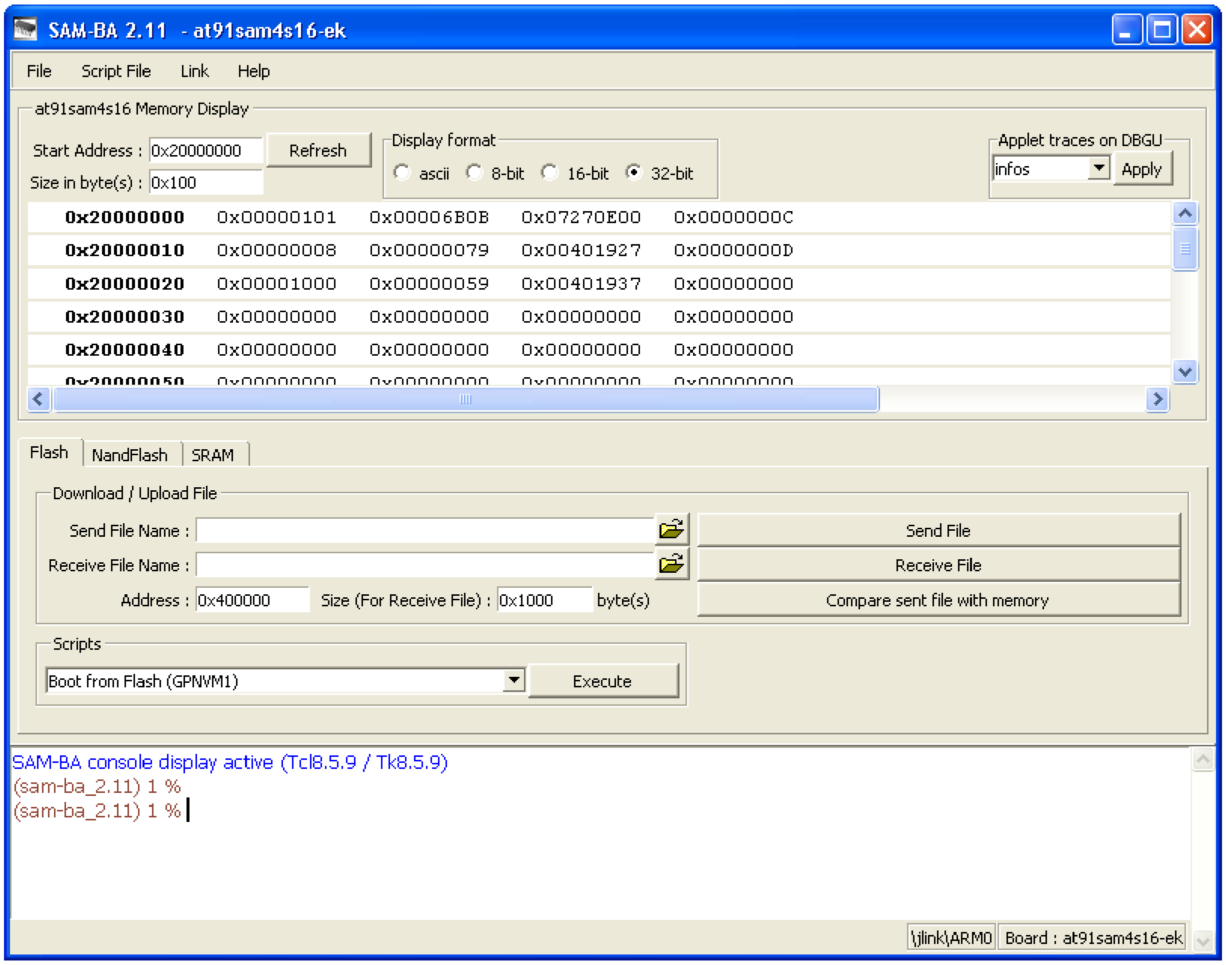


Select the connection : \jlink\ARM0
Select your board : at91sam4s16-ek
JLink speed : default

Choose the connection and the SAM4S16-EK board, and click Connect.

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

- Refresh the start Address
- Enable Flash access script (Scripts drop-down list), 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



at91sam4s16 Memory Display

Start Address : 0x20000000

Size in byte(s) : 0x100

Display format: ascii 8-bit 16-bit [✓ 32-bit]

Applet traces on DBGU: infos Apply

```
0x20000000   0x00000101   0x00006B0B   0x07270E00   0x0000000C
0x20000010   0x00000008   0x00000079   0x00401927   0x0000000D
0x20000020   0x00001000   0x00000059   0x00401937   0x00000000
0x20000030   0x00000000   0x00000000   0x00000000   0x00000000
0x20000040   0x00000000   0x00000000   0x00000000   0x00000000
0x20000050   0x00000000   0x00000000   0x00000000   0x00000000
```

Send File Name :

Receive File Name :

Address : 0x400000 Size (For Receive File) : 0x1000 byte(s)

Scripts: Boot from Flash (GPNVM1)

```
SAM-BA console display active (Tcl8.5.9 / Tk8.5.9)
(sam-ba_2.11) 1 %
(sam-ba_2.11) 1 %
```

\jlink\ARM0 Board : at91sam4s16-ek

The same procedure can be done using the USB port. Choose \USBserial\COMxx connection instead.

Appendix A. 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:

- SAM4S-WPIR-RD_RevA.pdf: Schematics, PDF format
- SAM4S-WPIR-RD _REVA.DSN: Schematics, Cadence® OrCAD® Capture format
- SAM4S-WPIR-RD _REVA.brd: PCB project, Allegro PCB Design
- SAM4S-WPIR-RD _REVA.xls: Bill of Materials

Board manufacturing files:

- GERBER files format

Appendix B. References

[1] Passive Infrared Sensors at: http://en.wikipedia.org/wiki/Passive_infrared_sensor.

[2] SAM4S full datasheet at: http://www.atmel.com/products/microcontrollers/arm/sam4s.aspx?tab=documents.

[3] AT91 In-system Programmer (SAM-BA) opens set of tools, complete software and support documentation at: http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx.

[4] AVR2054 Serial Bootloader Package: http://www.atmel.com/tools/BITCLOUD-ZIGBEEPRO.aspx?tab=documents.

[5] RZ600: http://www.atmel.com/tools/RZ600.aspx.

[6] RF231USB-RD: http://www.atmel.com/tools/RF231USB-RD.aspx?tab=overview.

[7] REB231ED-EK: http://www.atmel.com/tools/REB231ED-EK.aspx.

[8] ASF Sensor Examples Documentation: http://asf.atmel.no/docs/latest/asf_sensors.html

[9] ASF Board Support and Driver Examples Documentation: http://asf.atmel.no/docs/latest/search.html?board=SAM4S-WPIR-RD

Appendix C. Revision History

Doc. Rev	Date	Comments
42159C	04/2014	Modified information on software package
42159B	12/2013	Added information on Wireless Image Transfer Demo
42159A	07/2013	Initial revision

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium 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 G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370