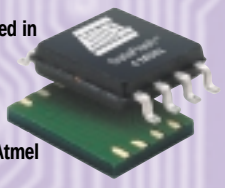
ATMEL APPLICATIONS Number 4 • Spring 2005

JOURNAL

In This Issue:



ATMEL®

Everywhere You Are™

Different architectures from Atmel – one solution from IAR Systems

Secure your investment in application code development and re-use your code in any Atmel device. IAR Embedded Workbench® supports the Atmel AT89, AT91 and AVR families.

IAR Embedded Workbench® for any Atmel MCU
The state-of-the-art integrated development environment with a highly optimized IAR C/C++ compiler and versatile IAR C-SPY™ source and assembly level debugger. Full support for the AT89, AVR and AT91 (including SAM7) families.

NEW IAR Embedded Workbench for 8051 v 6.10 brings the most compact code to all AT89 users. With all powerful new features introduced, this is the most efficient C development tool on the market.

IAR visualSTATE® for any Atmel MCU
The only state machine design tool for embedded systems generating micro-tight C code.

IAR SYSTEMS
From Idea to Target®

Visit us at **www.iar.com**

Download evaluation versions today, and find out for yourself how easily you can develop and re-use your application code across different Atmel platforms.

Introduction

By Jeff C. Katz,
VP Marketing, Atmel

Welcome to Atmel's Application Journal, Vol 4.

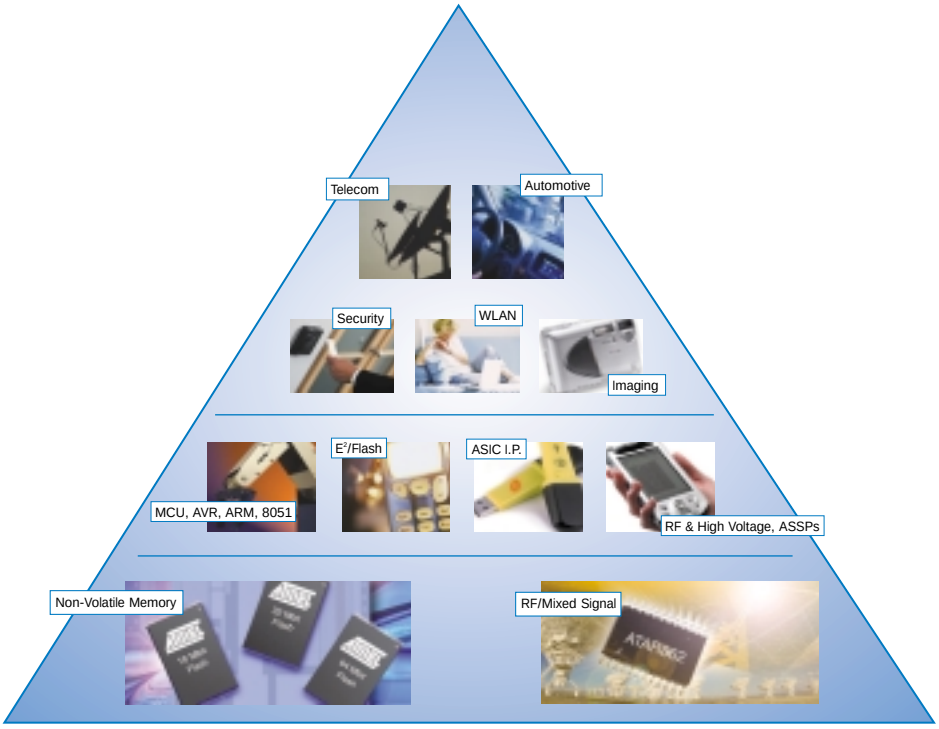
In every issue of the Applications Journal we strive to provide our readers, the electronics design engineering community, with a strong representation of our ability to offer technology for complete systems, including System Building Blocks like our microcontroller lines, Basic Technologies such as non-volatile memory and RF/Mixed Signal, and Complete System Solutions for the leading embedded applications (as exemplified in the illustration below).

Readers of previous issues will notice some subtle changes in the way we are now formatting the Applications Journal: each issue in the future will be segmented into key applications sectors such as Industrial, Consumer & Computing, Wireless, Automotive and Security. Each application section will carry applications articles on creating product designs using Atmel's products and embedded and applications-specific tools and services from our partner's suppliers.

In addition, this issue features articles covering recent initiatives involving Atmel's AVR microcontroller family and some key applications associated to current and future designs taking advantage of the benefits afforded by this IC family. For those looking for more processing power we feature two applications involving our 32-bit ARM support for today's market requirements. We also continue to carry our popular "Designer's Corner" and "Code Patch" sections for our readers who wish to go a little deeper into the content and access the code shortcuts (this time featuring the UNIX/Linux environment) provided in each issue.

We trust that the articles will provide a clear understanding of how you can use Atmel's product line-up to meet your exacting demands within key areas of these application segments such as GPS design, automotive safety, computer and network security and residential/industrial wireless data transfer. Within these areas of end-product design we feature the complementary nature of our product lines and how to get the most from working with us and getting all the support you need, both from Atmel as well as our worldwide tools and engineering support from independent suppliers.

Throughout 2005 and beyond, Atmel will continue to provide you, our customers, with the standard products, application-specific standard products (ASSPs) or customer-specific products (ASICs) in order to allow for a rapid and flexible response to the needs of your customers.



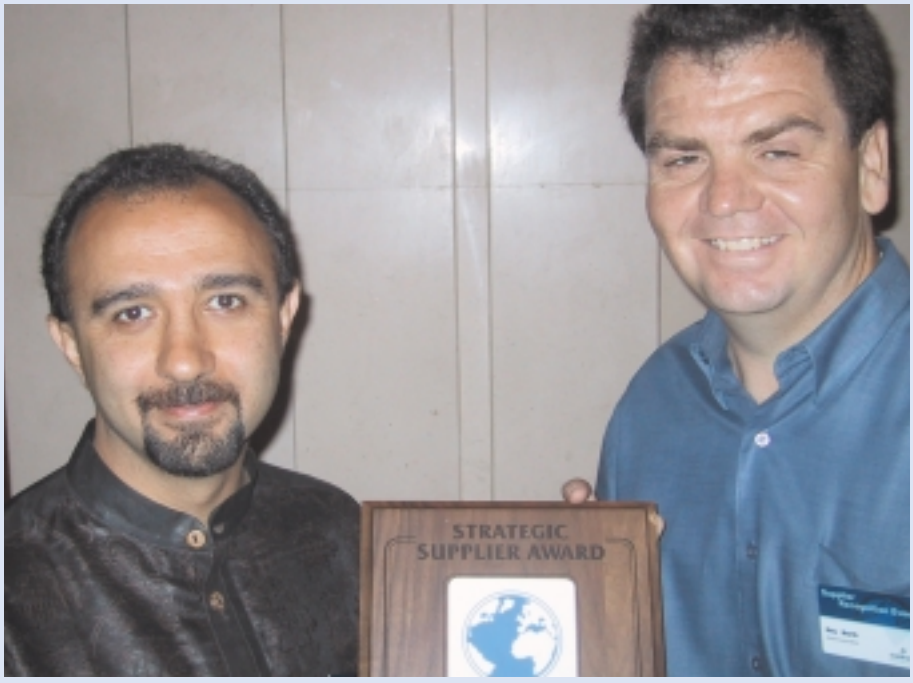
JOURNAL

TABLE OF CONTENTS

Everywhere You Are™

Atmel News:

More information on the products and services in these articles can be found at www.atmel.com



Emerson Corporation Strategic Supplier Award

Atmel was one of two semiconductor suppliers recognized for exceptional Quality, Service and Customer Support. Presenting the award is Ahmad Kian, Director of Strategic Sourcing, Rosemont Process Management Division, Emerson Corporation. Atmel's Ray Barth, District Sales Manager, accepts the award on behalf of the Corporation. This is the second consecutive year that Atmel received this prestigious Award.

WiMAX and Atmel Target Broadband Wireless Market

Atmel has announced its Principal membership in the WiMAX Forum™, an industry-led organization that promotes the interoperability and certification of broadband wireless products based on the IEEE 802.16 standard. As an experienced manufacturer of advanced wireless semiconductors, Atmel will play an important role in the design, production and marketing of silicon to be used in WiMAX Forum Certified™ applications for this key emerging market.

The WiMAX Forum is striving to create economies of scale made possible by standards-based, interoperable products that drive price and performance levels not achievable by proprietary approaches. WiMAX technology is designed to help service providers across global markets deliver economical broadband data, voice and video services to both residential and business customers.

Frank Draper, Director of Membership at the WiMAX Forum, says, "Atmel will be a significant contributor to the WiMAX Forum bringing their vast experience as a wireless IC manufacturer and their unique ability to offer complete out-of-the-box solutions."

Media Processor for Digital TVs and Projectors

Atmel's next generation media processor, the ARM7TDMI™ core-based AT76C120, is ideal for standard and high-definition TVs, flat-panel displays, projectors, portable media servers, and standalone playback boxes. The AT76C120 integrates all the functions required to implement media playback functions onto a single chip. Atmel's Director of Multimedia Products, Tassos Markas, remarked, "With the introduction of the AT76C120 device, Atmel is enabling a new line of consumer products with rich media capabilities. Our goal is to enhance the digital experience and allow consumers to enjoy their media content in their native high resolution. The AT76C120 device has the processing power and performance to just do that."

The AT76C120 device supports high performance still-image and video decoding of media files encoded according to JPEG, MPEG4, MPEG1, Motion JPEG standards as well as of other popular media formats found in digital still cameras. The device contains a high-speed flash card interface that allows quick access to high-resolution images, video and audio streams stored in flash cards by digital still cameras, video cameras, PDAs or camera phones.

Production samples, evaluation boards, software libraries, as well as full application software are available now. The part is priced at $8 for quantities of 100,000 parts.

Atmel Receives 2004 Frost & Sullivan Award for Smart Card IC

Frost & Sullivan announced at the Smart Card Alliance Conference in San Francisco, that Atmel has been presented with the prestigious Frost & Sullivan award for Global Competitive Strategy Leadership, in recognition of its achievements in the smart card IC market throughout 2003. "Atmel's ever-rising market ranking and market shares each year validate its success in the smart card IC industry," said Prianka Chopra, Smart Cards Program Manager at Frost & Sullivan.

New Highly LF Antenna-driver IC for Automotive Tire Pressure Monitoring Applications

Atmel announced at the Convergence Show in Detroit last month the release of a new IC for automotive Pressure-on-Demand (POD) systems in Tire-Pressure Monitoring System (TPMS) applications. POD systems provide the automotive industry's highest flexibility during wheel change and for TPMS system operation. Also, they help to reduce RF noise since the driver stage is placed together with the antenna inside of the wheel well. The ATA5275 features the highest integration for embedded automotive systems and maximum performance. Due to its self-tuning resonance capability, the antenna is able to operate with maximum voltage, maximum current and maximum field strength. The ATA5275 extends Atmel's broad TPMS product portfolio, which already includes UHF transmitter and receiver ICs, LF receiver and antenna driver ICs as well as microcontroller devices to design complete TPMS systems.

Comparable solutions are usually designed as discrete solutions, where the system is not integrated, requiring more design effort, more external devices and a higher bill of materials. The new ATA5275 is the first non-discrete antenna driver IC available on the market, making TPMS/POD designs simpler and more cost-effective. The device is highly integrated, and also provides a microcontroller interface (bi-directional K-line interface) that allows the transfer of diagnostic data.

The new antenna driver IC provides several diagnostic features such as short-circuit, open-circuit and over-temperature detection. The IC is load-dump protected up to 45 V and operates directly from the car battery. The ATA5275's time-out interface supervisor serves to avoid unintended continuous transmission in case of line errors. The temperature range is -40°C/-104°F to +105°C/221°F. Samples of the antenna driver IC ATA5275 in QFN20 packages are available now. Pricing starts at 1.94 US $ (10k).

Trusted Computing Standard Extended to Embedded Systems

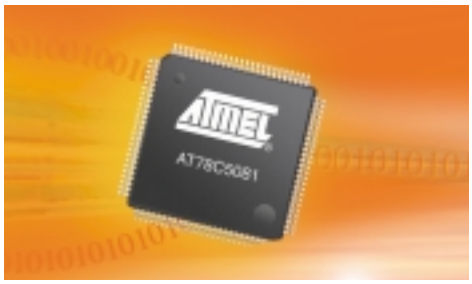
Atmel has announced the AT97SC3201S Trusted Platform Module (TPM), a single-chip hardware security subsystem designed specifically for embedded systems, such as voting machines, gaming systems, PDAs, set-top-boxes, POS terminals, ATMs, portable mass storage devices, and industrial controls. Based on Trusted Computing Group industry standards, the new device offers embedded systems ultra-security that, until now, has only been available for personal computers.

According the Kevin Schutz, Atmel's Product Line Manager for Trusted Platform Modules, "The AT97SC3201S gives embedded systems the capability to provide a variety of software integrity measurements, perform mutual authentication processes, and present credentials that have not been available to embedded system designs before. Now any embedded system can take advantage of hardware security based on TCG standards."

Price and Availability - The 3.3 volt AT97SC3201S is available immediately and is priced at $4.50 in quantities of 10,000.

Two New Physical Layer Chips Target Network Storage, PC and Consumer Electronics Markets

Two new PHY (Physical Layer) stand-alone chips from the Serial ATA (SATA) product family targeted for PC and a range of storage applications such as SATA Host/Device Controllers, Redundant Array of Independent Disks (RAID) and Network Attached Storage (NAS) and SAN Storage Area Networks are now available from Atmel. Other applications include Router Switches that can also be integrated in consumer electronic products such as set-top-boxes. The AT78C5081 is a single port and AT78C5090 is a dual port Serial ATA PHY, that both exceed the parameters necessary to be compliant with the Serial ATA Gen1 revision 1.0 specification therefore providing a very high-speed data transmission at 1.5 Gbps.



"Atmel Network Storage Products BU is dedicated to designing high performance products for PC, networking and consumer products, by making the Serial ATA Physical Layer available as a stand-alone product in the market. We are able to provide more flexibility for new designs in PC and Network products; in addition, to participating into consumer electronic products," said Max Bathaee, Sr. Marketing Manager for Network Storage Products at Atmel.

AT78C5081 is available in TQFP 100-pin package. Pricing is based on volume shipment or $1.95 for 10K and below volume. AT78C5090 is available in QFP 120 pin package. Pricing is based on volume shipment or $2.95 for 10K and below volume. Both products are available in a variety of packages based on customer requirements.

IPWireless and Atmel Partner to Develop UMTS TDD/VoIP Mobile Handsets

Atmel and IPWireless announced that they have collaborated, along with an OEM, to develop a UMTS TDD mobile handset. The mobile handset will allow UMTS TDD network operators to offer carrier-grade mobile voice over IP (VoIP) services, in addition to existing broadband and other packet based services, on their converged networks. IPWireless and Atmel have already completed the first successful transmission of a call from a mobile VoIP handset over UMTS TDD, an important milestone in the development of commercial UMTS TDD handsets.

UMTS TDD, the 3GPP standard optimized for high-speed data, is ideal for carrier-grade voice applications with its high capacity, low latency, and low power requirements. IPWireless' latest UMTS TDD software supports both dedicated and shared channels, enabling operators to deliver circuit-switched quality while enjoying packet-based system economic benefits. Operators can leverage existing network infrastructure to add VoIP over UMTS TDD services in a highly efficient, reliable, and economic manner, thereby increasing revenue while augmenting subscriber services. With a UMTS TDD VoIP handset, subscribers will experience the same quality of voice service they expect from traditional mobile phones. In addition, the handset's USB port allows users to connect it to computers or other devices and use it as a very high-speed broadband modem.

The UMTS TDD VoIP handset will use Atmel's highly integrated AT76C902 VoIP system-on-a-chip as the main processor, with the IPWireless TDD Module performing the UMTS TDD modem functions. UMTS TDD handsets prototypes will be ready by the end of 2004 with commercial availability of phones expected mid-2005.

Web Site Expanded to Support Chinese Language

The Atmel web site, www.atmel.com now supports a Chinese language version. Users can now switch from English to Chinese from any page of the site and smoothly navigate the site in the selected language. Specific information is available in simplified Chinese to introduce the company activity and the various Atmel product lines. Additionally, the majority of AVR® microcontroller datasheets have been translated in Chinese as well as other AVR related technical documentation. Atmel plans to progressively extend translation to other Atmel product literature to support the engineering community in mainland China and Taiwan.

According to Jensan Jen, Sr. Vice President and General Manager, Asia and Japan Operations, "Atmel is a global player in the semiconductor market and our products are very well received in Asia. They are also among one of the favorites in the Chinese engineering community. Although many engineers have the professional level of English reading capability, it is still preferred to see Chinese website content and other technical documentation being translated, in order to easily comprehend and enhance the development efficiency."

We believe that this new web service combined with an increasing support from local technical teams, will significantly strengthen Atmel's presence in China," he added.

First ARM7® Core-based Flash µC Under $3 Introduced at ARM Developers' Conference

At the ARM Developers' Conference in Santa Clara last month, Atmel introduced the world's first family of ARM7 core-based flash microcontrollers with prices starting at less than $3. The product line is targeted at 8-bit applications such as designs with in-system programmable Flash, single supply operation, BOD, POR, RC Oscillator, HW security and USB device.

The AT91SAM7S32 and the AT91SAM7S64 are the first members of a series of Small pincount Smart ARM7 Microcontrollers (SAM7S-series) with Flash densities of respectively 32-kByte and 64-kBytes. A 128-kbyte and 256-kbyte version, the AT91SAM7S128 and AT91SAM7S256, are planned for availability at the end of this year. Besides a jump in real-time performance over 8-bit microcontrollers, they are the world's first 32-bit microcontrollers to integrate a complete set of secure operation functions including a watchdog clocked by an on-chip RC oscillator, power supply monitors and hardware protection of the Flash memory.

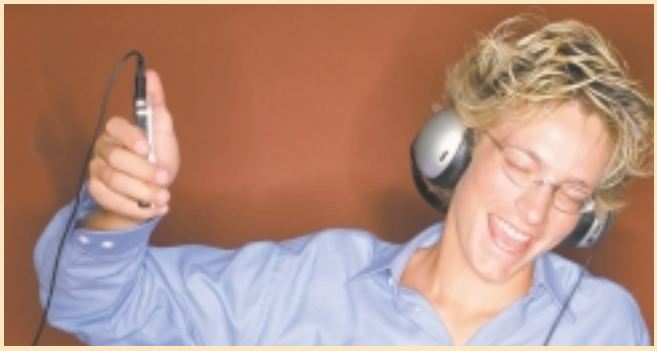
The AT91SAM7 microcontrollers are supported by low-cost development tools, bringing 32-bit real-time processing power to a wide range of cost-sensitive applications that have until now been limited to 8-bit performance. Target applications for the AT91SAM7S series include appliance control, utility metering, security systems, data loggers, USB-based mobile phone and PC accessories.

Ultra Low Power MP3 Decoder for Cell Phones

The AT89C51SND2C is a new MP3 decoder which combines in a single package all features needed in a cell phone to play MP3 ring tones and to convert a cell phone into a pen drive. Integrated in a cell phone, the AT89C51SND2C can easily replace sound generators, while offering MMC/SD card reader capability, MP3 music decoding and USB connection to PC. This provides cell phone manufacturers with more functionality while reducing the system size and costs.

Derived from the AT89C51SND1C, a fully integrated standalone hardwired MPEGI/II Layer-3 decoder, with an 8-bit microcontroller core, handling dataflow and MP3-player control, the AT89C51SND2C offers 64 Kbytes of Flash memory, an embedded MP3 decoder, host communication (serial, parallel, USB) and a Multi Media Card interface. It also includes a high quality stereo DAC with a 50 mW earphone power amplifier, and a 500 mW speaker power amplifier.

The AT89C51SND2C has versatile functionality apart from generating MP3 tones. It can play MP3 music from the cell phone Flash through UART or SPI or from the Flash card (DataFlash®, NAND Flash, MultiMediaCard, Xd-Picture Card, Secure Digital, CompactFlash). It can also upload/download files into the cell phone NAND Flash or NOR Flash, upload files from the MMC to the cell phone NAND or NOR Flash.



"For just one dollar more, the AT89C51SND2C allows the cell phone manufacturers to get rid of the MIDI sound generator while offering to the market a product with extended functionality," said Bernard Bancelin, Marketing Manager of Audio Line at Atmel. "With its compact package, and low power consumption down to 35 mA while playing music at 2.7 V, the AT89C51SND2C offers longer battery life with optimized space," he added.

The AT89C51SND2C is available in a BGA100 (9 x 9 mm) package for industrial temperatures. Pricing for 500,000-unit quantity per year is $6.00. A masked ROM version is available for low cost application at $4.50 for 500,000 units.

A complete development package is available to help designers develop their new applications. It includes an emulator, a development board (AT89DVK-04) and a remote MP3-reference design (AT89RFD-08). All boards come with complete MP3-player firmware, schematics and host drivers.

AT90SC320288RCT High Security, High Performance, Cost Effective Member of the AT90SC Family

Atmel® Corporation introduces the AT90SC320288 RCT, a new member of the AT90SC family featuring outstanding on-chip compact memories: flexible 288-kbyte EEPROM, 320-kbytes of user-ROM, and 8-kbytes of RAM.

an existing product portfolio and offer new products with value-added services, quickly onto the market.

The AT90SC320288RCT can also be delivered with Atmel basic OS. When this basic OS is combined with

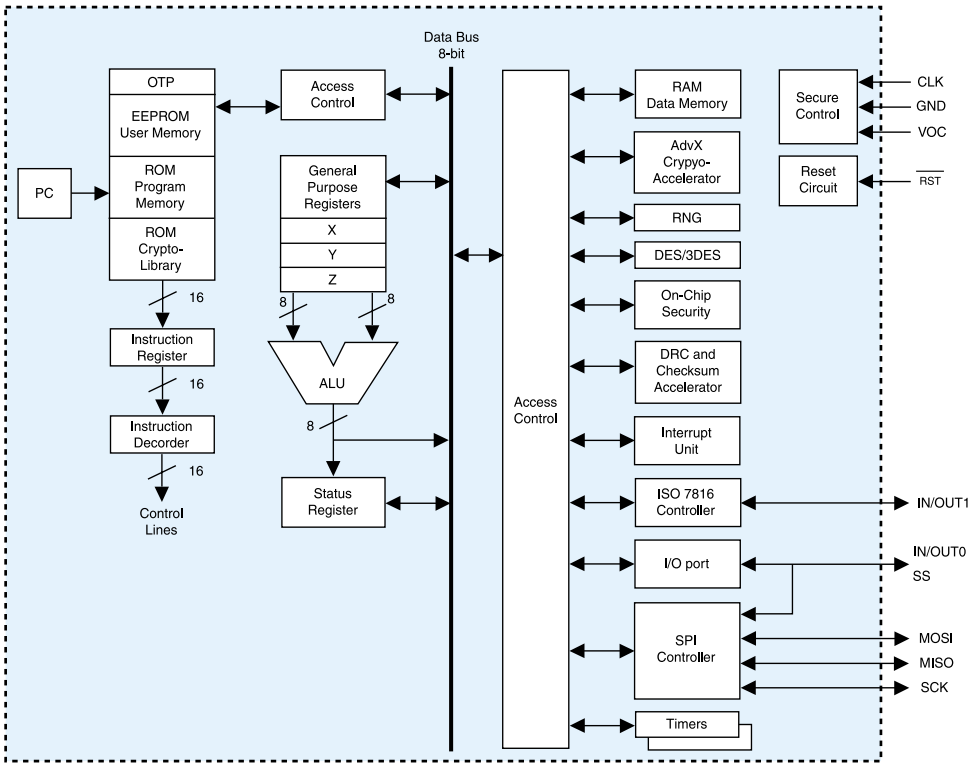


Figure 1: AT90SC320288RCT block diagram

Thanks to the performance and the density of the 8-/16-bit RISC SecureAVR™ CPU architecture, Atmel can propose an attractive product for high density cost sensitive applications such as SIM/USIM cards. Combining advanced technologies and optimized architecture, the AT90SC320288RCT can run at high speed and be compliant with the most stringent industry standards such as mobile phone specifications.

The concept of product family allows developers to reuse existing OS/application software and get immediate access to a product featuring 288-kbytes of EEPROM memory with minimum effort. For example, a developer who has an operating system up and running on 32/36-kbyte of EEPROM (such as the AT90SC19236R/T) or 64/72-kbyte (AT90SC25672 R/T or AT90SC7272C) Atmel platform, can port it, with a very limited amount of resources and effort, on this new product and get access to a 256/288-kbyte EEPROM product. We believe that this will create unique opportunities, for mobile network operators, to extend

the memory management block. It allows program developers to use the 288-kbytes of non volatile memory either as program memory or user memory.

For examples this device can be used as a 144-kbyts Program (Flash functionality) + 144-kbyte EEPROM or 160-kbyte of program and 128-kbyte of EEPROM, or 224-kbyte program and 64-kbytes EEPROM,…

With this configuration, developers can create and test products with high flexibility and introduce products within extremely short time frames.

Targeting high–security markets, the AT90SC 32088RCT is designed to meet Common Criteria EAL5+ and Visa Level3 requirements. Like most of AT90SC SecureAVR products, this device features (but not only) true random number generator, DES/3DES hardware, unique chip ID, memories encryption, Memory Protection Unit, active shield protection, countermeasures against SPA, DPA, glitches and side channel attacks, as well as voltage, frequency, temperature and light protections…

This device also features a SPI interface (up to 12 Mhz). The SPI interface can either be used as additional port to communicate with external device, or to connect an external Flash memory to create a very high secure mass storage device (see figure 2). In that case, data is ciphered by the high security and tamper resistant secure microcontroller (storing encryption keys) and transferred crypted to the Flash memory through the SPI interface.

On the top of the high speed high security DES/3DES engine, the AdvX® high performance cryptographic accelerator and it's embedded firmware (located in an additional 32KBytes dedicated-ROM) supports standard finite fields arithmetic functions including RSA, DSA, DH, ECC, AES.

Atmel hardware emulator "Voyager Emulation Platform "(ATV4) can be combined either with Software Development IAR Systems C-Spy Debugger or Atmel's AVR Studio Version 4.07. Software Libraries (crypto Hardware Abstraction Layers) and application notes are also available. The addition of accurate and complete documentation create a powerful user friendly development environment.

For more information on the secureAVR(tm) product family, please visit Atmel's website at:
http://www.atmel.com/products/SecureAVR

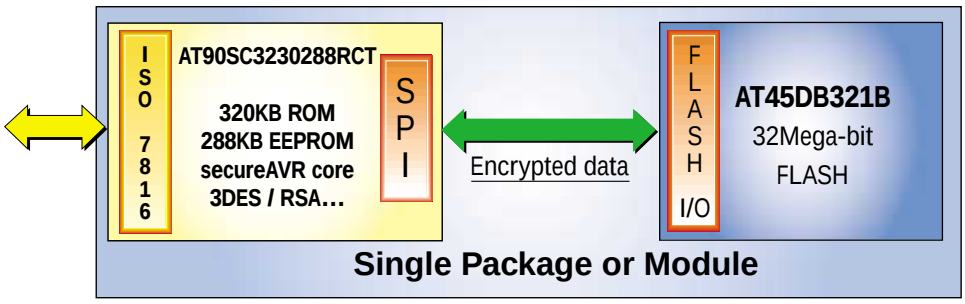


Figure 2: SIP Interface

Atmel's ARM-based Flash µCs Optimized for Real-Time Control Applications

ATMEL®'S AT91SAM7S SERIES OF ARM®-BASED FLASH MICROCONTROLLERS AIM TO BRIDGE THE GAP BETWEEN 8- AND 32-BIT MICROCONTROLLER APPLICATIONS. IN PARTICULAR THEY FOCUS ON REAL-TIME CONTROL APPLICATIONS THAT REQUIRE FEATURES TRADITIONALLY INCORPORATED INTO 8-BIT MCUS (NOTABLY A DETERMINISTIC RESPONSE TO COMMANDS AND INTERRUPTS) TOGETHER WITH THE SUPERIOR PROCESSING POWER AND EXTENDED ADDRESS SPACE OF 32-BIT MCUS. THIS ARTICLE REVIEWS THE GENERAL ARCHITECTURE OF THE AT91SAM7S SERIES AND GIVES DETAILS OF THE UNIQUE FEATURES THAT ATMEL HAS BUILT INTO THESE PRODUCTS IN ORDER TO ENSURE A DETERMINISTIC RESPONSE TO AN EXTERNAL EVENT: A GUARANTEED RESPONSE WITHIN A SPECIFIED TIME INTERVAL.

By: Peter Bishop, Communications Manager, Atmel Rousset

AT91SAM7S Architecture and Real-Time Control Features

As shown in Figure 1, Atmel's AT91SAM7S series microcontrollers are built around a 32-bit ARM7TDMI® processor core and on-chip Flash and SRAM memories. Flash memory densities range from 32K to 512-kbytes. An integrated Flash Programmer facilitates programming (or re-programming) of the Flash memory as required. A memory lock bit, when set, makes it impossible to read the Flash memory by an external device, in order to protect valuable application code and sensitive data.

The System Controller includes a Brownout Detector and Power On Reset controller to ensure power-down and power-up without accidental (or deliberate) corruption of data or code. A Voltage Regulator enables the device to be supplied by a single 3.3 V supply and a Power Management Controller can put the processor and any combination of peripherals into idle mode when they are not in use in order to minimize power consumption.

The rich peripheral set of the AT91SAM7S features a USB V2.0 full speed device for PC connectivity, and a number of peripherals adapted for real-time control. These include a Pulse Width Modulator, several Timers and Timer/Counters supported by on-chip Oscillators and a Phase Locked Loop, and an 8-channel 10-bit ADC. A Parallel I/O Controller multiplexes peripheral I/Os with a set of general-purpose I/O lines in order to reduce the pin count and provide flexibility of external access.

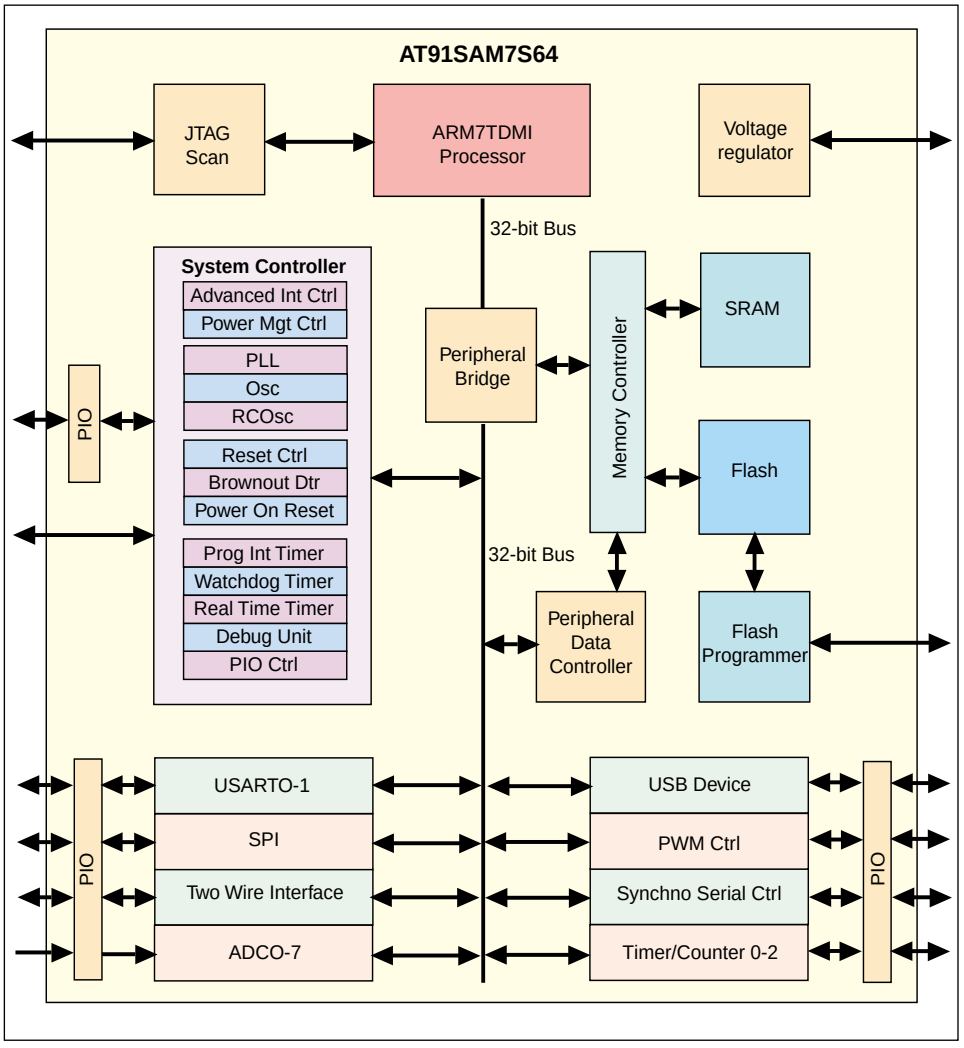


Figure1: AT91SAM7S Architecture

Single-Cycle Code Access from Flash Memory

The limiting deterministic performance of a fast processor running code from memory is established by the single-cycle access time of the memory. (Cache memories cannot be used where deterministic performance is mandatory.) Atmel's integrated high-speed Flash memory is capable of single-cycle access in 33 ns, giving maximum deterministic performance by the ARM7TDMI processor of 27 MIPS. This is a significant improvement over 8-bit MCU performance under deterministic conditions.

Rapid Interrupt Handling

Real-time control applications are typically interrupt-driven, with a response required within a specified time to each external event. Atmel has significantly enhanced the basic interrupt structure of the ARM7TDMI processor in order to provide an interrupt handling capability that meets this requirement. The Advanced Interrupt Controller (AIC) controls the two interrupt lines of the ARM7 processor (normal and fast interrupt). It provides a set of individually maskable, vectored interrupt sources with 8-level priority. One source is for the fast interrupt request, one is for the system peripherals and the remainder are for interrupts from peripherals or external sources. The basic interrupt handler, permanently stored in SRAM, resolves interrupt priorities and then executes an instruction that uses the interrupt number as an offset to load the vector for the required interrupt service routine directly into the Program Counter (Figure 2). This simple, robust mechanism ensures the transfer of control to the required interrupt service routine in the minimum number of instruction cycles.

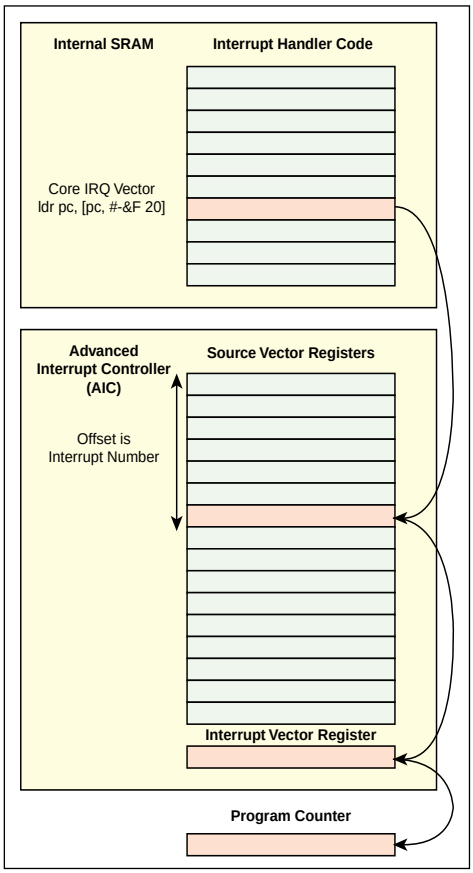


Figure 2: AT91SAM7S Advanced Interrupt Controller

Single-Cycle Bit Set and Reset

Control applications frequently require individual bit set/reset operations, but the basic ARM7 processor architecture requires that this be accomplished by a read-modify-write sequence of operations that requires interrupt masking (itself a bit set/reset operation) to ensure that the entire sequence is carried to completion once it has started. To overcome this shortcoming, Atmel has provided the AT91SAM7S series with an atomic bit set/reset facility that works in the same way for all the peripheral control/monitoring registers.

As shown in Figure 3, complementary Enable and Disable Registers are connected by RS flip-flops to each Status Register. A bit is set in the Status Register by writing 1 to the corresponding bit position in its Enable Register (writing a zero has no effect). Writing a 1 in the Reset register clears the corresponding bit in the Status Register.

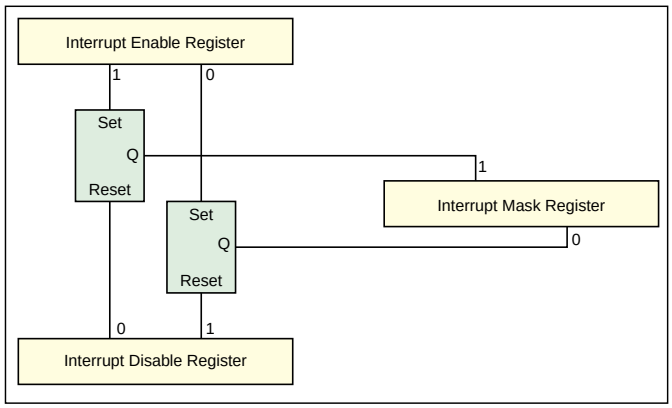


Figure 3: AT91SAM7S Register Bit Set/Reset Mechanism

This provision, commonly found in 8-bit microcontrollers, reduces bit manipulation to a single instruction executed in a single clock cycle. It reduces code size (by approximately 60% for the instruction sequence concerned) and speeds program execution.

Peripheral Data Controller for Direct Memory Access

Many of the AT91SAM7S peripherals are designed to transfer blocks of data between SRAM and an external communications channel (USART, Debug Unit, SSC, SPI and ADC). In order to carry out these transfers as efficiently as possible, Atmel has incorporated a Peripheral Data Controller (PDC) into the AT91SAM7S architecture that provides a direct memory access (DMA) function.

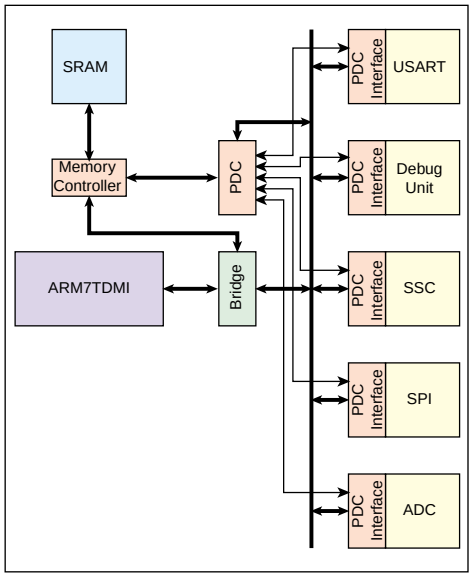


Figure 4: AT91SAM7S Peripheral Data Controller

The PDC (Figure 4) manages the transfer of blocks of data between SRAM and these serial peripherals with minimum processor overhead. On the one hand this reduces the programming requirement for data transfers to writing or updating the contents of the appropriate PDC Interface registers, and on the other hand the transfer time for blocks of data is considerably reduced. The transfer of successive blocks can be chained to avoid generating unnecessary interrupts until the entire transfer is complete. These features significantly enhance the real-time performance of the AT91SAM7S.

Conclusion

The AT91SAM7S series incorporates a number of architectural features that bring the real-time control capabilities of 8-bit microcontrollers into the 32-bit arena with hardly any price adder. Most of these features are Atmel enhancements to the basic ARM7TDMI architecture, with the common aim of providing deterministic performance at a clock speed and data throughput significantly higher than can be achieved with 8-bit microcontrollers. At the same time these devices achieve a commendably low power consumption while driven from a single-voltage supply, and protect valuable application code and reference data.

Further information can be obtained from Atmel's Web site at www.atmel.com.

Reconfigurable System on a Programmable Chip Platform

THIS PAPER PRESENTS A UNIVERSAL RECONFIGURABLE SOPC PLATFORM BASED ON A COMBINATION OF THE ATMEL AT94K FPSLIC DEVICE AND AN EXTERNAL MEMORY. THE PRESENTED PLATFORM INCREASES THE POWER OF THE FPSLIC DEVICE BOTH BY EXTENDING THE INTERNAL ADDRESS SPACE THROUGH AN INTRODUCTION OF A VIRTUAL PROGRAM MEMORY AND BY PROVIDING A TRANSPARENT INFRASTRUCTURE FOR FPGA RECONFIGURATION. THE PLATFORM IS DEMONSTRATED ON TWO SIMPLE DESIGNS THAT DEMONSTRATE BOTH ASPECTS.

By: M. Danek, P. Honzık, J. Kadlec, R. Matousek, Z. Pohl of the Institute of Information Theory and Automation, Dept. of Computer Science & Eng. and Centre of Applied Cybernetics*

Introduction

Field-programmable gate arrays are configurable VLSI devices that can implement various logic functions. Classical SRAM-based FPGA chips introduced in 1984 were designed to be configured only once at the beginning of their operation and to enable a designer to improve the functionality after a device was shipped to the end user.

Now, almost two decades later, the current FPGA technology includes the concept of reconfigurability. The reconfigurability has to be looked at from two levels. One level is the actual dynamic reconfigurability of a device, while the other is a support built into the CAD tools supplied for the device. At present, devices that support limited or full version of limited reconfigurability are available, for example, the recently introduced devices from Xilinx (Spartan2, VirtexII) or Atmel (AT40K, AT94K), but it has not been incorporated into the supplied design tools [3]. The reason for the latter fact is the rather complicated methodology that has to be implemented in the tools [6].

This paper describes a universal reconfigurable SOPC platform built around the Atmel AT94K FPSLIC1 device. The platform enables a software programmer to take advantage of a possible hardware implementation of a user function while not being directly involved in the hardware design process. This is made possible by providing a universal transparent hardware-software interface and a library of pre-compiled configuration bitstreams that are selected by the programmer according to the required hardware functions.

The platform implements an extended virtual memory space that increases the standard 64KB address space that is accessible by the FPGA and AVR processor inside AT94K. The extended memory space can be used for AVR programs, data, and FPGA configuration bitstreams, which increases the power of the AT94K device [5]. Similar architectures can be found in [1], [2].

The paper is structured in the following way: Section 2 describes basic features of the AT94K device, Section 3 presents basic features of the presented platform, Section 4 shows the use of the platform on two sample applications, and Section 5 concludes the paper.

AT94K FPSLIC

The Atmel AT94K device (see [11] and Figure 1) consists of an AT40K FPGA [12], an 8-bit AVR microcontroller, a hardware multiplier, two UARTs, a two-wire serial port (TWS), three counters, a watchdog timer, and a 36KB SRAM memory that is partitioned to a 20-32KB AVR program space and a 4-16KB AVR data space. Prototyping boards based on this device are available from Atmel (ATSTK94, ATSTK594).

The AVR microcontroller can be programmed both in assembler or in C compiled, for example, by the Imagecraft C compiler or the free AVR gcc compiler.

The FPGA can be programmed by bitstreams generated by the Atmel-supplied Figaro place & route tool in combination with common design synthesis tools (Mentor Graphics Leonardo Spectrum, Synplicity Synplify, etc.) [8], [9].

The AT94K device can implement both pure AVR or pure AT40K designs or their combination. The standard AVR-FPGA data interface consists of 8 data inputs, 8

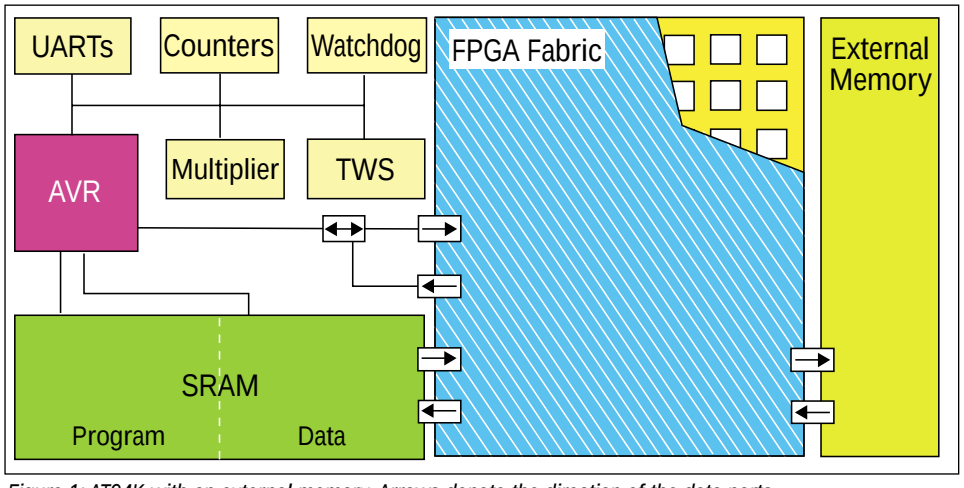


Figure 1: AT94K with an external memory. Arrows denote the direction of the data ports.

* This work has been partially supported by the EU IST programme under No. IST-2001-34016 and by the Grant Agency of the Czech Republic under No. 102/04/2137.

[1]Field-Programmable System Level Integrated Circuit

data outputs, 16 FPGA select lines generated by the AVR, and 16 AVR interrupt lines generated by the FPGA. The FPGA can access both the AVR program area and AVR data area of the built-in SRAM. In addition, the AVR can reconfigure the FPGA using Mode 4 bitstreams and four FPGA configuration registers (FPGAX, FPGAY, FPGAZ, FPGAD) [10], [11].

AT94K alone is suitable for simple tasks that do not deal with large amount of data and that do not require high data throughput. Fortunately the AT94K target is supported by available C compilers, so programming the AT94K is a matter of routine C programming and FPGA design.

Its limits are mainly the size of its internal memory and the rather small size of the built-in AT40K FPGA. The internal memory is 36KB, out of which max. 32KB is available for program code and max. 16KB for user data. In the best configuration the built-in FPGA contains 50K equivalent gates. Also, the available C compilers do not provide any sophisticated interface to access the FPGA fabric.

Extending the AT94K Power

To increase the capabilities of AT94K it is necessary to increase mainly the real size of the program memory by implementing a virtual memory and the size of the data memory that stores FPGA bitstreams. Notice that a common size of a simple compiled AVR code is about 10KB; a common size of a moderate size bitstream in Mode 4 [10] is well over 100KB.

Both problems can be solved by using an external data memory. The proposed platform (Figure 2) uses an external FLASH memory to store fixed program and bitstream data that are copied at diferent times to specific locations of the internal SRAM memory. The FPGA is divided to a static part that implements an access to the FPGA reconfiguration data and the AVR program code, and to a dynamic part that contains user-defined designs; the designs can be reconfigured at run time.

Data Sharing between AVR and FPGA

Before the AT94K FPGA can be used as an AVR coprocessor, it is necessary to define data exchange schemes between AVR and FPGA. There are two possible schemes: to use registers implemented in the FPGA, or to use the internal SRAM.

The width of the FPGA registers is limited by the 8-bit AVR-FPGA data bus. The advantage of this scheme is its simplicity, a major drawback is its slow transfer rate: each AVR load or store instruction requires 2 clock cycles.

The SRAM exchange scheme is faster, since the FPGA can implement a DMA controller that can access the memory directly without an AVR support in one clock cycle, which saves at least 4 clock cycles per operation (when processing one 8-bit data value at a time). Its disadvantage is the necessity to implement a data sharing or protection mechanism; in its simplest form it may be represented by a convention that a given memory area must not be modified by the AVR until the FPGA has finished the calculation.

Virtual Program Memory

The virtual program memory is often implemented using program overlays or memory paging. Since AVR is intended for simple tasks, we think that program overlays without memory protection are sucient. The following text considers 2KB long overlays, since this size seems adequate for AVR application programs. The data memory can be extended using a similar concept.

The implementation uses the fact that the program memory can be accessed as data by the FPGA by setting bit 36 in the FPSLIC system control register [11]. The implementation consists of two parts: a software overlay support contained in the AVR BIOS, and a hardware DMA controller implemented in the FPGA that transfers overlays from an external memory to a given location in the program SRAM. The DMA controller consists of a context register that determines the starting address (its MSbits) of the overlay in the external memory and of a 12-bit counter that generates the LS address bits of the overlay address. A write to the context register initiates a data transfer. When the counter overflows, it means that the transfer has been completed and the controller generates an interrupt.

Since a sequence of locations as accessed by the AVR in the program memory maps to diferent clusters of locations in the data memory, it is necessary to translate the compiled overlay code. It is convenient to perform this translation at compile time, since the resulting data can be just streamed from the external memory to the internal SRAM memory by the DMA controller each time an overlay is required and no run-time address translation is necessary.

The overlay service is accessed through a BIOS function call with a parameter that identifies the required overlay. This parameter is translated and stored to the context register. The BIOS function ensures that on completion the memory contains a valid overlay, a jump to its starting address is then performed in the application program.

Programming Overlays in GCC

The overlay concept requires that a common C program is transformed to several functions with a common start function, all contained in one overlay that is brought into the AVR program memory by the AVR BIOS on demand. The start function is similar to the main function in C. From the start function another sub-functions are called that are not visible from the AVR BIOS. The AVR BIOS can download a particular overlay to the memory, and give control to the start function in the actual overlay. The function for changing overlays is a loop that checks the ID of the actual overlay in the memory and compares it with that of the required overlay. When a diferent overlay is requested, it must be downloaded from the external memory (in our case, an external flash memory is used to store overlays). The overlay transfer process is carried out by the FPGA logic; when a user program calls a particular overlay, the call initiates an FPGA-controlled DMA transfer. On completion the FPGA generates an interrupt.

The AVR BIOS does not write into the program space or call any function from the overlay memory. The AVR BIOS knows only the overlay starting address; an overlay is executed by a function call to that address. Each overlay needs to communicate with the AVR BIOS. To do this a service table is located at a fixed address in the AVR BIOS. The service table describes calling addresses of services that are provided by the AVR BIOS. The services include registering an interrupt, or starting a timer. This organization ensures the independence of overlays; the only implementation-dependent part of an overlay is a header file with the definition of the service table.

FPGA Run-Time Reconfiguration

Run-time reconfiguration enables the designer to increase the functional density of the FPGA coprocessor, in other words, to download functions to the FPGA as they are needed. AT94K supports four configuration modes, out of which the simplest reconfiguration mode is Mode 4 (simplest = principally does not require an external hardware). Mode 4 reconfiguration is based on four configuration registers serviced by the AVR [10], [11].

The most convenient implementation of the extended bitstream memory uses one FPGA register for context (bitstream) selection and another for data passing. The static part of the FPGA implements an address register that consists of the context register (MS bits) and a counter. On writing to the context register the counter is reset. Each time the data register is read by the AVR the counter increments. When the top address specified in the bitstream header is reached, i.e. when the reconfiguration of the dynamic part is finished, the FPGA interrupts the AVR operation.

When not considering different execution time due to reconfiguration, the reconfiguration process is transparent to the application software. The access to the FPGA coprocessor is implemented as a special BIOS function, whose parameters are the required operation and its operands passed either as direct values in the case of the register transfer, or as a starting address of their location and their number in the case of the SRAM transfer. When BIOS detects a request for an operation different than the one currently present in the FPGA, it calls a function that reconfigures the FPGA.

This function first translates the requested operation to the context (address) of the corresponding bitstream and writes it to the context register. Then it sequentially reads 4-tuples of values and writes them to the four FPGA configuration registers (X, Y, Z, D) until the FPGA generates an interrupt.

Generation of Reconfiguration Bitstreams

The runtime reconfiguration of the AT94K FPGA fabric requires partial bitstreams. Such bitstreams reconfigure only a part of the chip while the rest is not a ected in its operation. The Figaro design implementation tool provided by Atmel is meant to generate bitstreams that are not intended for partial reconfiguration of the FPGA. A special implementation procedure must be used to obtain such bitstreams [3]. The idea is to get several complete bitstreams with all diferent coprocessor contexts that contain the same placement and routing of the identical static part.

The Figaro tool works with a system of libraries. Any design component can be implemented as a macro and stored in a named library. The top-level design may contain instances of such components as black boxes (i.e. without a description of their content). Figaro will then search project libraries for components that fit the instance interfaces.

This feature is used in the described approach to implement and store diiferent contexts of the reconfigurable coprocessor in libraries with diferent names. Since all contexts are implemented, the top-level design can be opened in a new project that includes only one of the context libraries. A complete bitstream is generated by performing all implementation steps.

The next complete bitstream with a different coprocesor context must contain the same placement and routing of the identical static part. This is achieved by opening the same project under a different name with the placement and routing locked. Before the project is re-opened, the context library must be changed to the next coprocesor context. The Figaro tool then detects that the coprocessor has changed and it reimplements only the coprocessor without any modifications in the locked static part. This procedure is repeated until all coprocessor contexts have been generated.

To obtain partial bitstreams the complete bitstreams obtained in the previous step must be compared using the Figaro bitstream compression tool. The tool generates incremental changes that must be performed to switch from the configuration given by the base bitstream to the configuration given by the new bitstream. To be able to use the partial bitstreams for changing the coprocessor configuration all possible coprocessor context swap combinations must be generated. A direct approach leads to n! combinations (each to each), where n is the number of contexts. A significant reduction of combinations is obtained by introducing a common reference coprocessor configuration, such as an empty contents or the most frequently used function; this approach decreases the number of necessary bitstreams to 2*n*.

Sample Applications

The described platform (Figure 2) is presented on two simple applications: the first shows the use of overlays, the second demonstrates a transparent run-time reconfiguration of the FPGA triggered by the application program.

Program Overlays

Program overlays are user-defined functions compiled as a standard C code with a specific starting address, here denoted as *START_PROGRAM* (Figure 3). The overlays are implemented as data transfers from the external memory to specific locations in the program memory. The *START_PROGRAM* value has to be set in accordance with the starting location used by the FPGA DMA controller. Once the overlay has been loaded, the application code performs a jump to the *START_PROGRAM* location (appl is a pointer to a function). When the overlay finishes, the control is returned to the main application, which can load and execute another overlay.

The *LoadOvly* function implemented in BIOS translates the overlay ID to its context in the external memory (its MS address bits) and stores it to the *VIRTUAL MEM CTRL* context register (Figure 2). This write initiates a DMA operation (implemented in the FPGA) that transfers a 2KB block of data from the external memory to the AVR program SRAM. Once the transfer is completed the FPGA generates an interrupt.

```
if (LoadOvly(ovlyID) != TRUE)
 PrintStr("OVERLAY LOAD FAILED\r\n");
else {
 appl = START_PROGRAM;
 appl();
 }
```

Figure 3: An overlay use in an application program.

```
len = LoadData(&oper,data);
i=0;
while (i<(len/6)) {
 CallFPGA(oper,
    data[6*i],data[6*i+1],data[6*i+2],
    data[6*i+3],data[6*i+4],data[6*i+5],
    &results[3*i],&results[3*i+1],
    &results[3*i+2]);
 i++;
 }
```

Figure 4: An FPGA call in an application program.

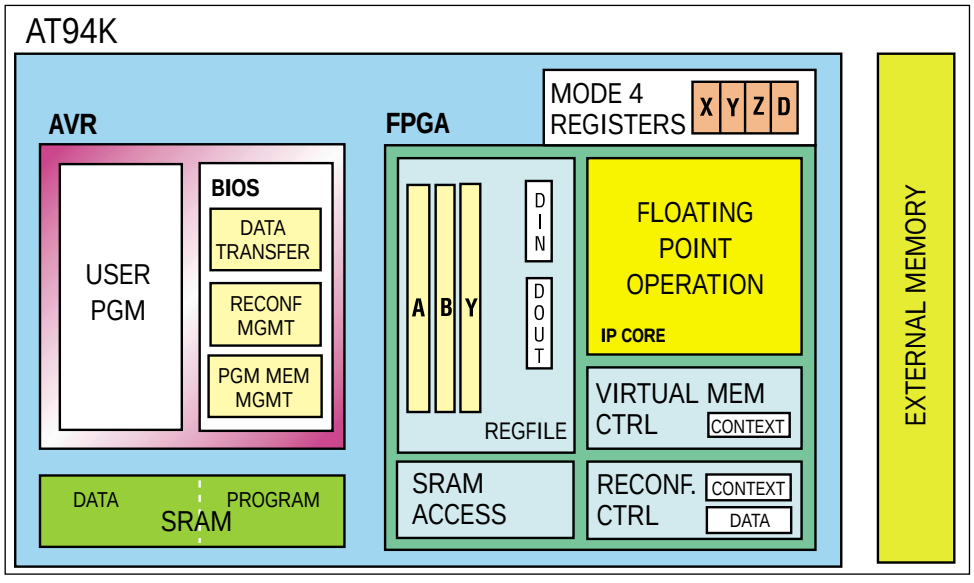


Figure 2: The AT94K-based reconfiguration platform.

Reconfigurable Coprocessor

The reconfigurable coprocessor implements different operations executed in the FPGA. The coprocessor can be accessed by an application program executed by the AVR microcontroller.

In the described application the coprocessor implements basic floating-point operations - ADD, MUL, DIV, SQRT - in a 24-bit precission (1-bit sign, 6-bit exponent, 17-bit mantissa). As the AT40K40 FPGA is too small to contain all the operations at once, the coprocessor is reconfigured prior to calculation when necessary.

The performed task is simple: the AVR gets a data block from the serial port, then it uses the FPGA floating-point coprocessor to calculate results for these values.

The design consists of four main parts (Figure 2):
1. the floating-point coprocessor implemented in the FPGA, denoted as *IP CORE*,
2. the data management, i.e. the data transfer part of the AVR BIOS and the *REGFILE* and *SRAM ACCESS* blocks in the FPGA,
3. the virtual memory management, implemented in the *PGM MEM MGMT* part of the BIOS and the *VIRTUAL MEM CTRL* block in the FPGA,
4. and the reconfiguration controller, shown in BIOS as *RECONF MGMT* and in the FPGA as *RECONF CTRL*.

A sample application program built upon the infrastructure is shown in Figure 4. The code first gets a data block, then it calls the FPGA to process the data. The first item in the block determines which operation should be performed, this is stored in the *oper* parameter.

The FPGA infrastructure is transparent to the programmer. The only place at which the AVR-FPGA interaction can be noticed is the function call CallFPGA. Each operand in the example is encoded using 24 bits; the amount of parameters passed to the function is determined by the 8-bit architecture of the internal AVR-FPGA data bus. An additional parameter called oper parameter is passed to the AVR BIOS to determine if it is necessary to reconfigure the FPGA. The BIOS then transfers the data to the coprocessor and gets the results.

Conclusion

This paper has presented a hardware platform that makes use of dynamic reconfiguration to increase a computational potential of a simple microcontroller. The platform uses an external memory to emulate a bigger program and data space, and to store reconfiguration bitstreams needed to program an FPGA-based accelerator connected to the microcontroller. Two sample applications have demonstrated the transparency of this approach in terms of application programming.

The most important advantage of the presented approach is the possibility to offer an unlimited number of different operations accelerated in the FPGA to an application programmer without his concern for digital circuit design.

Acknowledgements

The authors would like to gratefully acknowledge the support for this work provided by Atmel Hellas and Atmel Nantes.

References

[1] Collahan, T. J., Hauser, J. R., Wawrzynek, J.: The Garp architecture and C compiler. In IEEE Computer, vol. 33, iss. 4, 2000, pp. 62–66.

[2] Horta, E. L., Lockwood, J. W., Taylor, D. E., Parlour, D.: Dynamic hardware plugins in an FPGA with partial run-time reconfiguration. In Proceedings of the Design Automation Conference, 2002, pp. 343–348.

[3] Matousek, R., Danek, M., Pohl, Z., Kadlec: Dynamic runtime partial reconfiguration in FPGA. In ECMS2003, Liberec, 2003. pp. 294–297.

[4] Matousek, R., Pohl, Z., Danek, M., Kadlec, J.: Dynamic reconfiguration of FPGAs. In 10th International Workshop on Systems, Signals and Image Processing (IWSSIP'03), 2003, pp. 288–291.

[5] Wirthlin, M. J., Hutchings, B. L.: Improving functional density using run-time circuit reconfiguration. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 6, Iss. 2, 2002, pp. 247–256.

[6] X. Zhang, Kam W. Ng: A review of high-level synthesis for dynamically reconfigurable FPGAs. In: Microprocessors and Microsystems, No. 24 (2000), pp. 199-211.

[7] FPSLIC User's Guide and Tutorials, Revision 5 (Compatible with SystemDesigner 2.1), shipped with the ATSTK94 starter kit. Atmel, 2001.

[8] FPSLIC System Designer 3.0 User Guide. Atmel, 2003.

[9] Atmel FPGA Integrated Development System. Atmel, 2001.

[10] FPSLIC on-chip Partial Reconfiguration of the Embedded AT40K FPGA. Atmel, 2002.

[11] AT94K Series Field Programmable System Level Integrated Circuit. Atmel, 2002.

[12] AT40K 5K to 50K Gates Coprocessor FPGA with FreeRAM. Atmel, 2002.

Practical Aspects of ARM Core-based Product Development

ARM CORE ASIC PRODUCT DEVELOPMENT IS POSSIBLE BY THE MASSES, AND ARM CORE-BASED STANDARD PRODUCTS OFFER PROVEN PATHS TO ASIC SOLUTIONS. RE-USABLE/RE-CONFIGURABLE ARM PERIPHERAL IP IS A REALITY. ASIC SYSTEM SOFTWARE TEAMS CAN TAKE ADVANTAGE OF ARM STANDARD PRODUCT SOFTWARE, AND ATMEL'S SILICONCITY PLATFORM DESIGN APPROACH ENABLES FIRST TIME SUCCESS SILICON WITH REDUCED DEVELOPMENT LEAD TIMES.

By: Jim Hallman, ASIC Design Manager, Atmel Corporation

ASIC development of embedded ARM microcontroller core systems often exceed engineering estimates and management expectations. One might conclude that only experts should attempt embedded ARM core-based ASIC designs. Solutions have evolved to reduce product development cycle time, cost and risk while avoiding unexpected pitfalls along the way. ARM core-based system-on-a-chip (SoC) products, should no longer be considered only by elite ASIC designers with lots of time and big budgets.

Bringing an ARM core-based SoC product to production involves more than the hardware implementation of an ASIC concept. Full system considerations, including software development, must be examined when a microcontroller is embedded inside the silicon. It is now practical to develop system software in parallel to SoC development (Figure 1), enabling product release to production soon after hardware silicon samples are reviewed.

Many ARM7 and ARM9 products have a common set of featured peripherals to complement the chosen core. General-purpose standard products address a wide

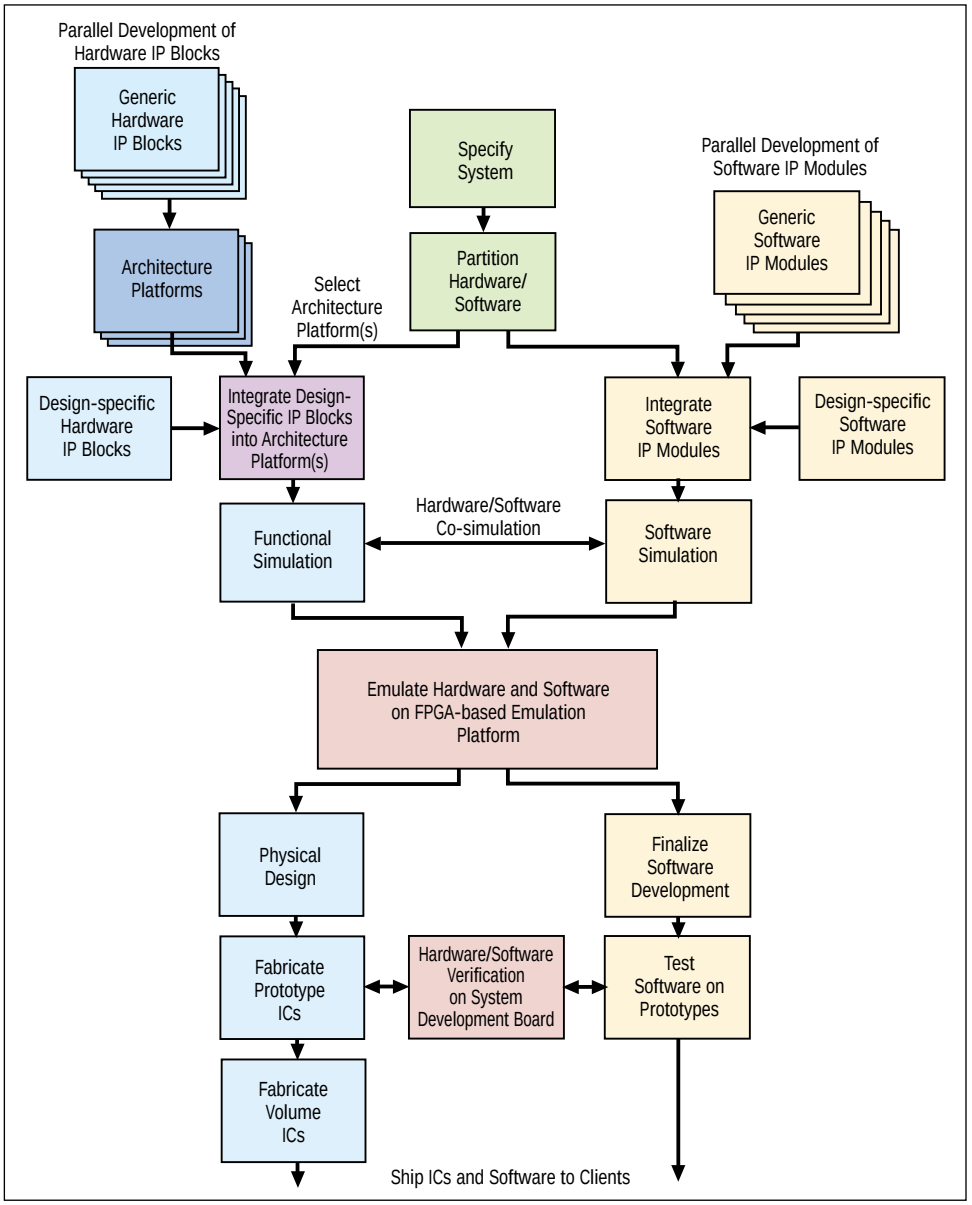


Figure 1: Parallel Development of SOC Hardware and System Software

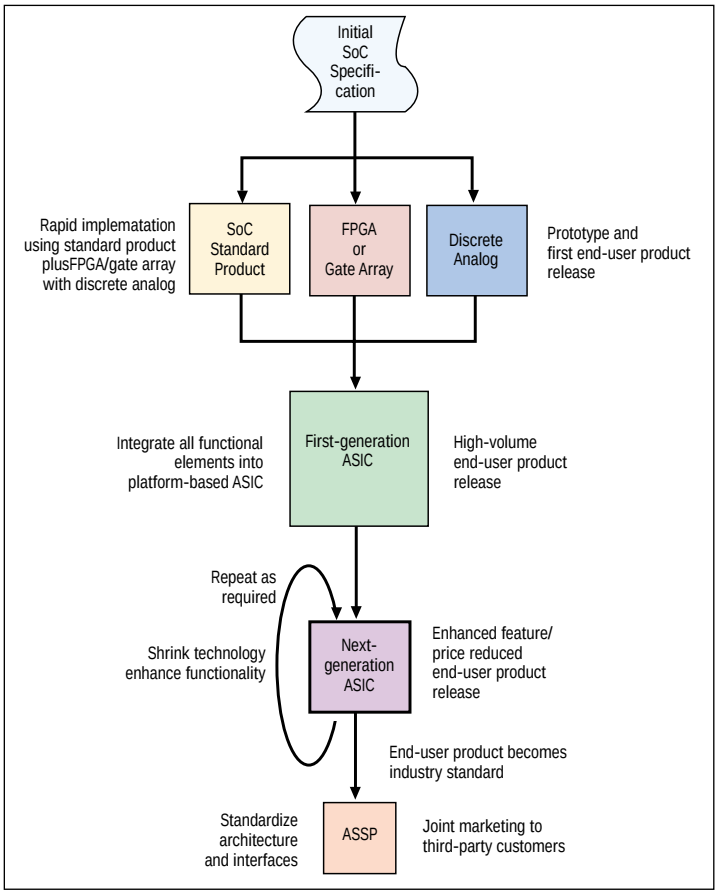


Figure 2: Product Evolution

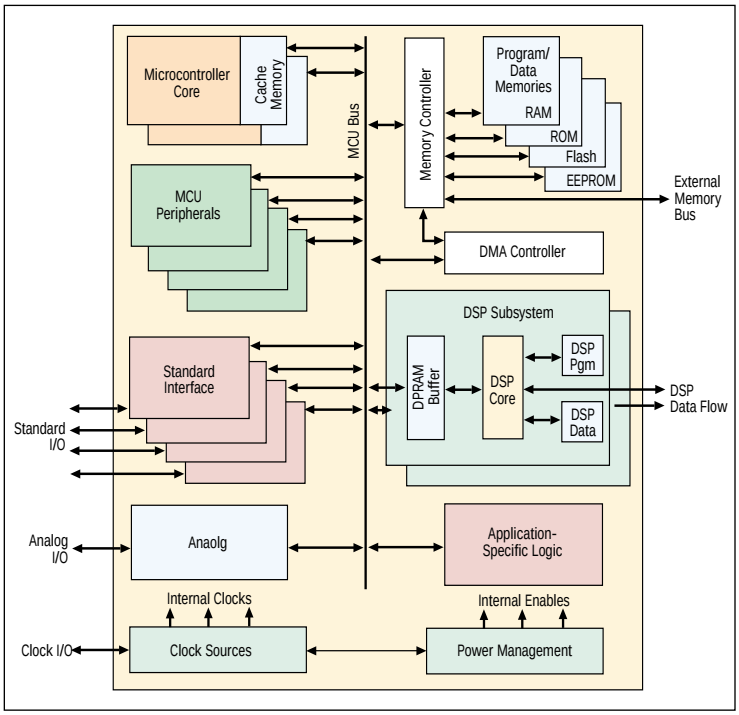


Figure 3: SoC Architecture Platform

application market, which means many features of the product go unused, even though paid for in silicon area. Systems typically want to use a large percentage of a selected standard product with the addition of application specific intellectual property (IP). This IP differentiates the end product from the competition. It is a natural progression for designers to begin a product development with a standard ARM product, complemented by an FPGA containing application specific IP or by discrete analog components. The product may even evolve into an application specific standard product (ASSP) marketed by the originator and the ASIC vendor (Figure 2). Atmel's approach to development, branded Silicon CITY™, encompasses standard product, ASIC and ASSP evolution into one common design and development environment. Atmel's SiliconCITY architecture platform and IP libraries permit migration of the utilized standard product components, application IP and analog into a single ASIC chip in a short period of time.

A key factor to reduce risk and design lead-time is to start development from validated and proven IP. Who wants to re-invent the wheel? SiliconCITY allows SOC designers to start ASIC product development from an SOC architecture platform netlist derived from a standard product netlist. Or they can start with a collection of IP building blocks used to create a standard product and configure their system netlist locally. Atmel introduces new ARM7 and ARM9 standard products through this same philosophy. An added advantage is the portability of products across multiple ASIC technologies.

For an architecture platform design system (Figure 3) to be real, it must be flexible enough to address a wide range of user requirements. IP within the system must be configurable for multiple methods of use. However, flexibility introduces complexity from a synthesis and verification perspective. This can translate into time, diminishing the advantage of a platform-based design system. Highly parameterized IP is the answer. Atmel's IP has been designed not only for direct re-use, but also for expandability. Every component of the system uses parameterized IP in a common approach. A clearly defined data structure, common clock and test approach with global system constraint files, insure a consistent design methodology across selected IP. IP parameters extend into synthesis scripts and test benches to assure automated verification for each configuration.

Why emphasize IP parameterization? To dramatically reduce development lead times, IP must be configurable for use in multiple applications, while retaining a robust verification methodology. Parameterization allows bus interfaces and peripheral features to evolve with market needs. Bit-oriented peripherals can expand or shrink to a specific configuration. Multiple functions can be duplicated. Standard interface protocols can be enforced or relaxed. System performance can demand increased pipelining of peripherals. A choice of multiple bus architecture interfaces expands usage options. System interfaces off-chip often have fixed requirements, which the ASIC must match. Designing IP to have a large feature set that can be selectively parsed down through parameters to meet multiple user needs, broadens the applications that may be addressed without excessive gate count overheads.

As an example of parameterization power, consider how the ARM AMBA bus standard has evolved. The APB protocol has changed between revision D (ASB) and revision 2 (AHB) related to the read data sampling and register reset timing points. An APB peripheral would require a different interface depending on the AMBA protocol in use for the overall system, requiring two versions of the peripheral design to be maintained.

However, all remaining features of the peripheral remain unchanged. Using a bus protocol parameter in the IP definition, "BRIDGE_TYPE", as part of the RTL code allows the synthesis process to realize either protocol.

IP parameterization does not come without a price. Certainly parameterized IP requires longer up front development time by the provider, whether it is offered by an ASIC provider or an internal design team. A rigid data structure across all IP must be adhered to, while verification test benches and synthesis scripts tend to be more complex. With parameterization comes an

exponential increase in possible configurations, which leaves potential for verification holes in test benches for unexpected combinations. This also leads to heavier documentation to cover "what if" scenarios.

Is a strong link between standard product IP and ASIC design IP necessary? First time silicon success is critical in today's markets. By using identical IP blocks implemented in a standard product, the ASIC designer is comfortable knowing that many users have heavily exercised the block function, insuring robust logic. For IP blocks such as USB devices and multi-media card interfaces, software drivers from a standard product may be easily updated to operate with an ASIC. Even operating systems such as Linux® and Windows® can be quickly ported. A software development system from a standard product can be used to start ASIC code development while ASIC specific emulation systems are in development. If an ASIC product has evolved from a standard product implementation, the ASIC will be able to use a large portion of the software application code previously created by the product design team.

Serious developers of ARM core-based SoCs will include the creation of an ASIC specific product emulation board into the overall program development schedule. Designers may create a specific board for their application or adopt a third party board, such as the Prototype and Emulation Platform (PEP) from Atmel, created for this purpose. This emulation board would include an ARM7 or ARM9 test chip, obtained from the ASIC manufacturer; an FPGA large enough to hold all of the ASIC logic, large memories; and numerous interface and clock circuits.

From a hardware perspective, the board will be used as an aid to prove hardware design of new system components. It can be used to heavily exercise components operating to a standard protocol. And it can be used to prove complex peripheral relationships not practical though ASIC simulation. From a software perspective, it provides the software team with a vehicle to exercise real application code. This will reduce software debug efforts when final silicon arrives. The software team may have started to prove their code using a code development board from an ARM standard product. They can now move this code over to the ASIC specific emulation system.

FPGA development should not be taken lightly. Too often ASIC design teams underestimate the effort required to configure an FPGA for an ASIC emulation system. This is often a project of its own requiring specific resources. ASIC designers must consider from the start that they will be partitioning the ASIC netlist for synthesis to an FPGA. Some logic does not lend itself to FPGA implementation and must be removed from the FPGA netlist. In other cases the ASIC design may be very large, exceeding the ability of a single FPGA. Third party tools can assist designers to prepare an ASIC netlist for FPGA porting. Preparing a solid hardware emulation platform in advance of the ASIC can have results that ASIC designers may not like. ASIC designers will spend time proving that "bugs" identified by the software team are in the software code and not in the hardware. This is an inevitable event that will occur at some stage of the product development, but it may also lead to discovery of true hardware "bugs". Addressing it early in the product development timeline increases the odds of first time success silicon but must be considered in the resource planning.

ARM core ASIC product development is possible by the masses, and ARM core-based standard products offer proven paths to ASIC solutions. Re-usable/re-configurable ARM peripheral IP is a reality. ASIC system software teams can take advantage of ARM standard product software, and Atmel's SiliconCITY platform design approach enables first time success silicon with reduced development lead times.

Designer's Corner

Designing for Efficient Production with In-System Re-programmable Flash µCs

By: OJ Svendlsi

For products where time-to-market and efficient production is important, selecting the right microcontroller architecture plays a big role. To get the shortest possible development time, the following requirements should be filled:

- Good and easy to use development tools
- Enough resources on-chip to meet requirements
- Efficient for high level languages
- Flash program memory for fast and reliable programming

The megaAVR family from Atmel has all these features and more, making them a perfect choice for advanced products requiring short time-to-market.

This white paper discusses the advantages of this family related to getting the shortest time-to-market and the most efficient production once the product is ready.

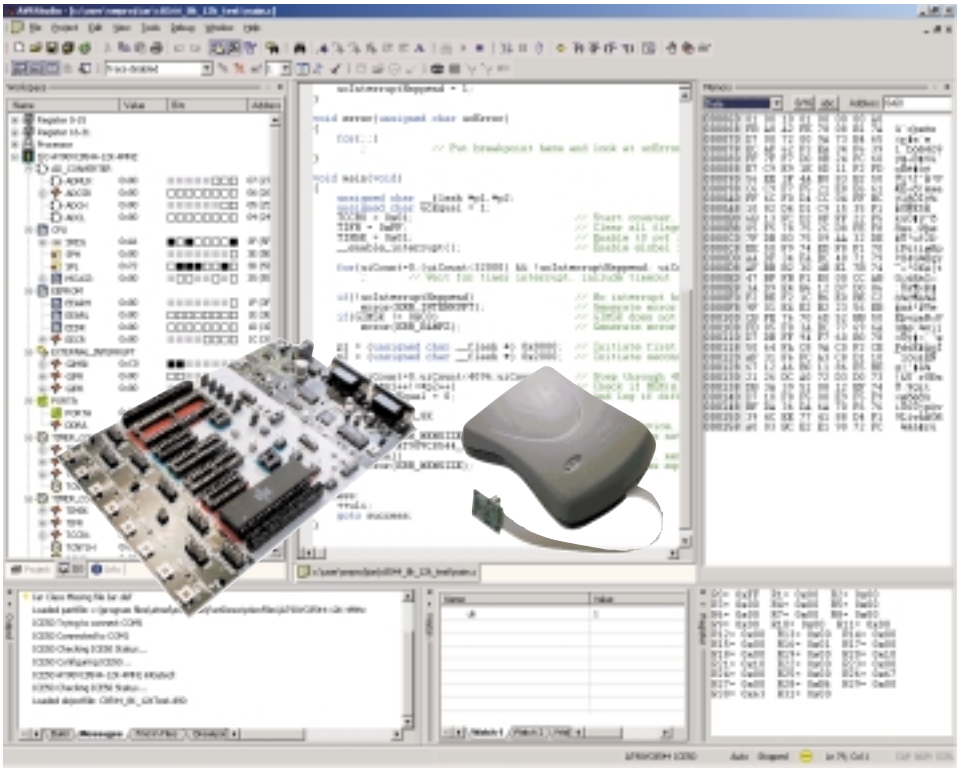
Development

During development of a new product, the microcontroller is normally the most important module, and always the component where the majority of the engineering hours are spent. Thus, making sure the microcontroller has what it takes to ease the development should be given some priority.

Selecting the Right Development Tools

When selecting a microcontroller, you are also selecting the development tools you are blessed with. These development tools range from the simplest evaluation kits to high-end In-circuit emulators and production programmers. Depending of the complexity of the end product, different debugging solutions will be optimal. A 1-2K microcontroller with a minimum of on-chip resources can often be debugged using a free architectural simulator such as AVR Studio. For complex applications on big parts with many peripherals, using an advanced In-circuit emulator with advanced features such as Trace capabilities, advanced breakpoints and Code coverage functionality might reduce the design time significantly.

Parts with on-chip debug capability offers a very low cost alternative for the debugging. The JTAGICE mk-II from Atmel is available for only $299, but still provides debugging capabilities like breakpoints, data watch, single stepping through code and full overview of processor internal resources. An additional important



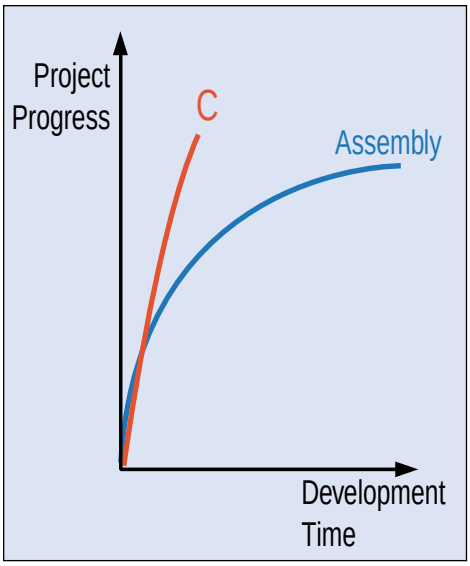
advantage of on-chip debug is that the debugging is done on actual production silicon. Because of this, many of the problems designers run into when switching from an emulator platform to the real thing is eliminated.

Starter kits are useful when evaluating a microcontroller that might fit in the application. Many starter kits can also be used as target hardware during initial code development before the target hardware is ready.

Programming in a High Level Language

By programming the microcontroller in a high-level language (HLL), like for instance 'C', it is possible to reduce the development time significantly compared to writing in assembly. Generally one can say that an experienced designer can write the same amount of lines of code per day in C and assembly. However, the code-lines written in C will do much more than the same number of lines written in assembly.

Typically, a program written in a high level language will also be much more structured than a similar program written in assembly. Because of this, it is generally easier to debug a program written in a high-level language.



Most 8-bit microcontroller architectures come with a C-compiler. However, there is a big difference in how efficient the architectures are for high-level languages, and how the C-code should be structured to be efficient with one particular microcontroller architecture. Generally one can say that accumulator-based architectures like the 8051 architecture from Intel works best with global variables, while register-based architectures like the AVR from Atmel works best with local variables.

The benefits of using local variables are that the code becomes more structured and the portability and maintenance of the code is simplified when compared to code written with very many global variables. It is also much easier to reuse code when it is written with extensive use of local variables. When all the parameters going in and out of a subroutine is defined in the function call, it is very easy to port that subroutine into a new project.

The biggest drawbacks of writing code in a HLL are that the code normally becomes bigger and slower than a similar program written in assembly. However, as the number of code-lines increases, the gap in size between the HLL code and the assembly code starts to shrink. For a typical AVR user, the crossover point where the HLL and assembly code is the same size is around 4K. However, the HLL code will almost never be faster than the assembly code. If execution speed of a certain part of the program is critical, the solution is often to write the code for the critical parts in assembly, and write the skeleton and less critical subroutines in HLL.

Integration

The level of integration in a microcontroller can also affect the development time, in addition to power consumption and board space. Integrated functions such as Brown-out protection, Watchdog timer and Power-on reset circuitry gives the most reliable operation of the microcontroller, while functions like integrated EEPROM, internal RC oscillator and strong push pull port drivers eliminates external components and reduces cost and complexity of the design. With fewer external devices on the PCB, the possibility of running into noise related problems is also reduced. Especially the internal RC simplifies the design from a noise tolerance point of view.

In-System Programmable Flash Program memory

Having In-System reprogrammable flash memory simplifies the development of a microcontroller application significantly compared to ROM/OTP solutions. The devices can be soldered into the application, and then be reprogrammed when a problem with the code is found. Using the AVR microcontrollers, the reprogramming can be done either through the SPI interface, or if using a JTAG on-chip debug or programming tool, through the integrated JTAG interface. If using a boot loader with the self-programming memory, other communication channels can also be used to reprogram the devices. The non-volatile memories of the AVR devices can be used to store history when debugging a program. The contents of the memories can then be read out after the program is finished.

Production

The advantages of using modern flash microcontrollers do not stop when the development is completed. Also when in full volume production, the benefits are many as outlined here.

Procurement

When using Flash microcontrollers, it is much easier to handle the procurement of the products. It is often easier to handle surprise orders since the microcontroller used is a standard line item. The chance that other companies are using the same microcontroller is usually big. This makes it possible to stock the microcontroller both at the distribution level and at the device manufacturer. If the same microcontroller is used for a number of applications, inventory can be moved from one product over to another to maximize the revenue.
Handling multiple versions

With a Flash microcontroller the same device can be used to handle multiple software versions. In many products, the only difference between a high-end and a low-end version of a product is the firmware. In these cases, a flash microcontroller is ideal. The PCB can be assembled and completely tested in advance. Once the order comes in for a specific product, the code for this product can be programmed in, and the product shipped to the customer.

Many larger manufacturers are selling the same product to different OEM customers. In many cases, the differences between the products shipped to these OEMs lies in the software. Again, an In-System reprogrammable flash microcontrollers can be programmed with the correct code at the production line. Alternatively, the OEM account can program the parts of the code that is individually changeable by using the boot-capability of the megaAVR devices.

Calibration

Many analog sensors have a relatively large offset error that needs to be calibrated out to achieve good measurements. Devices with integrated non-volatile data memory and high performance Analog to Digital converters are ideal for this. With the AVR microcontrollers, special calibration software can be used to run the calibration and store the calibration values in the internal EEPROM. After this, the main code can be programmed in, and use the values stored in the EEPROM.

Naval Research Lab's Data Acquisition Module for DSP

By: Flemming Christensen
Managing Director, Sundance

Sundance's new high throughout configurable data acquisition module (DAQ), the SMT391, is the flagship of Sundance's new family of high performance data acquisition systems. The SMT391 is a powerful 1 giga sample per second (GSPS), dual channel analog-to-digital converter (ADC). This high-performance 8-bit, DAQ system makes the SMT391 ideal for high bandwidth applications such as high-speed test and instrumentation, satellite communications, software defined radio (SDR), direct RF/IF processing, direct RF down conversion, and radar.

The SMT391, a "daughter module" that outputs data to a network of Sundance's reconfigurable computing and DSP systems, is managed by a Xilinx® Virtex-II Pro™ FPGA. The Virtex-II Pro FPGA, with its IBM PowerPC® embedded processor capabilities, manages the data transfers to a variety of communication channels such as ComPorts, Sundance High-speed Bus, and Xilinx RocketIO® Multi Gigabit Transceivers (MGTs). These channels are compatible to a wide range of Sundance processors and I/O modules. The FPGA controls all digital functions on the module as the digital output of the Atmel® broadband converter is fed into the FPGA. This data is then stored in an onboard DDR SDRAM for non real-time processing.

"The SMT391 leverages fully the power, performance and reconfigurability flexibility of the Xilinx® Virtex-II Pro™ and meets the stringent requirements of wireless and signal processing applications," said Dr. Nory Nakhaee CEO of Sundance. "The combination of Sundance's signal processing technologies and high-speed broadband expertise combined with the power of the Virtex-II Pro™, makes the SMT391 more than just a data acquisition module as it is also high-speed converter, a processor and a gateway to an array of powerful processing nodes," concluded Nakhaee.

A highly compact twin board design makes up the SMT391. A top layer board, the daughter board, is coupled to a base module via a Sundance LVDS Bus. This daughter board contains all the analog circuitry, the clock generation, trigger control, analog signal conditioning and as well a converter. This device handles all data acquisition and conversions. Analog data enters the top module via two analog data streams that are pre-conditioned before they enter the dual channel ADC converter. In conjunction to the two analog inputs, users can also provide the module with a custom clock and trigger. "Sundance was one of the first to recognize the benefits our Virtex-II Pro can offer to high speed broadband applications," said Jerry Banks, Director of Worldwide DSP Solutions Marketing at Xilinx. "The Sundance data acquisition system introduces a novel level of design, performance and flexibility that delivers the most advanced features engineers expect in a system oriented data acquisition module."

Sundance SMT39 module

The base module is a reconfigurable computing system also powered by a Xilinx Virtex-II Pro FPGA. The FPGA is center to the base module as it controls all digital functions on the module. By separating the analog circuitry from the digital one, Sundance has not only substantially reduced cross-talk, but has also provided an astute heat-dissipation scheme for the bottom board components. Configuring the main FPGA is done via an on-board and In-System programmable logic that configured through an on-board JTAG interface. Configuration, sampling and transfer modes are set by configuration data received over the ComPorts or the RocketIO Serial Links. The SMT391 is truly a reconfigurable data acquisition system; the Atmel ADC converter digital outputs are fed into the FPGA where they can be processed by an optional user-designed intellectual property core. The data is transferred to a 64 Mbytes (per channel) DDR SDRAM memory. When the SMT391is embedded in a larger system environment, this data is disseminated to other system modules via a high-speed bus interface. Alternately, data stream may also be transmitted over to the FPGA Serial Link interface. "One of the most challenging design issues we faced in developing our standalone systems was the implementation of a cost effective high-speed ADC," said Lan Tran of the Naval Research Laboratory, Information Technology Division. "With Sundance's SMT391 high sampling rate, throughput and compatible DSP and FPGA systems we managed to meet our specification requirements and remained within our project budget constraints."

Designed for high-speed I and Q channel type applications, the SMT391 has two identical channels and all settings are applicable to both. Operations for both channels are the same, and there is only one sample clock for both channels and both channels will respond to the same trigger. The dual channel Atmel® converter converts the analog data, and a single 16-bit parallel data-stream is generated for each channel. One stream is transmitted as is over the RSL interface for real-time type applications. The second data-stream can either be transmitted straight over the SHB interface or stored in DDR SDRAM every time a trigger is received.

Simplified Ethernet Stack Core or, How to Reduce System Cost

By: Guy Lafayette, Atmel

COST IS VERY OFTEN AN IMPORTANT ASPECT OF EMBEDDED SYSTEM IMPLEMENTATION CHOICES. SOFTWARE AND HARDWARE PARTITIONING IS KEY IN THAT PROCESS DECISION. FUNCTIONS IMPLEMENTED IN SOFTWARE ARE GENERALLY LOWER COST THAN THE ONE IMPLEMENTED IN HARDWARE. ON THE OTHER HAND, HARDWARE GIVES SOME PERFORMANCE SOFTWARE COULD NOT ACHIEVE.

ANOTHER WAY TO LOOK FOR SYSTEM COST IS ON THE IP USED. IP VENDORS TENDS TO SUPPLY FULL FEATURED IP, THEY ARE PRIMARILY DESIGNED FOR HARDWIRED ASIC AND ARE FULLY CONFIGURABLE BY SOFTWARE TO COVER THE MAXIMUM OF APPLICATIONS. VERY OFTEN THE FULL FEATURE IP IS OVERKILLED FOR THE APPLICATIONS. REMOVING UNNECESSARY FEATURES WHILE KEEPING THE COMPATIBILITY WITH STANDARD ARE SOMETIMES VERY EFFICIENT FOR COST REDUCTION.

PARTIAL AND DYNAMIC RECONFIGURATION CAPABILITY OF SOME FPGA ALLOWS YOU TO FURTHER REDUCE THE COST IN SOME CASES.

IN THE FOLLOWING ARTICLE, WE WILL TAKE THE EXAMPLE OF THE ETHERNET STACK AND DESCRIBE HOW YOU CAN IMPLEMENT A SIMPLIFIED ETHERNET STACK (SES) USING THE ATMEL FPSLIC.

FPSLIC Architecture

Atmel's AT94K and AT94S family of Field Programmable System Level Integrated Circuits (FPSLIC devices) combine all the basic building blocks of a system (logic, memory and uC) in an SRAM-based monolithic field programmable device.

The FPSLIC Secure (AT94S) family of devices offers and even higher level of integration. In addition to the FPGA, AVR and SRAM memory, these devices include on-chip serial configuration memory. The combination of the FPSLIC device and the embedded serial configuration memory in a single package allows for ISP and remote updates of FPSLIC while adding external read protection, thereby protecting the designer's FPGA and AVR programming code from theft.

FPSLIC devices combine 5K to 50K gates of Atmel's patented AT40K FPGA architecture, a 25 MIPS AVR 8-bit RISC microprocessor core, numerous fixed microcontroller peripherals and up to 36 Kbytes of program and data SRAM.

SES Features

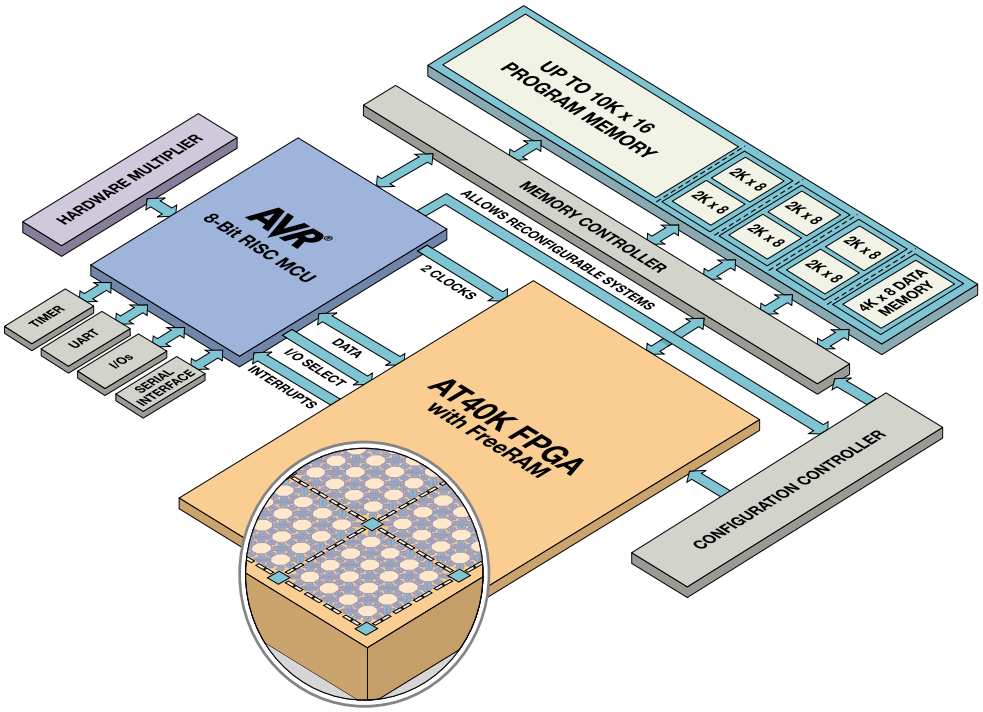
- Provides 10 Mbps Ethernet connectivity to FPSLIC via standard MII bus.
- Requires only external Ethernet transceiver.
- Provides stripped-down hardware implementation of MAC and IP Layers.
- Well suited for control applications.
- Shares overall functionality among hardware (FPGA) and software (AVR).

Outline

The SES (Simplified Ethernet Stack) core is aimed to provide a limited Ethernet communication stack to the ATMEL FPSLIC System-on-Chip. Although an embedded system often needs to support Ethernet functionality, rarely are all the features provided by the TCP-UDP/IP/MAC Layer protocols really necessary. The SES core fills that gap and provides FPSLIC with a minimal implementation of the TCP over Ethernet suite of protocols, while retaining compatibility to existing standards.

Description

The SES core includes part of the MAC layer functionality, implemented in the FPGA section of the FPSLIC. Therefore only a PHY transceiver is necessary for interfacing to the Ethernet channel through the standard MII bus. The IP protocol implementation is distributed among hardware running on the FPSLIC's FPGA core and software executed on the AVR core. Additionally, the UDP protocol is implemented in software. This approach offers a good compromise between software and hardware in terms of resources, utilization and execution speed.



To transmit a packet, the AVR places the data to be transmitted into the Shared SRAM block and issues a transmit command. The Tx State Machine controls data flow from the Shared SRAM and handles control signals to/from the Ethernet PHY. The TxFCS Calculation module generates the FCS and appends it to the outgoing frame. An interrupt notifies the AVR when the packet has been sent, or if it has been rejected due to excessive number of collisions.

For packet reception, the AVR enables the receiver, and when a valid frame has been received an interrupt is issued. To identify an incoming frame as valid, the receiver module checks each frame being received against a set of predefined rules; if the frame qualifies for those rules it is stored in the Shared SRAM, else it is rejected. The three receiver modules (Rx Validation Machine, Rx FCS Validation, and Rx IP Checksum Validation) perform frame validation, while the Rx Validation Machine handles the control signals to/from the PHY, AVR and SRAM.

Since the SES core is mainly targeted to control applications running on top of the UPD protocol, various optimizations have been introduced. They include:

• Acceptance of IP frames with specific header attributes: This allows for the implementation of a state machine that qualifies all incoming messages and discards those that have fields with inappropriate values.

• Elimination of incoming frame buffers: The receiver accepts incoming frames as soon as the Application Layer is ready. This limits the amount of on-chip memory that is allocated for frame buffering, simplifies the Application Layer functionality and allows more resources to be available for the end application.

• Utilization of an IP address-based addressing scheme rather than the default MAC address-based one. Since the SES core is naturally intended for handling small messages (usually of length much smaller than the maximum allowed), frame broadcasting can be used in the MAC layer and addressing can be performed in the IP layer. This is not a violation of the Ethernet protocol standards: a network node that implements the full TCP/IP stack will discard such an 'invalid' broadcast frame since it will not be of any of the standard protocols that use Ethernet broadcasting (for example, ARP). However, an SES-based node will accept such a frame. This technique allows the elimination of the ARP protocol and overcomes the need for MAC addresses allocation.

The optimizations described above, lead to a compact and efficient implementation of an Ethernet-based stack, yet compatible with full implementations. Therefore an SES-based device can coexist with devices that implement the whole TCP/IP suite of protocols. As a consequence, software can be developed to allow a Host access the SES-based devices and perform network maintenance functions (such as IP address assignment, status query) as well as remote administration (such as status update).

Design Implementation and Verification

The FPSLIC design software (System Designer™) is based on industry standard FPGA and microcontroller design tools. The System Designer environment from Atmel includes: VHDL and Verilog synthesis, VHDL simulation, FPGA place and Route, AVR studio tools. FPSLIC supports most of the standard AVR C/C++ compilers. In addition, Atmel has combined state of the art co-verification tools to allow concurrent hardware and software development and debugging, greatly reducing overall time-to-market.

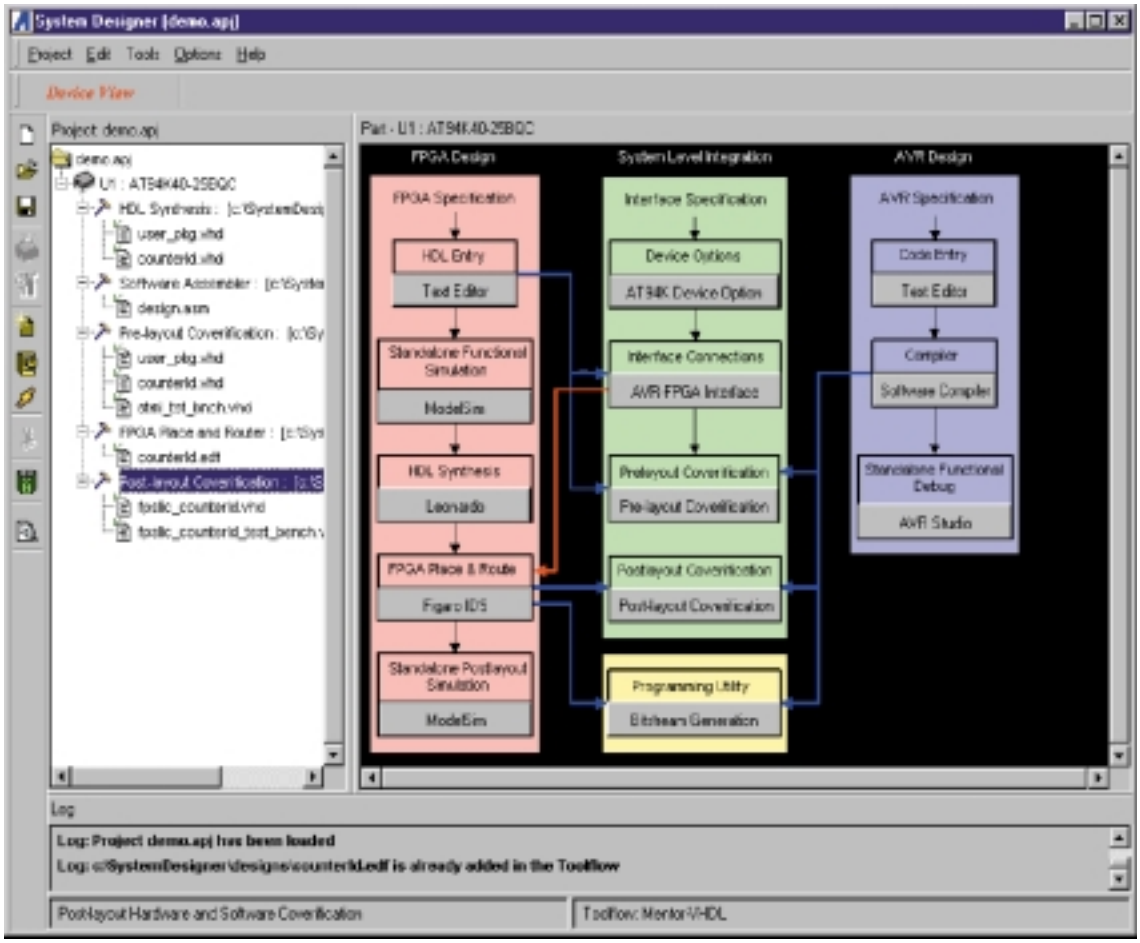
The tool set is completely integrated and includes everything necessary to implement designs -- including complete FPGA and AVR microcontroller tools, along with Exemplar Synthesis, Model Tech Simulator and Mentor's Seamless Co-verification tools.

Applications

The SES core is mainly targeted (but not limited) to control applications over the Ethernet, such as remote sensor reading, control of digital and analog I/O, etc.

FPSLIC Resources Requirements

The core fits in AT94K10 and AT94K40 devices. Implementation on AT94K05 devices using the Partial Reconfiguration capabilities of the FPSLIC family could be considered, too.



FPSLIC design software (System Designer™)

A Low-end Approach to Tamper-proof Energy Metering with the AVR465

By: Kim Meyer,
Atmel Field Applications Engineer

TRADITIONALLY, ELECTRICAL ENERGY HAS BEEN MEASURED USING ELECTROMECHANICAL METERS, ALSO KNOWN AS FERRARIS METERS OR, MORE DESCRIPTIVELY, TURNING-WHEEL METERS. GRADUALLY, AND AT AN INCREASING RATE, MECHANICAL METERS ARE NOW BEING PHASED OUT BY ELECTRONIC. CONTEMPORARY DESIGNS INCLUDE THE USE OF HIGH-RESOLUTION A/D CONVERTERS, 16- OR 32-BIT DSPS, µCS.

ATMEL NOW CHALLENGES THIS SETTING BY INTRODUCING A CONCEPT WHERE A SINGLE LOW-END, COST-EFFECTIVE, 8-BIT AVR MICROCONTROLLER HANDLES ALL MEASUREMENT TASKS REQUIRED IN A SINGLE-PHASE METER. THE DESIGN PRESENTED IS A TAMPER-PROOF METER, AIMED TO PARTICIPATE IN THE MOVEMENT AGAINST ENERGY THEFT BUT IT IS EASILY MODIFIED TO FIT OTHER ENERGY METERING DEMANDS, AS WELL.

Introduction

Energy or power metering is about collecting data on energy consumption for billing purposes; the electricity utility provides customers with energy and customers pay for the amount of energy consumed. The energy meter measures energy and stores billing data for the utility provider to collect, either manually or electronically.

Traditionally, energy meters have been mechanical but are rapidly being replaced by electrical due to many reasons. Foremost, the price of electronic meters has come closer to that of mechanical but other factors are also playing increasingly more important roles. Electronic meters are fast and easy to calibrate, calibration can be automated, measurement results don't degrade over time and it is easy to add more sophisticated functions like remote reading to an already electronic design. In selected areas the most important issue, however, is that electronic meters can resist tamper attempts and continue to measure reliably in situations where the mechanical meter could not.

It has been estimated that some utility companies are loosing more than 40% of their revenue due to defective or illegal connections, defective or dead meters and meter tampering. The major cause for revenue loss is tampering, i.e. illicit interference with the meter and wiring in an attempt to withdraw more energy than the customer is paying for. Mechanical meters are highly subjected by tampering but most pilfering schemes can be detected, signalled and even overcome by electronic meters. Replacing mechanical meters with electronic gives many benefits, including an immediate improvement in revenue yield.

Application note AVR465 describes how to use the AVR microcontroller to build a tamper-resistant meter. Besides meeting all the demands on tamper prevention it gives all the benefits one can ask for in a modern energy meter. Also, the design is easily customized to fit special needs, such as remote reading and demand recording. It is easy to replace the default ATmega88 with the pin-compatible ATmega168 and gain twice the Flash memory for program enhancements whereas migrating to, say an ATmega169 gives the benefit of an integrated LCD driver.

Tamper-proof metering is a special case of single-phase metering, the difference being that tamper-proof metering requires current to be measured in live and neutral wires, whereas traditional single-phase metering relies on only one current transducer. Application note AVR465 is easily modified (reduced, actually) to fit traditional single-phase metering by simply removing the second current transducer, related measurement algorithms and tamper logic. The reduced code roughly fits a 4kB AVR microcontroller such as the ATmega48, cutting the overall cost of the meter. Current transformers are typically expensive and easily mount up to a quarter or even half of overall component cost. As a result, the bill of materials of a single-phase meter can be nearly half of that of the presented tamper-proof design.

Key Features and Benefits

Albeit many mechanical meters are yet to be replaced, there are many solutions for electronic metering. The three most used solutions are based on ASICs, ASSPs and microcontrollers. Very few companies use ASICs anymore, leaving the struggle between ASSPs and microcontrollers.

Cost

Meter manufacturers naturally want to drive the overall cost of the meter as low as possible. As a discrete component, the ATmega88 used in this application is less expensive than most ASSP devices and rival microcontrollers. Considering the integrated peripherals of the ATmega88, the price benefit increases, as there is no need to include external devices such as an EEPROM for calibration data storage.

In addition, many ASSP designs include a microcontroller.

Flash Memory

Because of the competition, many manufacturers want to offer a meter with unique features or personalized settings. Also, since many meters are based on the same technology and the use of similar components, manufacturers want to offer something personalized, a meter with unique functions. In addition, it should be easy to make modifications to the design, when required.

The integrated Flash memory of the AVR makes all the above possible at no extra cost. The firmware is easily customized to fit any needs of the manufacturer and it can be updated at any time, even in the field. This is a huge benefit over ASSP designs, which are fully static and therefore make the design easy to copy by rival meter manufacturers. The AVR enables the firmware to be securely locked inside the microcontroller, making it impossible to copy the design but yet allowing the firmware to be updated at any time using a bootloader. Nonvolatile memory can also be used in more advanced meters for demand recording and logging measurement data.

Integrated Peripherals

The AVR microcontroller includes many peripheral

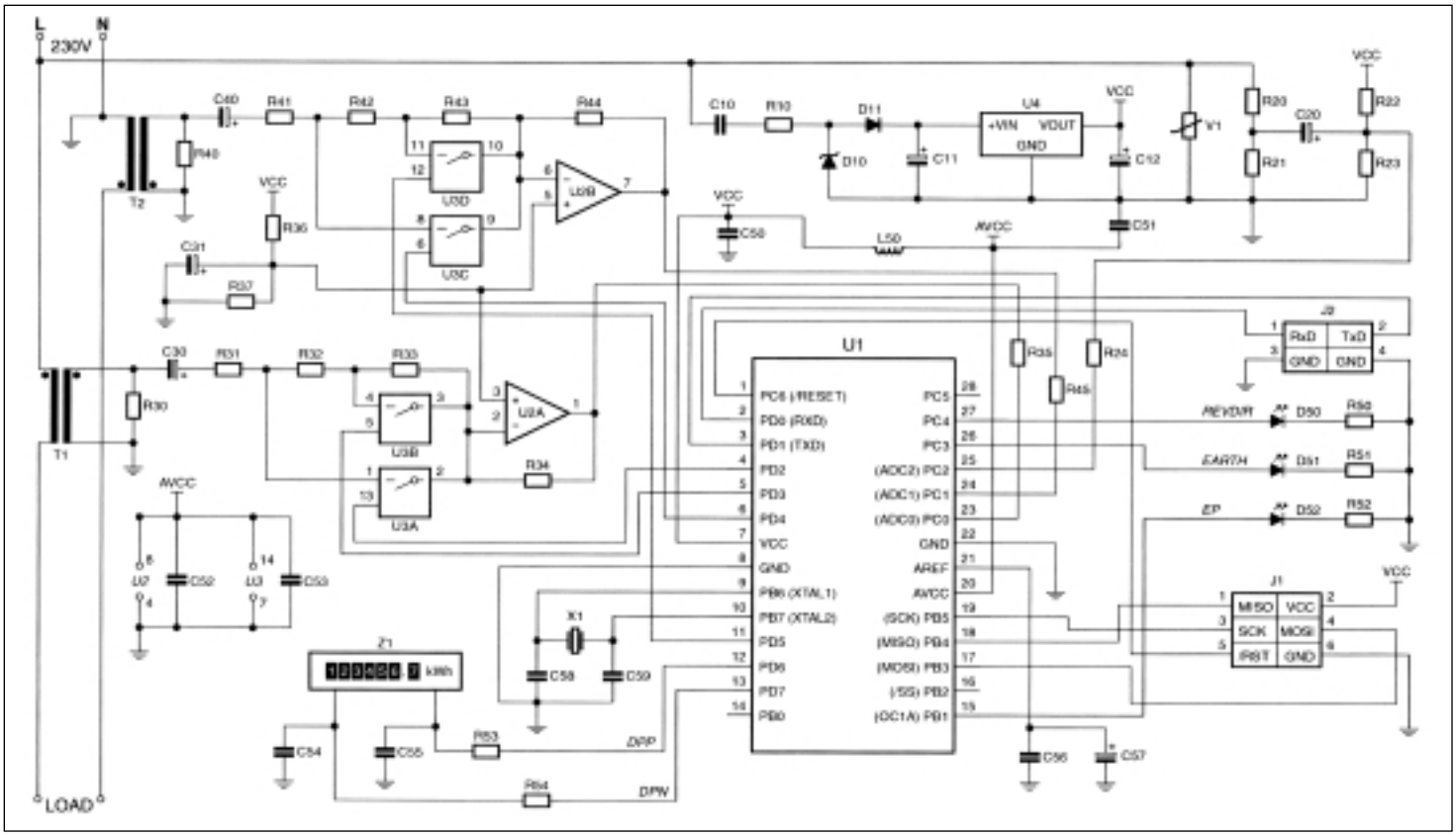


Figure.1: AVR465 Schematic

functions not available in any ASSP and not many rival microcontrollers. The USART interface enables easy remote reading and calibration via the PC serial port, but also comes in handy if the meter is to be equipped with an optical interface. ASSP designs typically do not include serial interfaces at all. Also inherent to the ATmega88, SPI and TWI enable fast and efficient communication with external components, for example. In addition, the fully programmable I/O ports of AVR microcontrollers allow easy control of other external components, such as LEDs.

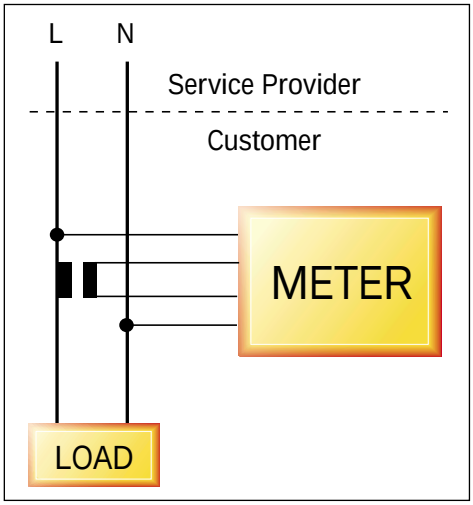


Figure 2: Single-phase Metering

Ease of Migration

The meter design is easily customized, not only because of the integrated Flash memory but also because the firmware readily migrates to other AVR microcontrollers. For example, in a matter of seconds the firmware is migrated to ATmega168, effectively doubling the Flash size without changing the pin configuration. Similarly, the design can be stripped down to fit the 4kB Flash of an ATmega48, effectively reducing the component cost. Additional features, such as an LCD driver are readily added by migrating to ATmega169, for example.

Target Applications

Application note AVR465 describes a tamper-proof single-phase meter, but it also serves well as a starting point for other types of energy meter designs.

Tamper-proof Single-phase Meters

Tamper-proof meters rely on two current transducers, where traditional single-phase meters rely on one. The idea is to measure current flowing to the load (through the live wire) and current returning from the same (through the neutral wire). The meter then assesses the validity of the measurements and uses the assumption to overcome attempts of tampering.

Traditional Single-phase Meters

Tamper-proof meters are required in selected regions where revenue loss due to pilfering is significant. Traditional single-phase meters do not need two current transducers and, more to the point, regional demands may even forbid meter connections to cut the neutral wire due to safety regulations.

The design presented in application note AVR465 is readily reduced to a traditional single-phase meter. Measurement circuitry and measurement algorithms related to the neutral wire are easily removed, effectively reducing the bill of materials at the same time. Reductions in board space, component cost and firmware size are typically between 25% and 50%.

Advanced Single-phase Meters

An advanced meter typically does something else than just measures and records energy consumption. Additional features such as remote reading, demand recording and displaying measurement results on an LCD are typically considered. Remote reading is readily implemented using Atmel's SmartRF™ solutions, for example. Demand recording and profiling requires little more than a modification to the firmware and some spare Flash or EEPROM, both of which are already included on-chip. LCD support is also an option in some AVR microcontrollers.

Single-phase, Three-wire Meters

Single-phase energy service in the US is distributed using three wires; two for live voltage and one as grounded neutral. Similarly as in the AVR465 design, two transducers are required for measuring current in both live wires. Minor changes to the hardware and a

few changes to the firmware effectively transform the AVR465 design into a single-phase, three-wire meter.

Polyphase Meters

Most polyphase energy service is three-phase. The design presented in AVR465 serves as a starting point in a polyphase meter design where each phase is monitored by a separate microcontroller.

Development Tools

Firmware development, downloading the code to the microcontroller, reading measurement results and calibrating the meter can all be done using Atmel's AVR Starter Kit, STK500. For program development and debugging purposes and for easier access to on-chip EEPROM during calibration Atmel recommends using AVR development tool JTAGICE mkII.

Related Products

As part of an attempt to minimize the theft of energy, many utility companies consider using more than just a stand-alone meter for billing and monitoring. Although an energy meter built around the principles illustrated in AVR465 does function properly under listed tamper conditions people involved in energy theft sometimes use brutal methods, including partial or full destruction of the meter. Some meter manufacturers may therefore consider enhancements to the basic design, such as remote reading or prepayment systems.

In addition, the meter manufacturer may want to provide a wide product range, with meter options altered either during production (as population alternatives), or after (as plug-ins). Market volumes for ehanced meters may not be as large as those for plain, but the manufacturer probably wants the option to exist in the basic meter design. Hence, additional circuitry needs to be designed already at an early stage.

Remote Metering

Remote metering is typically not a default option, but something provided for selected customers. The preferred customer base may include suspicious clients or those located very close to others, such as in a high-rise building. In the latter case, tens or hundreds of meters may use RF to send billing data to a common collector unit, which then transmits the data package to the service provider using telephone modem, cable TV, internet connection, or similar.

Atmel has a broad range of communication products that fit remote reading demands, such as Wireless LAN, BlueTooth, SmartRF, and similar devices.

Prepayment Metering

Prepayment metering means the meter is equipped with an interface to a token or a chip, which contains validation data that either allows or prohibits energy withdrawal. If no validation is found the meter will cut power line connections, for example when all money tokens have been used. Prepayment meters are often used in temporary connections.

Atmel has a selection of SmartCard products that fit prepayment demands.

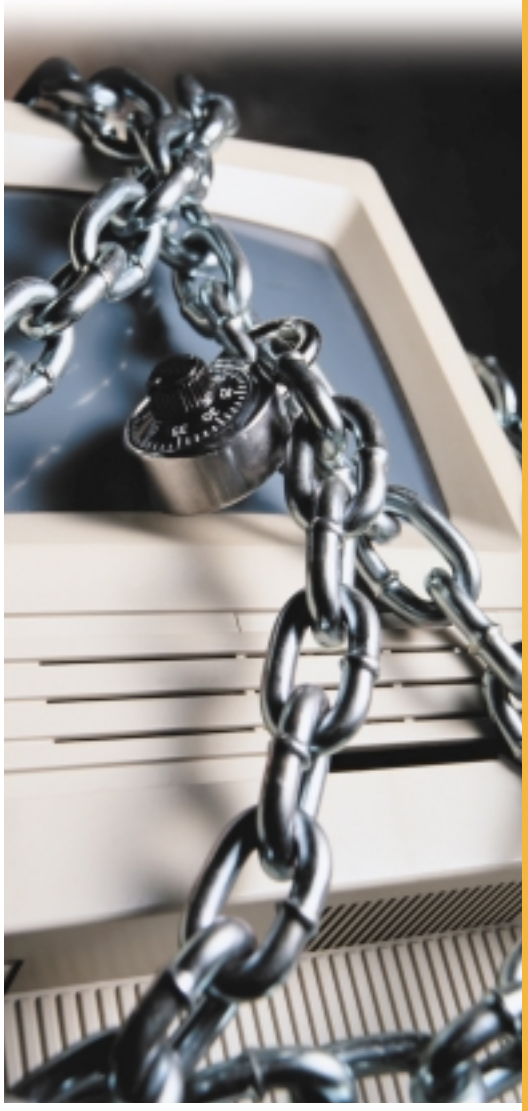
The ATmega88 Microcontroller

ATmega88 is a fresh addition to the well-established 8-bit AVR Flash micontroller family. The mega88 (and pin-compatible ATmega48 and ATmega168) AVR Flash microcontrollers expand the AVR's reach into the low pin count, medium memory size market. Designed with low-power consumption in mind, this sub-family introduces many of the high-end AVR features into the low price point market. Features such as Self-Programmable Flash, Data EEPROM, and On-Chip Debug (OCD) are rarely seen on low cost devices in this range, but Atmel's high-density non-volatile memory process technology allows the integration of such advanced features at minimum cost.

Further information can be obtained from Atmel's Web site at www.atmel.com. Contact: Kim Meyer, AVR Applications Group, Finland. Tel: +358 (0) 9 4520 820, email: kmeyer@atmel.com.

Limiting Illegal Hardware Copies by Using Secure Hardware Authentication

By: Dany Nativel, Atmel



Intellectual Property (IP) theft is a problem confronting many system manufacturers. Market share has been eroded due to other competitors providing a similar platform at a much lower price. In some cases, the entire system including hardware and software is being duplicated. IP theft has allowed unscrupulous manufacturers to avoid the costly development and validation phases and to offer a lower price for a comparable system.

Protecting Hardware IP

Protection has always been a tradeoff between increased costs versus additional security features. An ASIC system architecture can provide very good protection against illegal copies. However, if no additional protection exists at the silicon level, reverse engineering, although costly, is still feasible. An ASIC is inflexible and requires a significant investment in time and money for development and validation. In many cases, it is easier to use an existing standard product such as a microprocessor and define the security perimeter at the system level rather than the chip level. Re-programmable solutions can also enhance the overall security but are limited to the FPGA's security profile.

Filtering By Online Authentication

A networked system has the advantage of having a piece of hardware that is in theory not physically accessible to the end user. Even if this doesn't prevent an attack against the main server, it can help a service provider to guarantee that only genuine systems are connected to the network.

An easy and cost effective way to limit the proliferation of unbranded systems is to securely authenticate the system board. It is much easier to secure a system operation (e.g. authentication) rather than trying to secure a whole complex process. The Media Access Control (MAC) address is often used on cable modems to determine customer identity before attributing an IP address. Unfortunately, a MAC address can easily be forged or duplicated. Currently, this is not a major issue of concern because all cable providers give subscribers the option to use their own modems. This situation could possibly change in the future if new innovative features are added to cable modems which could cause the Internet Service Provides (ISP) to reconsider their third-party modem policy. They might want only specific modems to be connected to their network. A secure access module that combines unique serial number and strong online authentication allows identification and authorization of the piece of hardware trying to connect to the network. Such a module could easily be added to an existing system using various interfaces such as the smartcard interface ISO 7816, Serial Peripheral Interface (SPI), Low-Pin-Count bus (LPC), Universal Serial Bus (USB) or any proprietary interface based on some General Purpose I/Os (GPIO).

The minor board modifications required to accommodate the small secure IC significantly increases IP protection and system security.

Insecure Authentication Using Conventional ICs

In a typical network, the processor connects to the network by using some kind of authentication. Unfortunately, the application software and other data including secret keys are often stored on some external Flash/SRAM/EEPROM. Some modern microcontrollers offer on-chip Non-Volatile Memory (NVM) but they do not offer efficient protection against reverse engineering, leaving most of the system from PCB layout to Flash content susceptible to duplication.

Instead of having the main processor manage the secret key and compute the authentication challenge, this critical task can be delegated to an external secure IC that:

- Protects the key/serial number and detects any tampering
- Securely performs cryptographic operations

Key Protection and Tamper Detection

Using an innovative custom design for both logic and NVM, Atmel secure products can provide both key protection and tamper protection along with several additional security features to provide secure code execution.

A metal shield protects the on-chip memory from being probed by an attacker. This shield will also detect any tampering attempt and will trigger associated security alarms. Automatic self-erasure of both program and data space can be setup for maximum security.

Additional features provided by Atmel Secure products include power, frequency and temperature protection logic, logical scrambling on program data and addresses, power analysis countermeasures, and memory accesses controlled by a supervisor mode.

Secure Cryptographic Operations

Atmel secure microcontrollers contain a crypto engine that offers significant security benefits compared to conventional software or firmware implementations:
It will perform cryptographic operations much faster than software implementations based on a similar architecture.

The crypto engine uses advanced techniques to avoid predictable behavior such as timing or power con-

sumption that could help an attacker to determine the value of a shared or private key.

The SC16 crypto engine featured in the AT90SC series is a 16-bit processor dedicated to performing fast encryption and authentication functions on large keys. It comes with Pre-programmed functions for Cryptography and Authentication including RSA, DSA, Key Generation, and ECC.

Authentication strategies

When the system connects to the network it will delegate the authentication process to the companion chip so the main processor will act as a bridge between the network and the Secure chip.

There are many different authentication schemes available. The selection will be driven by the complexity of the network and number of involved clients. The schemes can be split into two groups :

- Public Key Authentication using asymmetrical encryption (e.g. RSA)
- Pre-Shared Key Authentication using symmetrical encryption (e.g. DES)

Public Key Authentication

A typical secure authentication involves a challenge-response exchange between the central server and the remote system.

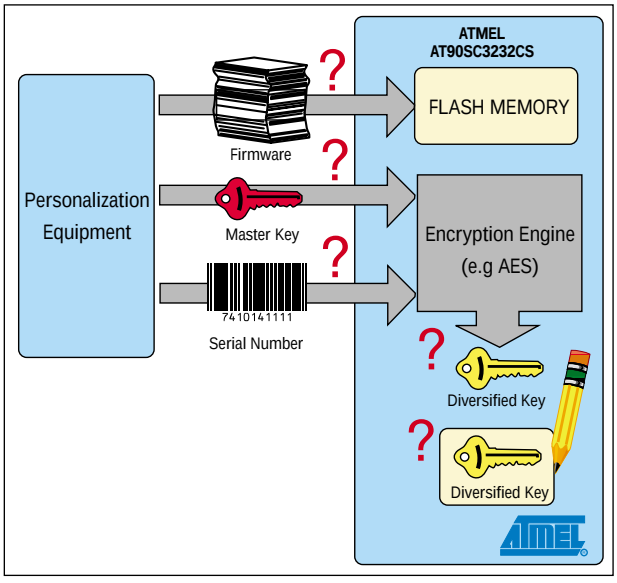


Figure 1: Chip Personalization

When the system wants to connect to the network, it first sends its serial number to the main server. The server then generates a random number and encrypts it with the public key associated with the serial number it has just received. This “encrypted” random number can then be sent over the network to the system board. Only the Secure chip containing the corresponding Private key can decrypt this “encrypted” random number. After decryption, the random number is sent back to the server (in clear text this time). Upon reception of this data, the server compares it to the original random number. A match would indicate correct authentication.

The authentication scheme described above is very powerful and offers strong security on both sides (system board and server). The main drawback of this scheme is the complexity of the key management limiting overall scalability.

After each system board personalization, a unique S/N and public key have to be sent to the central server. When deployed to millions of units, the cost and complexity of updating and maintaining such databases can be prohibitive.

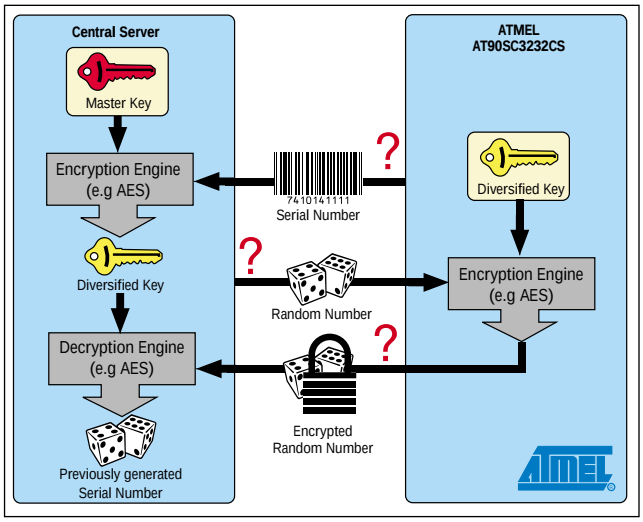


Figure 2: Pre-shared Key Authentication

Pre-shared Key Authentication

Another way to perform secure online authentication without the difficulty of public key management is to use diversified keys based on a more conventional symmetrical encryption. This authentication scheme offers both scalability and ease of implementation. In a symmetrical key environment the same shared key is used to encrypt and decrypt a message. If the shared key becomes compromised, the system becomes exposed.

In order to enable the secure authentication, the server and Secure IC must be personalized. That includes the generation of a Master Key for the server and unique Diversified key for each Secure IC. The Master Key resides on the server and can be generated locally or using dedicated equipment.

During the personalization phase the Secure Chip will be programmed with the following items:

- Application software (fig 1, u)
- Copy of the Master Key (fig 1, v)
- Unique Serial Number (fig 1, w)

The Atmel Secure Chip also provides a unique hardware serial number that can be used in conjunction with the one sent during the personalization.

From a security point of view this phase is critical because the Master Key is exposed for a short amount of time. Maximum security should be observed during personalization.

At the final stage of personalization, the on-chip crypto accelerator encrypts the Serial Number using the Master Key and algorithms such as DES/3DES/AES (fig 1, x). The Master Key is then erased from the Secure Chip and the previously computed key also called Diversified Key is stored into the on-chip EEPROM (fig 1, y).

Grouping Serial Numbers and protect them with different Master Keys reduces the risk of compromising the whole system if a Master Key leaks.

The system is now ready to perform a secure authentication over the network:

- The Secure IC starts by sending its serial number to the server (fig 2, u).
- The server responds to the Secure IC with a random number (fig 2, v).
- Using its Diversified Key, the Secure IC encrypts the given random number and sends it back to the server (fig 2, w).

For a same given random number, the response sent by each system board will be different due to the different key used on each system.

Using the system board’s Serial Number and its own Master Key, the server can compute the decryption key (aka Diversified Key) and decrypt the message sent by the Secure IC (fig 2, x). If the decrypted message matches the original random number the authentication is successful.

Adding a Secure Chip to a system board can eliminate illegal system board duplication. A system based on traditional ICs can be easily upgraded by adding a Secure Chip. The manufacturing can done anywhere without compromising the security of the system. Additionally, on-chip Secure Flash and EEPROM provide unmatched flexibility as well as permitting secure firmware updates and allowing a quicker time to market. This cost effective solution ensures that only genuine boards can connect to the network.

Publisher: Glenn ImObersteg
Glenn@convergencepromotions.com

Managing Editor: Bob Henderson
Bob@convergencepromotions.com

Technical Editor: Markus Levy
Markus@convergencepromotions.com

Production Manager: Dave Ramos
dbyd@garlic.com

Designer: Judy Gosse
Judy@convergencepromothions.com

Is Security Needed in Your Memory Application?

By: Mary T. Jarboe, Atmel Marketing Manager, Crypto and RF Memories

Atmel now offers CryptoMemory, the industry's only Secure Serial EEPROM with the same pinout as the AT24xxx series, and also utilizing the same popular 2-wire serial interface. This allows an embedded application to be easily upgraded from an open EEPROM to a secure solution with its dual communications protocol. The feature comparison matrix between plain memories and CryptoMemories shown here, exhibit the similarities as well as the differences between serials and CryptoMemory.

CryptoMemory provides a secure location for a manufacturer to store information not accessible by the user and can protect specific user data from other users. It also provides a unique and secure ID for removable components to control use in a system and prevent counterfeiting. CryptoMemory offers four different levels of security allowing each zone of memory to be independently configured to different levels of security. The four options are no security, password protection, authentication, and data encryption and MACs.

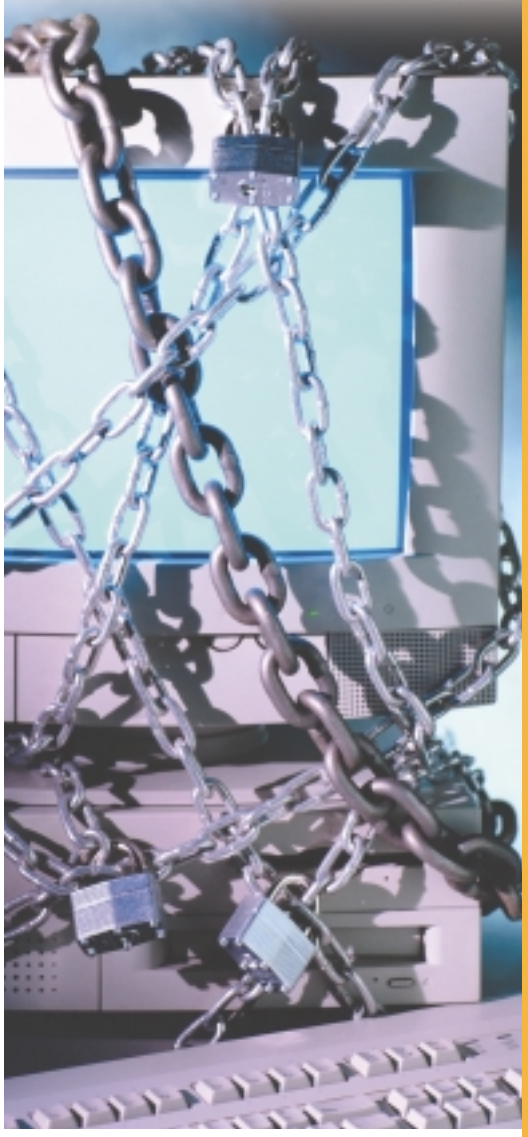
The first option is no security, where one or more zones are set up with open access for data that does not require protection. In this mode it operates much like a serial.

The second option is password protection, which requires a password to be successfully presented before access is granted to the protected memory zone. Only four attempts are allowed before that password and the protected zone are locked permanently.

The third security option is authentication. This option protects one or more memory zones by exchanging 64-bit cryptograms. Independent calculations are performed for a dual authentication between CryptoMemory and the host logic. Incorrect attempts are limited, and access is granted to the protected memory zone only after successful calculations.

The fourth and highest level of security offered in CryptoMemory is data encryption and message authentication codes (MACs). To use encryption, authentication much be successfully performed first. With each successful authentication, a new 64-bit session key is generated. This key is used for subsequent encrypted read and write operations. When a memory zone is configured to require encryption, all data exchanges will be encrypted. A MAC may be requested to further validate data read from the device, and a MAC is required for each data string to be written to CryptoMemory.

Tamper protection circuits have also been added to stop hackers. These circuits will detect operations outside the defined limits and will safely shut down the device, preventing all access to its contents. Any or all of these security levels can be used to enhance the security of a currently open serial memory application. More information is available about CryptoMemory at www.atmel.com/products/SecureMem



Features	Serial EEPROM	CryptoMemory
Catalog Part	AT24Cxx	AT88SCxxvvZ
Protocol	2-way synchronous	2-way synchronous
Package Types–8-lead	TSSOP, SOIC, PDIP 3x5 MLF (MAP)	SOIC, PDIP, 4.9x6 MLF (SAP)
Package Pinouts	Same	Same
Industrial Temperature	Yes	Yes
Operating Voltage	1.8V-5.5V	2.7V-5.5V
Clock Speed (VDD= 2.7V-5.5V)	400 KHz	1.5 MHz
Memory	1Kbit-1Mbit	1Kbit-256Kbits
Number of Memory Zones	1	4, 8 or 16
Separately Programmable Zone Security	N/A	Yes
Memory Density Indentification/Recognition	No	Yes
Idenification Area	None	296 bits
Configurable Security	No	Yes
• Encrypted PIN Security	No	Yes
• Symmetric Dynamic Mutual Authentication	No	Yes
• Attempts Counters	No	Yes
• Stream Encryption	No	Yes
• Read/Write Encrypted Checksum	No	Yes
• Write Lock Mode per Byte	No	Yes
• Power Loss Protection	No	Yes

xx = Kbits yy = number of zones C = encryption

Feature Comparison Between Plain Memories and CryptoMemories

DATA SAFE

Datakey Electronics' Products

Serial Memory Tokens

Serial Memory Keys

Serial Memory Extended Tokens

DATA LOST

Retail Products

Memory Stick®

SD

MMC

USB FOB

Smart Media

CompactFlash™

Smart Card

RFID

DATA SAFE or DATA LOST?

Choose Datakey Electronics' rugged and reliable keys & tokens for critical data

We programmed our keys and tokens, and ran over them at a gravel-pit with a semi-truck 8 times. Afterwards, they **all** still worked! We also ran over retail memory products with the opposite result.

Unlike consumer cards and other carriers that can easily fail in harsh conditions, Datakey Electronics' key and token bodies protect the embedded EEPROM. This is one of the many reasons Datakey Electronics' products are chosen daily in hundreds of industrial, military, medical, and commercial applications. They've been proven to continue working even after they've been:

- Sterilized
- Dropped in the mud
- Washed
- Zapped by static
- Exposed to chemicals
- *Run over by a semi-truck!*

Also, unlike retail memory products, Datakey Electronics offers long-term OEM support and product availability. Our Keys and Tokens have been proven rugged and reliable for over 28 years. Let our products carry your critical data!

Datakey® ELECTRONICS

1-800-328-

Implementing DES/3DES with Atmel FPSLIC

By: Axel Sikora,
University of Cooperative Education Loerrach, Germany

THE NEED FOR SECURE DATA NOT ONLY APPLIES TO WIRED AND WIRELESS COMMUNICATIONS, BUT IS ALSO IMPORTANT IN APPLICATIONS WHERE ACCESS CONTROL, DATA INTEGRITY, CONFIDENTIALITY, AND AUTHENTICATION ARE REQUIRED. FOR THIS REASON, CRYPTOGRAPHY WILL FIND ITS WAY INTO A HOST OF COMMON DEVICES, INCLUDING BANK ATMS, KIOSKS, INFORMATION PORTALS, VIDEO SURVEILLANCE EQUIPMENT, BUILDING ACCESS CONTROLS, AND THE LIKE.

ONE OF THE MOST POPULAR CRYPTOGRAPHIC ALGORITHMS IS THE DATA ENCRYPTION STANDARD (DES). DEVELOPED IN THE SEVENTIES, IT STILL SUSTAINS REASONABLE SECURITY LEVEL, WHEN RUN THREE TIMES AS TRIPLE DES.

THIS OVERVIEW GIVES A SHORT INTRODUCTION TO COMMON APPLICATIONS AND TO IMPLEMENTATION ISSUES, WITH A SPECIAL EMPHASIS ON SOFTWARE ASED, HARDWARE BASED AND SOFTWARE HARDWARE CO-DESIGNS.

DES/3DES: Applications

Applications can be found everywhere, where data has to be secured. Data storage and networking are centres of interest:

- DES/3DES is used for Virtual Private Networking (VPN).
  - In Secure IP (IPsec)-based implementations
  - In SSH-based implementations
  - In SSH2-based implementations
- As a consequence, DES/3DES is used for secure extranet protocols, such as Citrix Extranet or European Network Exchange.
- DES/3DES is also used for secure Storage devices, such as Smartcards. A prominent example for the species is “SIM Application Toolkit”.
- But also other applications ranging from XML encryption to secure eGovernment is dealing with DES/3DES.

Many of those applications are based on powerful processors. However, for a great number of new distributed services, the same security requirements are to be met, without the computing power being at hand. In these cases, also small, power-efficient and low-cost microcontrollers can be used.

- For many distributed embedded applications, communication is an important value-added, however puts an additional burden onto the processor, who also has to run the main task of controlling, measuring or displaying. When the amount of data to be encrypted or decrypted is small and no special timing for encryption is required, this may be organized with clever task management or a suitable real-time operating system (RTOS).

- However, when significant amount of data is to be encrypted or decrypted, system designers have two options. They may either use a more powerful CPU, thus leveraging power consumption and system cost. Or they may implement DES/3DES as a crypto co-processor.

Until now, hardware-based cryptography was not favourable for streamline embedded applications, with the exception of smart-card controllers.

- For many use-cases the additional cost of crypto graphic co-processors is prohibitive.

- Monolithic integration together with the target application is rarely available. Exceptions are found with multi-100 MHz microprocessors or 100k pieces per year products.

With System Level Integration (SLI) being at hand for programmable logic devices (PLD), the situation has significantly changed. This is even more the case, when the communication controller is available in hardwired hardware, and additional functionality can be added in programmable hardware. Atmel provides with its Field Programmable System Level Integrated Circuit (FPSLIC™) a powerful single-chip solution.

DES/3DES: Algorithm

DES is a symmetric block cipher algorithm that ciphers 64-bit blocks of clear text into 64-bit of cipher text. The encryption itself is based on XOR-operations and – most notably – on lookup-tables, called S-boxes. The encryption is performed in 16 stages of identical functionality, but using different keys. Those keys K1, K2, etc. are 48-bits wide and derived from 56-bits of the input key K. Additionally, block oriented permutations are performed.

The encryption flow is shown in Figure 1 below.

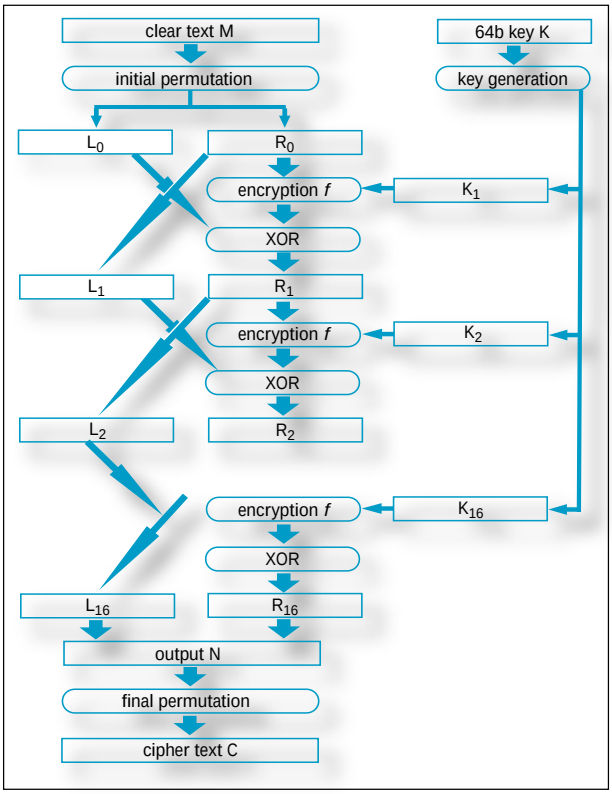


Figure 1: DES encryption flow

DES/3DES: Implementations

DES/3DES: Software Implementation

The implementation of DES in software on an 8-bit microprocessor is possible, however poses significant challenges on the computing power of the underlying processor. Line-to-column-permutations and mangling functions require a significant number of cycles for copying and normalization.

stzedn provides a software library as a demo application for Atmel FPSLIC. This library features:

- access to the internal SRAM,
- implementation of DES and optionally 3DES algorithm,
- key length 64- or 192-bit
- extensive comments, how to use the API

The software library can be compiled in two modes: DES or DES3.

DES/3DES: Hardware Implementation

When significant amount of data is to be encrypted or decrypted, system designers have two options. They may either use a more powerful CPU, thus leveraging power consumption and system cost. Or they may implement DES/3DES as a crypto co-processor.

As the DES-algorithm runs through 16 identical ciphering stages using different sub-keys, there are two different options to optimize the design:

- A speed optimized approach realizes all stages in a pipeline, delivering 64-bit encrypted or decrypted data with each second clock cycle. At a clock rate of 25 MHz, a data throughput of 100 Mbytes/s can be achieved. However, the required resources are enormous, as approximately 2000 registers are needed. In most cases, this high performance is not needed.
- Therefore, an area optimized version realizes only one stage cycles the data through this one stage. Encrypting a data set of 64 bytes requires 16 x 3 = 48 clock cycles, which delivers a data throughput of 4.2 Mbytes/s, which seems more than sufficient for most cases.

The hardware library implements

- access to the internal SRAM,
- DES & 3DES algorithm in hardware,
- easy-to-use finite state machine.

DES/3DES: Software-Hardware-Co-Design

Whereas the pure hardware approach is very strong with continuous data streams, one of its major caveats is the reduced flexibility in terms of key and data selection and the difficulties of integration into a control-oriented application. Therefore, a hardware-software-co-design seems to fit for a great number of applications.

Atmel's FPSLIC prepares an excellent platform for HW/HW-Co-Design. stzedn provides a VHDL based hardware library for use in FPSLIC. The hardware ciphering is controlled by AVR software. Key exchange and message communication between FPGA and AVR is done via the Dual-Port-SRAM, accessible for both parties. Additionally, the finite state machines of both portions communicate via control lines.

The software-hardware-co-design shows the best compromise for system level integration and enables a low-cost crypto-coprocessor with very reasonable performance. The performance of the hardware portion remains unchanged to the pure hardware implementation. Only eight additional clock cycles must be considered for reading or writing 64-bit data blocks from or into the DP-SRAM, as the width of the data bus is restricted to eight lines. The block-diagram is shown in Figure 2.

DES/3DES: Outlook

Using FPSLIC gives an optimum fit for many applications, which may even be leveraged when (partial) reconfiguration comes into the game. In these cases, the crypto hardware can be loaded only for those cases, when data bursts are to be processed. Other functionality may be re-loaded in the same hardware-cells.

Dynamic reconfiguration may also be used for hardware-based generation of keys, being switched to encryption algorithms afterwards. Further improvement on the security and the performance could be achieved by fixing the value of the key to a constant.

	Software	Hardware (speed optimized)	Hardware (area optimized)	Hardware-Software
Key generation	72300 cycles	32 cycles	32 cycles	32 cycles
Ciphering 64b block	124000 cycles	32 cycles	1700 cycles	1700 cycles
Pipelining	no	possible	no	no
ROM [bytes]	3516 bytes			3540 bytes
RAM [bytes]	532 bytes			941 bytes
LUT		~4100	~250	141

Table 1: Results of DES cipher function in software

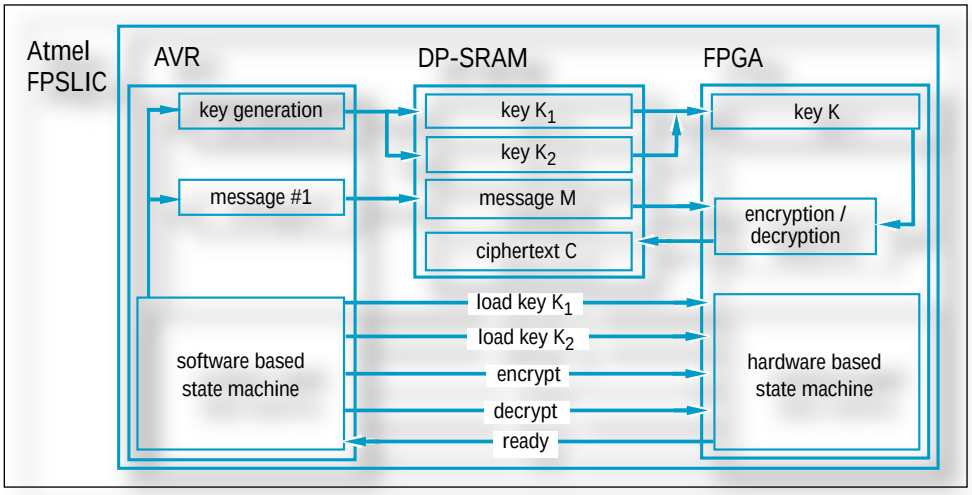


Figure 2: DES design in Hardware-Software-Co-Design

STZEDN

Steinbeis-Transfer-Center for Embedded Design and Networking (http://www.ba-loerrach.de/stzedn), founded in 2002, is led by Dr. Axel Sikora, Professor at the University of Cooperative Education, Loerrach, and Head of Information Technologies Department. He has long scientific and industrial experience in the semiconductor field and is author, co-author and editor of several textbooks and numerous papers.

The Transfer-Centre works in the interdisciplinary fields of Hardware-Software-Co-Design (combined products with MCUs, MPUs, DSPs and ASICs), preferably for networking applications. The Transfer-Centre has developed emBetter, a TCP/IP-stack optimized for use in 16- and 32-bit microcontrollers.

Wireless MP3 Remote Jukebox with Atmel AT90S2313-4PC

WE'VE GONE FROM RECORDS AND 8-TRACKS TO CDS AND MP3S. CONVERTING TO MP3S HAS BECOME POPULAR, AND HOW TO STORE THE FILES IS AN INTERESTING TOPIC THESE DAYS. TUNE IN, BECAUSE THIS PROJECT COVERS A STORAGE SCHEME AND EXTENDS YOUR CURRENT MP3 LISTENING AREA.

Reprinted from
Circuit Cellar #134

By: Brian Miller

Regardless of the success of various music file sharing services, it's a safe bet that many computer users are converting their record and CD collections to MP3 files and storing them on the ridiculously large hard drives available in modern computers. Tiny MP3 players compete with CD and tape walkmans for portable use. With many computers powered up continuously for Internet access, it occurred to me that it would be nice if my computer could act as the "server" for a wireless MP3 jukebox.

Let me explain the concept in more detail. The idea is to allow you to listen to MP3 music files at any location within the home where there is an FM receiver (which could be a walkman, for example). This is made possible by feeding the server computer's audio output to a low-power FM broadcaster. A portable unit displays the contents of the server's MP3 file folders to allow you to browse through your music collection.

For convenience, you can scroll through up to four different folders containing lists of songs. The functions of picking a particular song, starting, stopping, and skipping a song are accomplished using a universal IR remote control. These commands are then forwarded to the MP3 server computer via a 433-MHz wireless link. Photo 1 shows all the components of the system.

For the server end, I designed two modules, a 433 MHz receiver and FM broadcaster. The receiver picks up commands sent by the remote control unit and feeds them to the server computer via a serial port. A dedicated PC application running in the background receives these commands and dispatches them to the Windows Media Player, which then plays the requested selection. The second module, the FM broadcaster, gets audio from the sound card output of the computer and broadcasts it.

To reduce cost and simplify the design, the 433 MHz wireless link operates in one direction only. That is, after you have selected a function, that command is sent to the MP3 server computer via the wireless link. If that transmission never makes it to the server, you will hear that nothing has happened and can issue the command again. However, to ensure that spurious commands don't disrupt operation, a dedicated decoder/encoder chipset is used. This performs all necessary functions to ensure that only legitimate packets are passed to the MP3 server.

For the remote unit to be able to display the contents of the MP3 folders on the server, it must have the contents of those folders downloaded to it prior to use.

Photo 1: This photo shows the various parts of the system. The mini-box to the left houses the 433-MHz receiver. The case with the large LCD is the MP3 remote control unit. Sitting above it are the RCA remote control and the FM broadcaster module. In normal use, only the FM broadcaster and receiver are located with the host computer, the remote controller is placed wherever you want to listen to the MP3 music.

This is done via the serial port on the MP3 server computer using the same application software that passes the incoming wireless commands to the Microsoft Media Player. The firmware in the remote unit can handle up to four different music folders.

To avoid the need to constantly update the remote unit's flash memory, it's recommended that four stable folders are chosen for remote playing, with other folders used for newly downloaded music or selections that are always changing. The remote unit contains a 32K x 8 flash memory, which can hold up to 200 song titles per folder (800 total). Because flash memory is nonvolatile, it will hold this song database even when the unit is off, which is important because the remote unit is battery-operated.

Remote Control Unit

The heart of the system is the remote control unit shown in Figure 1. It is built around an Atmel AT90S2313-4PC which has 15 I/O lines (12 of which are used) as well as an internal hardware UART. Because the unit requires so little power to operate, I chose to power it with four AA cells followed by an LM2936CZ5 low-dropout regulator.

It became obvious early on that the song display would not be useful unless it was able to display the complete song title and/or artist's name. Because there can be a lot of songs to scroll through, I also decided that four songs should be displayed simultaneously. Therefore, I chose a 4 x 40 LCD panel. This unit has a different wiring scheme than most common LCDs. It contains two HD44780 controller LSIs, one for the top two lines and another for the bottom two. The LCD

connection to the micro is made through seven lines of port B using the common 4-bit data interface. The control lines consist of an RS line and two ENABLE lines, one for each controller LSI. The LCD is sent commands only, no status is read back, so the R/*W line is tied low.

Rather than putting a keypad or a lot of switches on the front panel, I chose instead to install an IR detector module and use a universal IR remote control for the user interface. The IR remote is a commonly available RCA model CRCU410 set to emulate a Quasar TV (code 054). I chose this because it's a simple IR code that is easy to decode in software. The keys that are decoded and their functions are shown in Table 1.

Nonvolatile memory storage of the song lists is provided by a serial flash memory EEPROM. I used Microchip's 24LC256I/P8EA because it's commonly available. This chip is an I2C device, therefore it needs only a two-wire interface to the '2313 micro. Unfortunately, the '2313 doesn't contain a hardware I²C port, so a bit-banged software routine must be used instead. However, fortunately Atmel provides an app note describing I²C read/write routines for the '2313 when used as the master device.

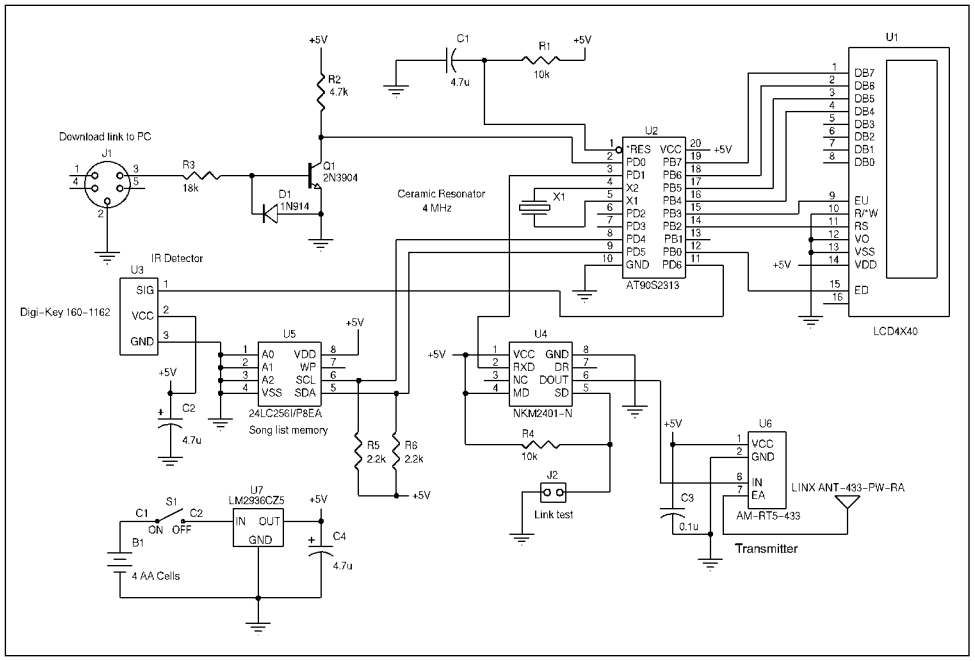


Figure 1: The MP3 remote controller is run by a AVR device.

Note that 2.2 kW pull-up resistors are required on both the SDA and SCL lines per the I²C specifications. The 24LC256I can assume up to eight different I²C addresses allowing for flash memory expansion to 256 KB, depending on the state of the A0 through A2 pins. Because you're using only one device, all three address lines are tied low.

The datasheet for this memory device states a 5-ms flash memory write cycle time. The flash memory is written to only during a data download, at which time the data comes in via the receive section of the UART of the '2313. My download protocol is strictly one-way from the server PC to the remote unit, using no handshaking. Therefore, a data rate of 1200 bps was chosen, which places the incoming data characters 8.3 ms apart. This period allows sufficient time to send the data to the I2C flash memory, even using software I²C routines, and have 5 ms to spare for the actual EEPROM write.

I didn't use a full-blown RS-232 interface like a MAX232. Instead, I converted RS-232 levels of the host computer to TTL with a single 2N3904 NPN transistor and a few passive components.

I chose an Abacom AM-RT5-433 for the wireless transmitter module. Conveniently, the AM-RT5-433 is contained in a small SIP package that's easy to mount. Abacom was kind enough to send me a sample of both the transmitter and receiver modules. These inexpensive modules are meant to cover a distance of 100 feet or so, using simple carrier on/off modulation for data transmission.

My other experience with small wireless modules was with the more costly Linx HP-II series (900 MHz). These are FSK units, the transmitter can be fed directly from a UART, and the receiver is a squelched unit that feeds a UART directly.

Luckily Abacom's technical staff pointed out that the low-cost units I had chosen can't be directly interfaced to a UART port. However, Abacom has designed a combo chip called the NKM2401-N, which can function as either a data encoder or decoder depending on the wiring of the mode pin. For this project, I use one of these devices at each end of the wireless data link.

In the remote unit, the NKM2401 accepts an 8-byte packet from the UART of the '2313 (at 2400 bps), adds sync, pre-/post-amble bytes and CRC, and performs Manchester encoding on the resulting data. Because the NKM2401 requires an 8-byte packet and my commands are only 2 bytes long, I add my own sync and filler bytes to make up an 8-byte packet. The MK2401's data output is connected directly to the data input of the AM-RT5-433 transmitter. For an antenna, I use a 1/4 wave whip.

I was hoping to be able to do without the encoded NKM2401 in the remote unit, instead depending on some '2313 firmware routines to perform the same functions. The remote unit's firmware is written in assembly language and uses only about one-third of the AT90S2313's 2 KB flash memory. There would have been plenty of room for the necessary routines there. However, at the receive end, I didn't plan to use a microcontroller, so I had to use an NKM2401 for decoding there. I was unable to convince Abacom to give me the exact protocol it uses for communication, which is understandable. Therefore, I wasn't able to write routines to do the encoding of the packets. Given more time, I could have captured the datastream using a digital scope or a program running on a micro and reverse-engineered it, but it wasn't in the cards.

Before moving on, let me mention a few miscellaneous details about the remote unit. The AT90S2313 operates at 4 MHz using a ceramic resonator. This is accurate enough for the slow serial data communications rate used. I had to set the UART to receive at 1200 bps (during data download) but transmit at 2400 bps (sending commands via NKM2401 and transmitter). The two rates are necessary because the NKM2401 works only at 2400 bps, and 1200 bps is the highest speed that can be used for downloads (considering the write cycle time of the serial flash EEPROM).

Lastly, there is a jumper labeled J1 Link Test on the remote control unit. When in place, the NMK2401-N sends out a constant "ABACOM" message, which can be used to check the wireless link.

Wireless Receiver Module

The job of the wireless receiver is to pick up the 433-MHz signal transmitted by the MP3 remote control unit and convert that signal to RS-232 level data for the server PC.

The Abacom AM-HRR3-433 receiver is shown in Figure 2. The receiver module is connected to the same type of 1/4 wave whip antenna as the remote transmitter.

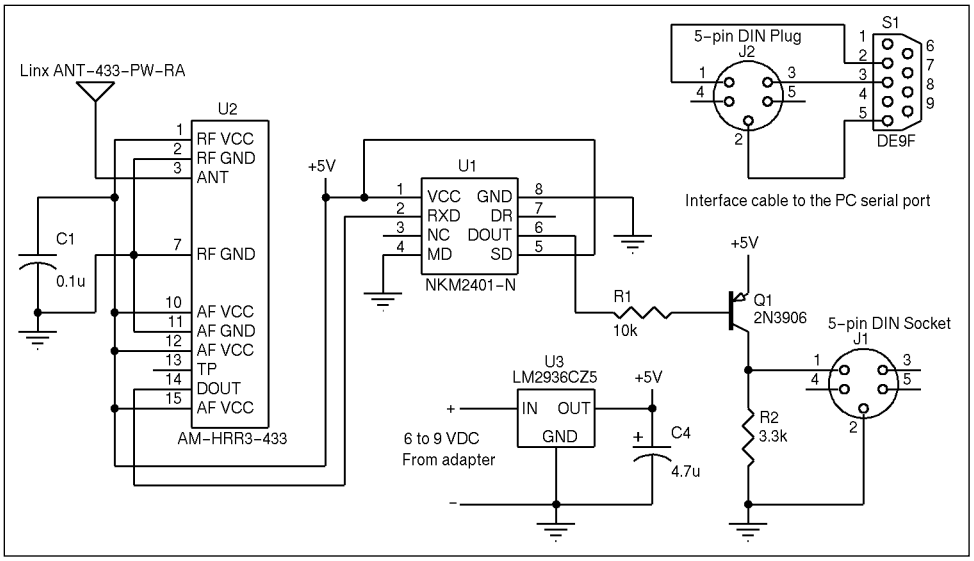


Figure 2: The 433-MHz receiver receives the wireless commands sent by the remote controller and passes them to the server computer for execution.

The output of the receiver module is full of spikes and noise in the absence of an actual signal coming in from the transmitter. I monitored it using an oscilloscope, and I live in a "quiet" RF rural area! For that reason, using Abacom's NKM2401-N decoder chip was a necessity. In the receiver, the NKM2401-N has its mode pin (pin 4) tied to ground to place it in Decode mode. A simple PNP transistor inverter is used to provide pseudo-RS-232 level signals to the server PC. An LM2936CZ5 low-dropout regulator is used to provide the 5 VDC needed for the receiver.

Most of the time this receiver module is left connected to a serial port on the MP3 server computer. However, from time to time, the remote control unit must be connected to the PC in its place (i.e., when the song lists are being downloaded to it). For that reason, I made up a short cable to connect the DB9 male socket on the PC to a five-pin DIN plug. Both the receiver and remote control unit use matching five-pin DIN sockets, and you simply hook up the proper unit as needed.

The Abacom receiver/transmitter modules, when used with NKM2401-N devices, are reliable. The wireless link for the command transmission was one of the smoother aspects of the project. The only complication I ran into was that I wasn't able to place the 433 MHz receiver and FM broadcaster modules in the same case. When the FM broadcast transmitter was placed next to the receiver, its RF output slightly desensitized the receiver. The result was that the wireless link would work for a distance of only about 20 feet, which was too short for my purpose. However, when I placed the FM broadcaster in its own case and moved it several feet away from the 433 MHz receiver, the problem disappeared and the range increased to about 50 feet (keep in mind, that's still within the house).

FM Broadcaster

When I originally conceived this project, I anticipated some challenging design or programming problems. I assumed getting a small FM broadcaster module would be easy, so I tackled that job last. However, of course, Murphy's Law dictated that this would turn out to be the most frustrating and time-consuming part of the project!

I had heard rumors that stereo FM transmitter kits based on the Rohm BA1404 IC were too unstable to be useful. Not easily deterred, I bought such a kit anyway. Alas, the rumors were true—its frequency stability is too poor to work with modern, digitally-tuned FM receivers. Even when I replaced the cheap RF tuning components with high-quality parts, the problem remained. In fairness to Rohm, I expect that this IC was designed before the advent of digital FM receivers. Older analog FM receivers had an automatic frequency control circuit that likely would have tracked the frequency changes that occur with this transmitter kit.

I found a PLL-stabilized FM broadcaster kit, but it cost about $200, which was too steep for this project. A number of years ago, I built several PLL-based frequency generators in the 10 to 400 MHz range, so I thought I'd try to roll my own FM broadcaster.

That's when the trouble began. The PLL chips that I had used in the past were no longer available. Most of the currently available PLLs are meant for cell phones and the like and won't work reliably below 100 MHz. I found some that were targeted for the FM broadcast band, but they were in such tiny packages that I couldn't solder anything to them.

At this point, I decided to try something different. Because I needed a microcontroller to load the PLL chip anyway, why not try to let the microcontroller measure and control the oscillator frequency and dispense with the PLL chip entirely? What I had in mind could be considered an automatic frequency stabilizer.

The basic concept is demonstrated in Figure 3. The oscillator frequency is determined primarily by the values of the inductor and variable capacitor. For this design, I chose an overall tuning range of roughly 88 to 92 MHz for two reasons. First, there are fewer commercial FM stations at the bottom of the band. More importantly, 96 MHz is the highest frequency that can be measured using this circuit.

Fine-tuning of the oscillator, both for stabilization and FM modulation purposes, is handled by a varactor diode. The bias on the varactor sets its capacitance. This bias has two components. A DC voltage level is applied by a 12-bit DAC, and an AC signal is superim-

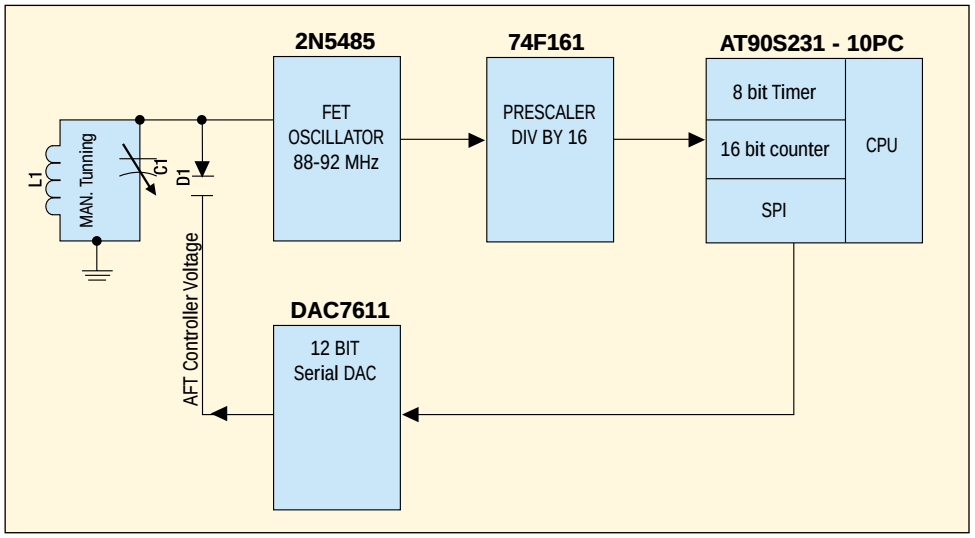


Figure 3: The block diagram shown here outlines the FM broadcaster.

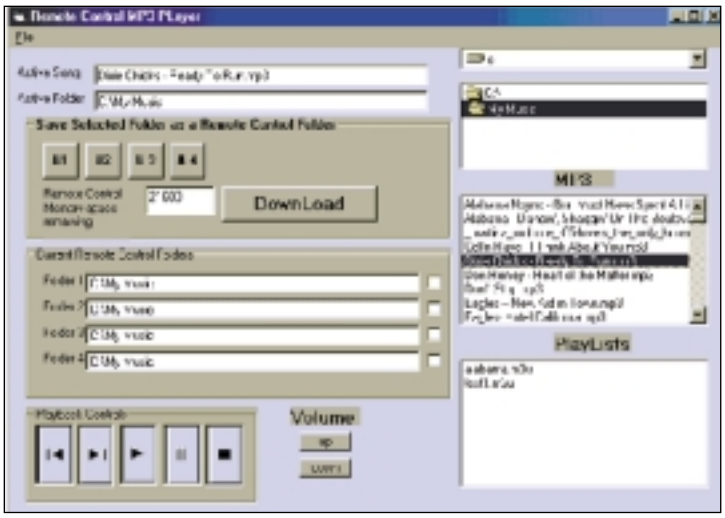


Photo 2: This is the PC application program that controls the playing of MP3 files through the Media Player, from wireless commands received from the MP3 remote controller. The code's other main function is to allow you to choose up to four music folders and then download the song titles (file names) to the flash memory in the remote controller.

posed on it to perform the frequency modulation. The DAC output voltage is initially set to a middle range (2 V) and you manually tune the oscillator to the desired frequency by adjusting the variable (trimmer) capacitor. Thereafter, the microcontroller will adjust the DAC voltage up and down slightly to maintain the proper frequency.

For the microcontroller to measure the oscillator frequency, it must first be prescaled by a factor of 16. This is done using a common 74F161 4-bit divider chip. The prescaler output is in the 5 to 6 MHz range, which can be directly counted by the microcontroller using its 16-bit counter/timer.

To determine the oscillator frequency, clear the 16-bit timer and then read it after a fixed interval. This fixed interval is provided for by another counter/timer in the microcontroller, which is programmed to provide a periodic interrupt every 5.461 ms. Using this arrangement, the expected value in the 16-bit counter equals:

$$\text{counter/timer Value} = \left(\frac{f_{osc}}{16}\right) x\,(5{,}461x10^{-3})$$

In operation, the microcontroller reads the count in the 16-bit timer and compares it to a fixed constant derived from this equation using the FM broadcast frequency chosen by you. If the oscillator frequency is too low, it bumps up the DAC value by one and tries again. Conversely, if the oscillator frequency is too high, it decreases the DAC value by one. This process is repeated until the oscillator frequency falls within a narrow band around your set point frequency.

The circuit described here normally would be hunting constantly for two reasons. First, there always would be a one-count bobble in the counter/timer value because of variations in the position of the sampling interval with respect to the oscillator signal. Secondly, because the oscillator is being frequency-modulated by an audio signal, its frequency will vary from this modulation voltage.

This hunting is undesirable because it produces artifacts in the music heard through the FM receiver. To prevent this, as soon as the microcontroller has tuned the oscillator so that the 16-bit counter/timer is within two counts of ideal (an accuracy of about 6 kHz), it will "go to sleep" for 10 minutes, after which time it will check the frequency again. Unless the room temperature changes significantly, the frequency of the oscillator will require little correction, and this circuit will apply it as necessary.

AVR AFC

While I describe how I implemented the automatic frequency-controlled FM broadcaster in detail, refer to Figure 4 on page 35for a visual representation.

To begin, I needed a microcontroller that could count pulses at a 6 MHz rate, also containing another timer to generate the periodic interrupt needed to read and clear this counter. I chose the Atmel AT90S2313-10PC because it has the required functionality. However, I have to run the device at 12 MHz instead of its rated 10 MHz to achieve the 6 MHz counting rate. By the way, I have not experienced any problems "over-clocking" the '2313 by this modest amount.

The setting of the FM oscillator frequency will be performed only once, when the unit is first set up and a clear channel is found on the FM dial. Therefore, to make things simple, the desired broadcast frequency is entered into the program code as a constant at the beginning of the program. The program is then compiled and downloaded into the '2313, resulting in a fixed-frequency FM transmitter.

Ten years ago, you could find prescaler chips readily available that would divide by 256 at frequencies up to 1 GHz, but these are no longer manufactured. So, I used a 74F161 four-stage counter to implement a divide by 16 prescaler. It handles frequencies greater than 100 MHz, costs less than $1, and is readily available.

I built my own VCO around a 2N5485 FET device. The resonant frequency of the VCO is largely determined by the values of L1 and C12. The latter is a trimmer capacitor that is adjusted manually when the unit is first powered up to achieve roughly the desired frequency. This tuning is performed with jumper J1 in place, which causes the microcontroller to set the DAC to mid-scale.

Thereafter, the jumper is removed. And when the unit is powered up again, the automatic frequency stabilization circuit starts to operate and the oscillator is fine-tuned to the desired frequency by D1, the varactor diode. The 74F161 prescaler requires several volts of signal for proper clocking. The FET oscillator can provide this (most other oscillator configurations are not capable).

The RF output from the oscillator is taken from a tap on L1 to reduce loading effects. This is capacitor-coupled to the clock input of the first stage of the 74F161 prescaler. Potentiometer R7 is adjusted to provide the proper bias on the clock input pin, allowing the oscillator signal to properly trigger the input divider stage. It should be set for around 2 to 2.5 V, but is best set up by using an oscilloscope and looking for a clean 5 to 6 MHz waveform on pin 11 of the 74F161.

The 12-bit DAC (U4) that controls the VCO's fine-tuning is a TI DAC7611, with an SPI interface. Although the '2313 doesn't have a user SPI port (its SPI port works strictly for flash memory programming), it's simple to bit bang the SPI data out to the DAC using I/O lines PB1 through PB4.

Shown just below the '2313 in Figure 4 is jumper J1, which is connected to port line PD6. At reset, the microcontroller checks the state of this line; if the jumper is in place, it merely sets the DAC to mid-scale (2.048 V) and waits. This allows you to set the mechanical trimmer capacitor (C12) for an oscillator frequency as close as possible to the desired FM broadcast frequency. In North America, all FM broadcasting is done at odd multiples of 100 kHz, so pick a frequency accordingly.

Having accomplished this, next you remove the jumper and powerup the unit again. The device should home in on the desired frequency within a few seconds by repeatedly adjusting the DAC voltage and measuring the resulting frequency via the prescaler. This is the normal operating mode when subsequently being used as an FM broadcaster module.

Line-level stereo audio from the host computer's sound card line out is first run through a pre-emphasis network (for each channel), and then mixed down to a monaural signal. This low-level AC signal is superimposed on the DC control voltage of the DAC and used to frequency-modulate the oscillator.

The RC values in the pre-emphasis network were

picked by monitoring the output of a stereo receiver and aiming at a flat-frequency response. The values shown in Figure 4 come pretty close. Don't expect the chosen component values to provide a time constant of 75 µs, which is the standard pre-emphasis used by broadcasters. Consider that there are many other factors in the VCO that affect the modulation characteristics. My values provide an overall flat-frequency response as measured with a high-quality FM receiver.

Note that no antenna is illustrated in Figure 4. If the project is placed in a plastic enclosure, it will radiate enough signal to cover a 50¢ radius. This works out well, because government regulations prevent using a transmitter capable of covering a much greater range than this.

I built the VCO section of the circuit (shown within the dashed lines in Figure 4) on a small, single-sided PC board with dimensions of about 1.5" x 1". The remaining part of the circuit was hand-wired on a Simm-Stick protocard. The VCO PCB is designed like a SIP package and mounts vertically on the Simm-Stick.

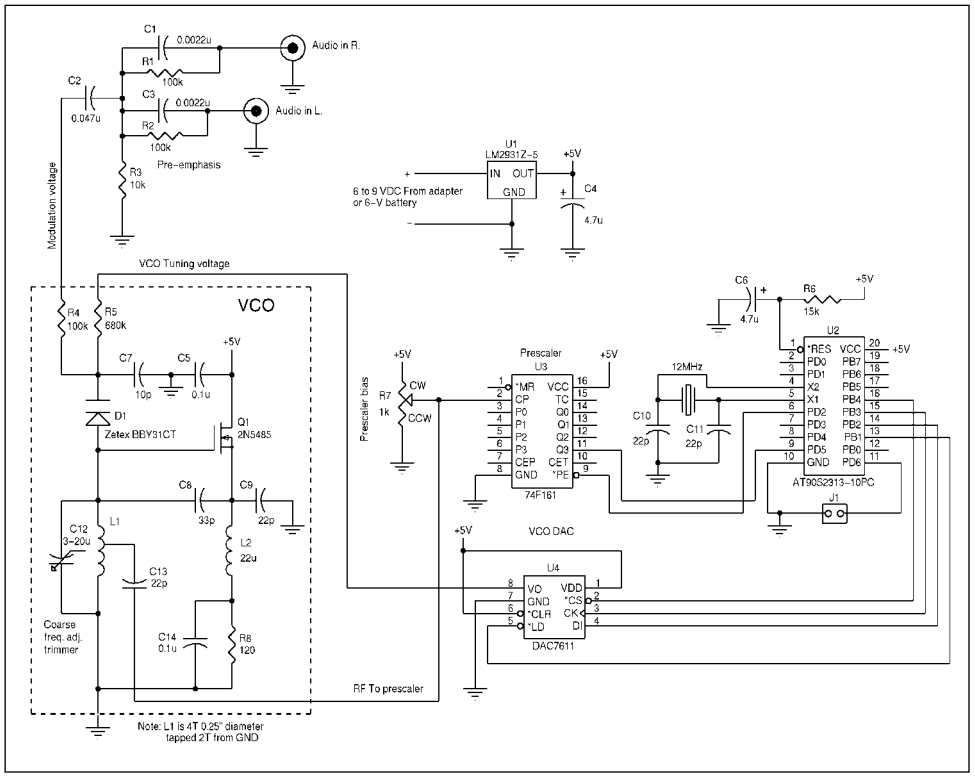


Figure 4: The FM broadcaster unit is automatically fine-tuned.

Software and Firmware

The application software consists of a server application running on a computer and client firmware running on the remote controller. At the PC end, the server software is written in Visual Basic 6.

The remote controller firmware is written in AVR assembly language. The FM broadcaster is frequency stabilized by another AT90S2313. The program required for this is trivial, so I used the BASCOM-AVR compiler for that application.

Client Application Software

The client application running on the PC has two main functions. Most of the time it is polling the COM1 serial port looking for commands that have been sent to it by the remote controller. Its other main function is to allow you to browse through your directory structure and designate up to four folders as jukebox folders.

The file names in these folders are then converted into a database record and sent to the remote controller using the transmit section of the COM1 serial port. This download is performed only once unless the contents of the folders change, because this data is stored in the song list flash memory within the remote controller (Photo 2).

Let's look at the first function in more detail. As mentioned earlier, the connection between the computer and remote controller is a 433 MHz wireless link. At the PC end, the 433 MHz receiver takes the RF signal and converts it into serial data at 2400 bps, which is fed into the COM1 port. All data formatting and error checking are performed in the hardware using the NKM2401 encoder/decoder chips. This method ensures that any command received by the client application will be legitimate.

The NKM2401 uses 8-byte packets. All commands sent by the remote controller consist of a 2-byte sync pattern (0xAA, 0x55) followed by a 16-bit command word and four dummy bytes. Two bits of the command word are used to designate the four commands: Play (Stop), Play Next, Play Last, and Pause. The other 14 bits are used to select both the active folder and offset of the song's file name within that folder.

Earlier, when you picked out the folder for use by this application, that folder was parsed and a fixed record length database was generated, a copy of which is maintained both by this application (in a file) and in the remote controller (in the song list flash memory). Doing it this way allows you to see all of the song names on the remote LCD readout. In addition, it means you can select songs to play by merely sending a number corresponding to the song's position in the database file.

After the client program knows what song to play, how does it actually get the computer to play that particular song? If the songs were in WAV format, it would be easy, as Visual Basic has built-in support for multimedia functions including the playback of WAV files. However, it does not support the playing of MP3 files, at least not the version that I have. I looked around for a shareware/freeware ActiveX control that plays MP3 files but wasn't able to find one that was free or reasonably priced.

I took a different tack at this point. I had just downloaded the Windows Media Player V.7 that Microsoft distributes for free. This multi-purpose program handles MP3 files nicely and has all the bells and whistles people want. Like many Windows programs, it allows for keyboard shortcuts, an advantageous capability. My idea then was to run my MP3 Jukebox client application and the Windows Media Player concurrently and let my client application send keyboard strokes to the Media Player to control it.

This is done using a couple of Visual Basic commands. The Shell command transfers program flow over to the Media Player, and the Sendkeys command sends the proper codes to the Media Player, causing it to select the desired song and play it or do some other functions. To check that the client program is actually communicating with the Windows Media Player, I added the capability of picking a specific song from a folder and playing it (without a command coming in from the remote control). As Photo 2 demonstrates, you may choose the song to be played using the drive, folder, and file windows on the right, and the transport controls located along the bottom will play or stop it.

The second function, used occasionally, chooses the desired music folders (up to four) and downloads the

contents into the remote controller's flash memory. You do this by picking out a specific folder and then clicking on one of the four numbered buttons located to the left of the form. A window keeps track of the space remaining in the song list flash memory. Each folder can contain almost 200 songs before its flash memory allocation is exceeded.

To simplify the firmware in the remote controller, I make the assumption that all four folders will be used, and therefore downloaded. If you have fewer folders, the remaining folders should be designated as duplicates of the desired folder(s).

Before pushing the Download button, the remote controller must be plugged into the COM1 port of the PC, temporarily displacing the 433 MHz receiver. I use the custom cable described earlier, which stays plugged into the PC. The other end of the cable is a five-pin DIN plug, which fits sockets on either the receiver or the remote controller. The download time is dictated by the write timing of the flash memory in the remote controller. Download time is about 4 minutes if all four directories contain the maximum of about 200 songs. The download progress is indicated by check boxes appearing next to the client program's folder list as well as messages on the remote controller's LCD screen.

AVR Firmware

I already outlined the automatic frequency control of the FM broadcaster. The program that accomplishes this is simple, and therefore was written in Basic and compiled using the BASCOM-AVR compiler. For more details, download the program listing contained in the file MP3FM.bas, which is available on Circuit Cellar's web site as well as my personal web site.

The remote controller firmware was much more involved. I first tried to write it in Basic using the BASCOM-AVR compiler, but the program code generated would not fit into the flash memory of the '2313. Using assembly language, I accomplished the same thing in less than half of the flash memory space.

As with the PC client software, the remote controller performs two functions, one of which happens infrequently. When turned on, it displays the first four songs in folder one. It then goes into a polling loop to detect IR commands sent by the RCA universal remote and received by the IR receiver module. This IR signal comes into port D6 of the '2313, which is the INPUT CAPTURE pin.

The IR codes are deciphered by using the input capture feature of timer1, a 16-bit timer/counter. I chose a simple IR command structure (RCA code 54, Quasar TV) to make my job easier. This command structure has a fixed-length start pulse at the beginning of each command sent, followed by 8 bits of data. The data bits are represented by two different intervals between pulses. After recognizing the fixed length start pulse, you have to do only two things. You must capture the timing of the eight subsequent pulses and, from the interval separating them, assign the proper bit values. As mentioned earlier, the remote control constantly polls the IR receiver for commands and then performs the proper function. Many keys allow for navigation only through the song lists and moving from one folder to another. Do this by adjusting pointers into the storage flash chip, reading the song names, and transferring these ASCII characters to the LCD screen.

Key	Function
One through four	Play one of the four songs currently shown on display
Five	Skip to next song
Six	Jump back to previous song
Seven	Stop/start again
Enter	Prepare for song download from host PC
CH+	Scroll to next screen of four songs
CH–	Scroll to previous screen
Left VOL	Select previous folder (four total)
Right VOL	Select next folder (four total)

Table 1: I defined the buttons of the RCA universal IR remote control to function properly with the MP3 remote controller. The remote control must be set up to emulate a Quasar TV remote for this application (code 54).

The keys used for Play, Play Next, Play Last, and Pause actually send out the proper command via the UART transmit port of the '2313. Again, the NKM2401 uses 8-byte packets, so sync and filler bytes are added to the 16-bit command word as needed.

There is a key dedicated to the download function. When pressed, it shifts program execution to a routine that accepts characters coming into the UART receive port of the '2313. The database generated by the PC is transferred into the song flash memory using this function.

The Atmel 24C256 flash memory chip is an I²C device with a 32K x 8 storage capacity. Its 5 ms write timing isn't a problem, because the data coming in from the server computer is sent at 1200 bps, which is an 8.3 ms per character rate. The '2313 doesn't have a dedicated I²C port so this function must be done in firmware. The master I²C routines were taken directly from Atmel's application note and work fine.

The LCD is a 4 x 40 unit that uses the ubiquitous Hitachi HD44780 controller. Actually it contains two controller LSIs, with a common data/control interface in addition to two enable lines. I had to rewrite my trusty 4-bit LCD driver routines to handle the fact that lines one and two use controller one and lines three and four use controller two.

All Played Out

I found this project to be very interesting, partly because of the wireless aspect. The Abacom receiver/transmitter modules coupled with the company's encoder and decoder devices work well. Moreover, the user-friendly flash programming capability of the AVR device made writing the assembly language firmware nearly painless for me.

However, there was one disappointment. I didn't anticipate spending so much time coming up with a satisfactory FM broadcaster module. Although too late for this project, I recently came across the Rohm BH1416F Wireless Audio Link IC, which contains a complete PLL-stabilized FM transmitter and FM stereo modulator in a SOP22 package. I bought a few of these to try out later. If you're not a big music fan, maybe you'll be able to apply some of the ideas mentioned here for other useful remote control concepts.

Song Quality Improvement Using the DAC on AT89C51SND1 Reference Design

SIGNAL TO NOISE RATIO CAN BE IMPROVED BY CONTROLLING THE SONG LEVEL BOTH WITH THE EXTERNAL DAC AND WITH THE AT89C51SND1 SONG LEVEL SFR REGISTER.

TO CONTROL THE SONG LEVEL WITH THE DAC UDA1330ATS, THE L3 MODE MUST BE USED.

THIS APPLICATION NOTE GIVES SOME INFORMATION ON ACTIVATING/DE-ACTIVATING THE L3 MODE OF DAC UDA1330ATS (OR COMPATIBLE HARDWARE) FOR THE AT89C51SND1 REFERENCE DESIGN.

Application Modes

The DAC UDA1330ATS has two application modes, Standard mode, and L3 mode.

The application mode can be set with the three voltage levels on the APPSEL pin of UDA1330ATS (see Table 1). In standard mode (or static mode), APPLx pins are static input pins. In L3 mode, a serial protocol is specified over the APPLx pins. See Philips® datasheet UDA1330ATS for more information.

Figure 1 and Figure 2 are the functional schematics of the reference design in standard mode and L3 mode.

Voltage on Pin APPSEL	Mode	fsys
VSSD	L3 mode	256fs, 384fs or 512fs
1/2 VDDD	Static pin mode	384fs
VDDD		256fs

Table 1: Selecting Application Mode and System Clock Frequency Via Appsel Pin

Hardware Changes

1. Locate the MAX4468 component on the reference design (see the arrow on the top right side of the Figure 3).
2. Locate the 3 SMD resistor/capacitor aligned between the MAX4468 and the PCB edge.
3. To de-activate the L3 mode, from left to right, solder a 10K SMD resistor, another 10K SMD resistor and keep the SMD capacitor as is. To activate the L3 mode, from left to right, perform a short-circuit in place of the first 10K SMD resistor, remove the sec ond resistor and keep the SMD capacitor as is.
4. Upload the new firmware according to the DAC configuration. Proceed with software changes outlined in the following section.

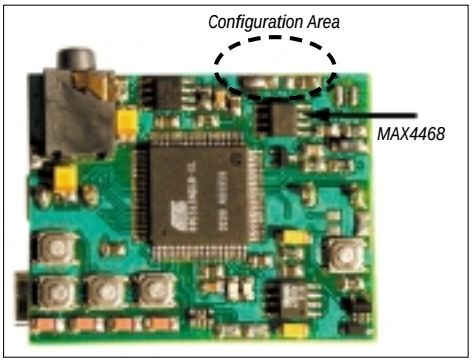


Figure 3: Hardware Changes Location (L3 Mode)

Software Changes

1. Unzip the source code package of the latest version of the reference design firm-ware: snd1c-refd-nf.

The files corresponding to DAC controls are: dac_drv.c and dac_drv.h in the directory snd1-refd-nf-x_x_x/lib_board/dac.

2. Edit the file snd1c-refd-nf-x_x_x/Lib_board/refd.h and comment or un-comment the line #define DAC_L3MODE to respectively de-activate or activate the L3 mode usage.
3. Recompile the project.
4. Upload the new Hex file on the configured or un-configured board in L3 mode according to the software selected (see Section “Hardware Changes”).

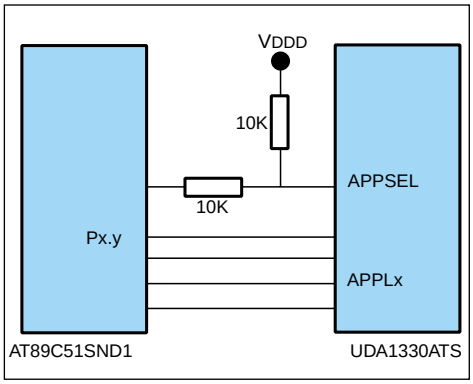


Figure 1: Functional Overview of the Reference Design with DAC in Standard Mode

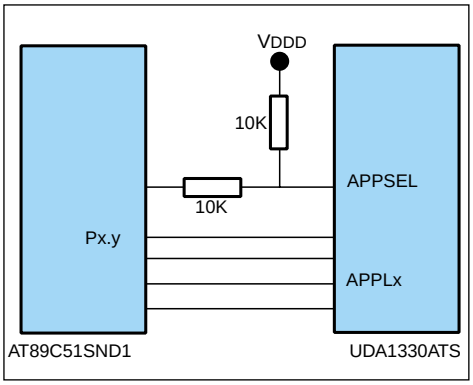


Figure 2: Functional Overview of the Reference Design with DAC in L3 Mode

Some people think we're a little obsessed with security.

We'll take that as a compliment. Because nobody has a bigger commitment to secure ICs than Atmel. And nobody has more security solutions.

That's why we can protect just about any system with the most advanced security features you can find. Our portfolio includes dozens of products that meet the world's toughest standards, including ITSEC E3, Common Criteria EAL4+ and Level 3 approval from Visa® International.

You might also like to know that we're the third-largest supplier of smart card ICs, including chips that power products from many of the world's largest banks and credit card companies. And that over 5 million of our ultra-secure Trusted Platform Modules are installed in some of the planet's best-selling laptops. We've also got a strong story in biometrics, where our FingerChip™ is the world's smallest and one of the most popular fingerprint sensors. And our CryptoMemory is the industry's only secure serial EEPROM.

But that's just the beginning. We also have a broad line of secure microcontrollers and card reader ICs, contact and contactless interface ICs (ISO 14443), plus a huge library of security IP.

So if your next project calls for secure ICs, make sure you call the obsessive engineers at Atmel. We'll do everything we can to help you make a name for yourself.

Check out Atmel's security solutions today at www.atmel.com/ad/securitysolutions

Secure Memory and Microcontrollers for Smart Cards			**Smart Card Reader ICs**
Biometrics	**Trusted Platform Module IC**	**Embedded Security**	**RF Identification**

ATMEL®

Everywhere You Are℠

Easy Image Processing: Camera Interfacing for Robotics

By: Daniel Herrington

SINCE CHILDHOOD, DANIEL HAS INCORPORATED HIGH-TECH TOYS IN HIS ROBOTICS PROJECTS. THUS, IT'S NO SURPRISE THAT HE USED A GAME BOY CAMERA IN ONE OF HIS RECENT DESIGNS. NOW, HE'LL SHOW YOU HOW TO USE THE CAMERA TO ENHANCE THE NAVIGATION SYSTEM ON YOUR OWN MOBILE ROBOT.

Reprinted from
Circuit Cellar #151

I've been interested in robotics since I was a little boy. Back when I was in junior high school, I built a mobile robot platform out of the drive portion of a child's motorized car and a Commodore VIC-20. Over the years, advances in technology have made experimenting with robotics more enjoyable. The Game Boy Camera is an ingenious addition to the Game Boy Color game unit that came out a couple of years ago. It's a black-and-white digital camera with a resolution of 123 x 128 pixels, operating at a rate of one to 30 frames per second.

The camera's original price was between $40 and $50, making it somewhat cost-prohibitive for hobbyists. However, because the product was recently discontinued, I found some on eBay selling for between $10 and $20. The reduced price makes the camera an attractive solution if you're interested in robot navigation. It's even less costly than a single Polaroid sonar module ($30 to $50) and in the same ballpark as reflective infrared sensors ($5 to $15).

The sensor inside the camera is a highly integrated CMOS array with built-in edge enhancement and extraction. Built-in image processing enables a microcontroller to perform object detection and tracking, provided certain assumptions about the background of the image are valid.

Atmel's AT90S8515 microcontroller has an external memory interface bus that allows you to easily connect an SRAM IC. The on-chip hardware UART makes it possible to output processed data without consuming precious processing resources, and the timers enable it to control hobby servo motors without much work. In addition, the AVR series of microcontrollers has a high-speed RISC architecture (e.g., most instructions take one or two clock cycles) that makes timing calculations simple. In short, the flexibility of the AVR 8-bit microcontrollers makes attaching special-purpose peripherals like the Game Boy Camera a breeze.

Figure 1 is a block diagram of a camera interface and object-tracking system. As you can see, the camera is controlled via some of the microcontroller's general-purpose I/O pins. The analog output of the camera is attached to the external A/D converter. The servos are connected to two more pins of the microcontroller, and the RS-232 converter conditions the UART's signals for connection to the outside world.

Figure 2 details the interface circuit. A few notes might be helpful here. The A/D converter needs to be fast enough to read out a pixel value every 2 µs if the maximum frame rate is desired. This means the sample frequency of the ADC must be at least 500 kHz. A speed requirement like this rules out the use of the built-in ADC on most microcontrollers (e.g., the AT90S8535).

For my circuit, I settled on the Analog Devices AD7822, which doesn't have the added complication of pipelining that many of the newer ADCs seem to have. Also, you don't need the RS-232 converter IC if the circuit will be interfaced directly to another microcontroller's UART. I used a 7.3728 MHz crystal to achieve compatibility with standard data rates. A speed of 115,200 bps is the maximum speed that a standard PC serial port supports, so I programmed the microcontroller to work at that speed.

The completed prototype circuit board is shown in Photo 1. It's a simple wire-wrapped board with all through-hole components except the ADC. The potentiometer is used to adjust the reference voltage for the microcontroller's on-chip analog comparator. The comparator is used in place of the ADC for the hobby robot depicted in Photo 2. I found that the optimum setting for the reference voltage for my test environment was about 4.66 V.

Photography 101

The Game Boy Camera uses Mitsubishi's M64282FP CMOS image sensor. This sensor is special because Mitsubishi designed some image processing capabilities right into the CMOS array itself.

Software isn't required to perform edge detection, enhancement, or extraction from the image. To get an edge-extracted image, simply put the sensor in the appropriate mode and then read out the resulting analog pixel voltages for the entire image. After this image is read out, it can be further manipulated by the microcontroller. Photo 3 shows the M64282FP chip inside the camera.

I didn't use the camera's cartridge base in these experiments. After the cartridge is opened, the ball can be disconnected from the cartridge. I disconnected the purple cable from the cartridge and soldered a standard 0.1" double-wide header on the end. This allowed a 0.050" ribbon cable to be used for a long-distance connection, although I don't recommend exceeding 1 foot or so.

By the way, I cut a hole in the back of the ball so that the cable could exit in a place where it doesn't interfere with mounting the camera on a pan-tilt base. You may download the pinout of the original connector that's coming out of the ball from the Circuit Cellar ftp site. The numbers refer to the corresponding pin numbers on the M64282FP chip.

To simplify the interfacing of the assembly code and speed things up, I turned the camera sensor board in

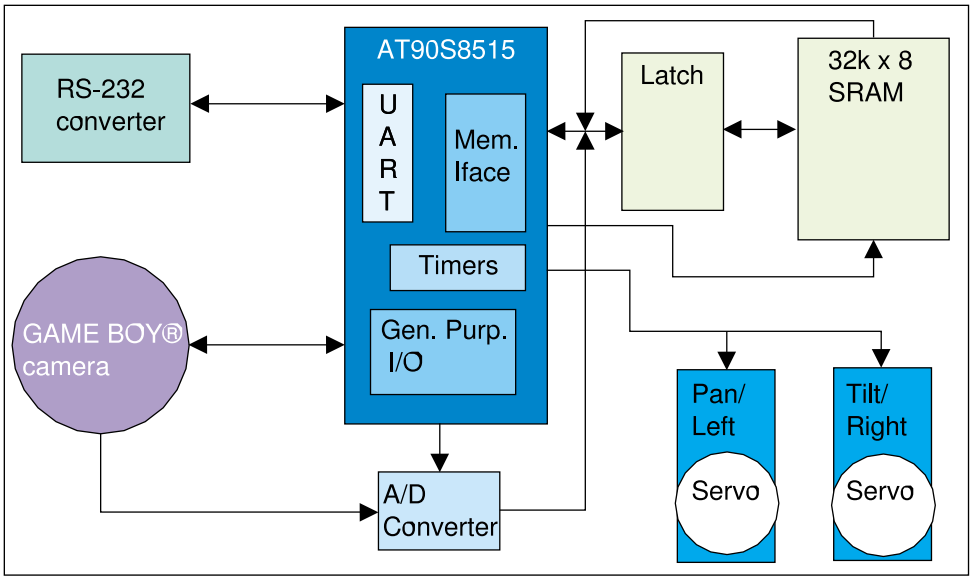


Figure 1: You can use the two servos for either panning/tilting a camera head or driving the left and right wheels of an autonomous robot. For the latter, the servos must be modified to allow for continuous rotation. This servo hack is common for hobby robots.

the camera ball upside-down. This ensures that the first pixels to be read from the camera are those corresponding to the bottom-right corner of the image instead of top-left. Furthermore, this makes the calculation of the nearest object faster, because the image is read out serially from the sensor.

The procedure for programming and using the M64282FP is straightforward. First, load the registers in the M64282FP using a synchronous serial protocol that is similar to other two-wire synchronous serial interfaces. The microcontroller generates the XCK, SIN (data), and LOAD signals for loading all of the registers in the camera IC.

Next, give the Start command. After the exposure is finished, the camera IC will return a READ signal. When the READ signal becomes active, read 15,744 pixels (123 x 128) worth of analog data on the VOUT pin synchronously with the XCK signal that the microcontroller generates. After all of the image data has been output, the READ signal becomes inactive, and the camera automatically starts another exposure.

In the first part of the programming process, you can set the camera's registers for a normal (positive) or inverted (negative) image, an edge-enhanced image, or an edge-extracted image. Register settings also control the camera's exposure time, the output voltage offset level, and the gain of the video signal (i.e., how much it varies from the output voltage offset).

The maximum frequency for the camera's XCK input is 500 kHz (T = 2 µs). With a microcontroller crystal frequency of 7.3728 MHz (T = 135.6336 ns), the time for each half-period of XCK is:

$$\frac{1\mu s}{135.6336ns}$$

or approximately eight microcontroller clock cycles. I tuned the timing of the assembly code by adding NOP instructions where appropriate.

It's interesting to see how different register settings affect the image output from the camera. Table 1 shows two settings: Normal mode and Edge mode. These settings were derived by experimentation and may need to be adjusted for any given environment.

I set up a test area with various medium- and high-contrast colored objects on a light-colored floor (see Photo 4). The top-center image frames within Photo 5a on page 43 show what the Game Boy Camera images look like with specific register settings: Photo 5a is a normal image; Photo 5b is a negative image; and Photo 5c is an edge-extracted image. I used this type of image for object tracking. Note that the light-colored objects (red and orange in this case) don't show up as well in the edge-extracted image. You can increase or decrease the exposure setting to allow these low-contrast objects to be seen in Edge mode.

Obstructed Views

Ian Horswill's 1993 dissertation, "Specialization of Perceptual Processes," details some of his research concerning image processing for robotics. Horswill outlines various assumptions that may be made by a robotic image-processing system for detecting obstacles in a given environment.

After the edges have been extracted from an image, the height of the first edge from the bottom of the image

Register	Address	Normal mode	Edge mode
0	000	0x80	0x3F
1	001	0xD6	0xD6
2	010	0x06	0x18
3	011	0x00	0x00
4	100	0x01	0x01
5	101	0x00	0x00
6	110	0x01	0x01
7	111	0x07	0xF3

Table 1: When you're switching from Normal to Edge mode, it's important to remember the M64282FP registers 0, 2, and 7.

can be determined. Let's assume the camera is mounted somewhere on the front of the robot, several inches or feet above the floor. If the camera is aimed forward and down, and if the floor doesn't have visible edges (i.e., the carpet color is constant, and there are no high-contrast seams or changes of color), then the only edges should be the obstacles on the floor or a floor-to-wall transition in front of the robot.

If the robot moves near a wall, and if there is enough of a contrast between the wall and floor, an edge will be detected at that location in the image. Using this technique, the robot can tell how far away the edge of the obstacle is by its height in the image.

If the image is divided into thirds (i.e., left third, center third, and right third), then the lowest edge in each third of the image gives the robot the distance it can move in that direction. Then, the robot can turn and move toward the farthest direction to avoid the closer obstacles. This "greatest third" approach is well suited for corridor following, because the greatest third of the image is most likely the direction of the corridor.

The camera takes care of extracting the edges from the image, but the microcontroller must perform any additional processing. For instance, if you want to know an object's distance (or depth) from the robot, then you'll need an algorithm to post-process the image and reduce the information down to a depth table. The index to this table could represent a given x location, or column. The entry at each index within the table could be written with the row of the lowest edge pixel in a given column. This algorithm is implemented in the microcontroller as shown in Listing 1.

Now, I'll explain the operation of the depth-finding code. Starting at the bottom-right corner of the image, count the number of pixels vertically until one is reached that surpasses a predefined threshold value. Put the row number (i.e., depth) of that pixel in a table, step to the next column to the left, and repeat the process.

When the depths of all of the columns of the image have been recorded, send that information out of the UART. A graphical representation of the depth map for

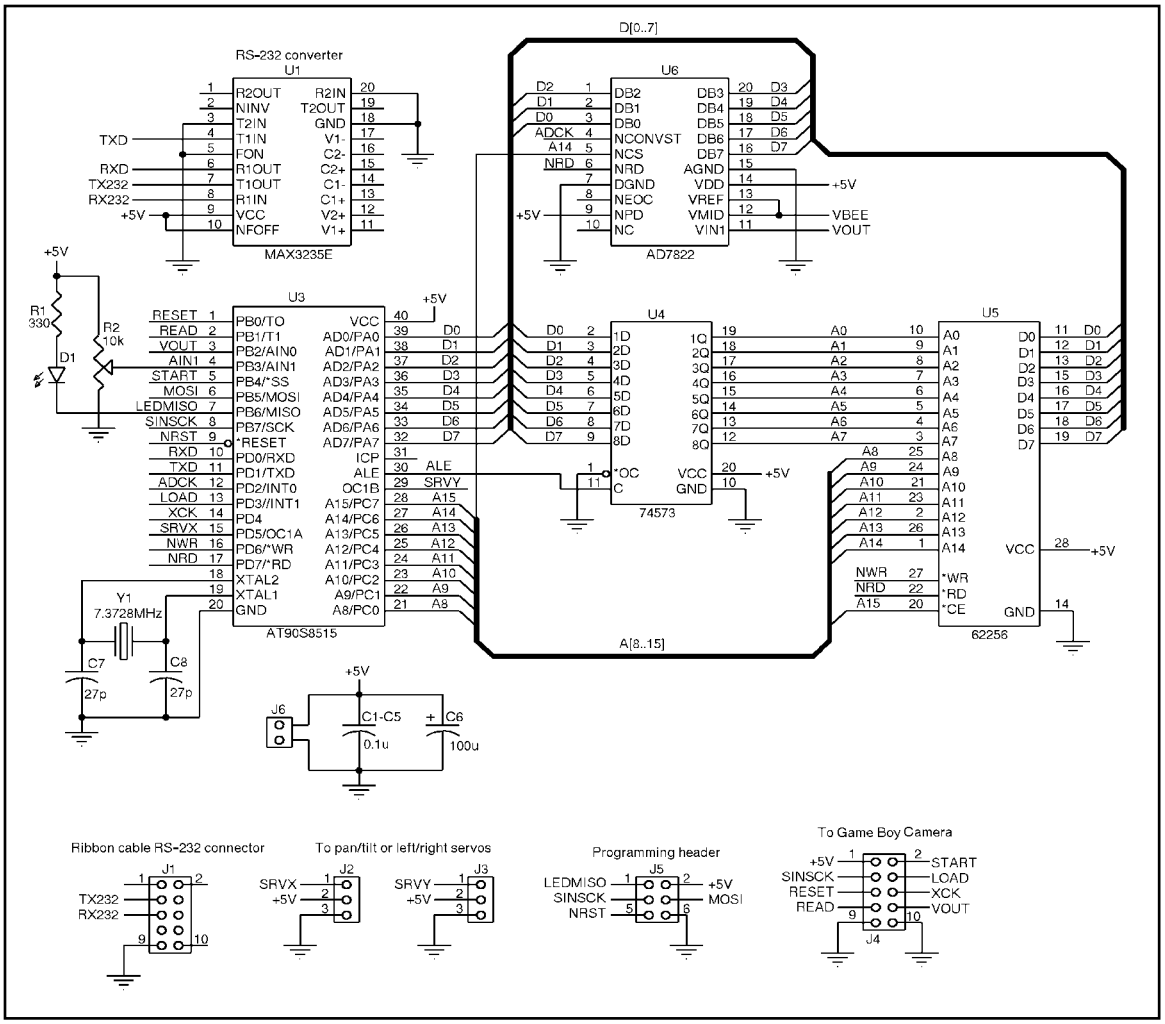


Figure 2: With this processor and interface circuitry and connector design, you can enhance your own robotics applications with the Game Boy Camera.

the test objects in Photo 4 is shown in the Depth/Nearest/Track frame in Photo 5c. The groups of shaded columns are areas that include objects.

Point and Shoot

The opposite of obstacle avoidance is object tracking. The camera can be panned and tilted in response to an object found in the image. Assuming the lowest edge in the image above some brightness threshold is an object to be tracked, the microcontroller can command servo motors to pan and tilt the camera to move the object to the center of the image. This requires some intelligent control of the motors to prevent a slow response, overshooting, and ringing.

To perform object tracking, the microcontroller searches the image in RAM from the bottom up. When it finds the first edge brighter than a given threshold value, it marks the x and y locations and measures the horizontal and vertical distance of this edge from the center of the image. Then, the microcontroller issues a corrective movement command to the servos, which respond by redirecting the camera until the object is centered in the view. Listing 2 shows how it's done.

Photos 6a and b show the pan-tilt servo mechanism. The pan servo is directly mounted to the tilt servo's control surface. Note that the sub-micro-sized servos in my photos allow for a compact installation.

The performance of the pan-tilt camera head is adequate for tracking small objects, provided that the object isn't moving faster than about 1 foot per second at a distance of 4 feet from the camera. This means you can roll a ping-pong ball at a moderate speed across the floor roughly 4 feet in front of the camera, and the camera will lock on and track the ball until the servos reach their limit.

The system won't notice a bouncing ball. Using a large ball (e.g., a basketball) causes different edges (left and right) of the ball to be detected, and the camera oscillates between the two nearest edges if they alternate in intensity or y position.

One helpful piece of equipment for tuning the system is a laser pointer. With a laser pointer, a bright point can

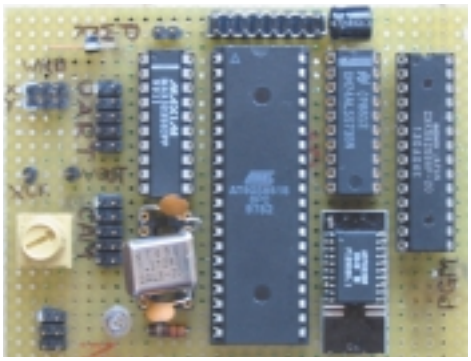


Photo 1: The prototype circuit board is small enough to fit inside a mobile hobby robot.

Photo 2: I've provided you with four views of the mobile robot. The servo that controls the tilt angle of the camera is for future expansion.

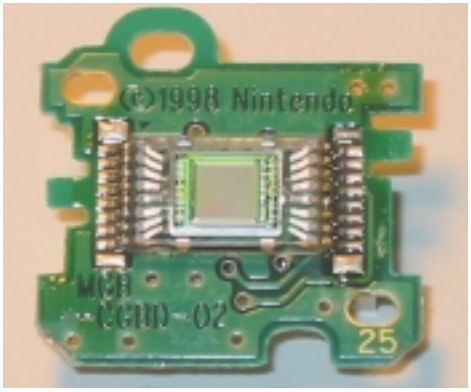


Photo 3: There's only one chip and two capacitors on the circuit board in the camera ball. Take a look at the clear-packaged M64282FP IC.

be moved from one location to another almost instantaneously. Using one, you can observe the reaction of the servos.

The gain of the system is set too high if the servos overshoot and "ring" (i.e., oscillate until they come to a rest) at the new location. The gain should be set by increasing or decreasing the divisor of the error amount so that the correction amount causes the servos to overshoot slightly without ringing. Look for the constant, TRACKDIV, in the assembly code for more information.

Incidentally, the entire image-capture/ process/output sequence takes roughly 11 ms, yielding a frame rate of about nine frames per second. The pan-tilt camera head is only able to track objects while they are within the servo's travel limits. If a subject is lost because it moved too far to the left or right, the camera will wait for up to 30 frames for the object to reappear before it returns to the center position to look for another object. You can overcome this limitation by giving the camera the ability to pan all the way around. To do this, mount it on a mobile robot.

Photo 4: The perfect testing area may be closer than you think. I placed the test objects on my kitchen floor, which has a practically constant texture and surface.

Follow that Subject!

You can apply the theory used for panning and tilting a camera to controlling a mobile robot. The camera itself is stationary with respect to the robot base. Instead of controlling the camera directly, the microcontroller commands the robot base to move in a certain direction at a specified speed. This arrangement allows the mobile robot to find high-contrast objects and approach them.

The robot is able to search for objects by spiraling out in an ever-widening arc until an object is within view. When an object is detected, the robot faces the object and speeds toward it. The robot slows down gradually until it stops with the object located in the center of the camera image. As long as the object doesn't move too fast, the robot will continue to rotate, move forward, or move backward to keep the object in the center of the image. Photo 2 shows the prototype of the mobile robot.

Instead of using the external ADC, the microcontroller uses the on-chip analog comparator to detect bright pixels. In addition, the RAM isn't used, because the only information the robotrequires is the nearest object location. To determine the location of the nearest object, the pixels om the camera are read and processed on the fly.

Because the information isn't sent from the UART, it's pointless to have the RS-232 converter on the board. Therefore, you can construct a reduced circuit for the mobile robot. The only components that you need for the mobile robot's microcontroller board are the microcontroller itself, the crystal, a potentiometer (which is used for adjusting the analog comparator reference voltage), and a few capacitors.

Regarding the performance of the mobile robot, the camera does an excellent job sensing high-contrast objects within its view; however, it is inadequate for detecting the majority of medium- and low-contrast obstacles. In the real world, you should always use multiple layers of sensors. It is a good idea to try supplementing the camera with a bumper panel or whiskers.

Developing

The software for this project consists of two parts. The first section consists of the assembly code in the AVR microcontroller that talks to the camera, RAM, and serial interface. The second part includes the C program for a Linux-based PC that reads and writes camera registers. In addition, this portion captures images and obtains depth information, nearest object information, or object-tracking locations.

I assembled the microcontroller code with the tavrasm AVR assembler and programmed the microcontroller with sp12. I wrote the C program for the host PC using the Simple DirectMedia Layer (SDL) library, which is a public cross-platform graphics and multimedia-programming library.

SDL includes routines for drawing pixels on the X-windows display. A user-contributed extension, Bfont, supplies the routines for writing text to the window. Refer

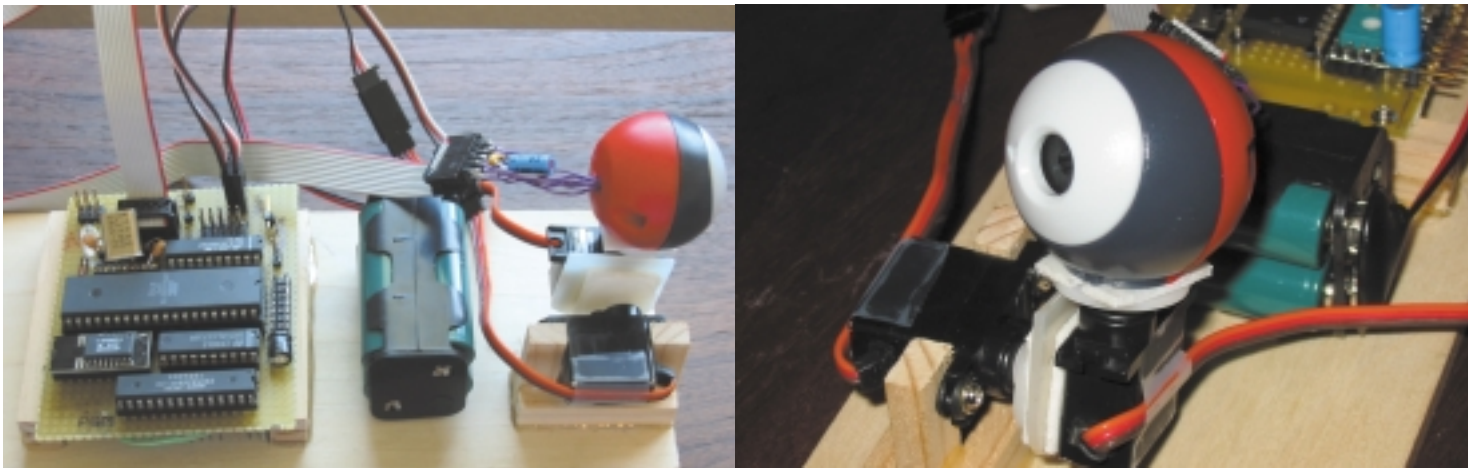


Photo 6a: Four AA NiCd or NiMH batteries power the circuit board used for the pan-tilt head. b—These servos have a quick transit time. It takes only 0.09 s to rotate the control surface through 60°! This fast response time keeps the servos from being the limiting factor in the system's reaction time.

Photo 5a

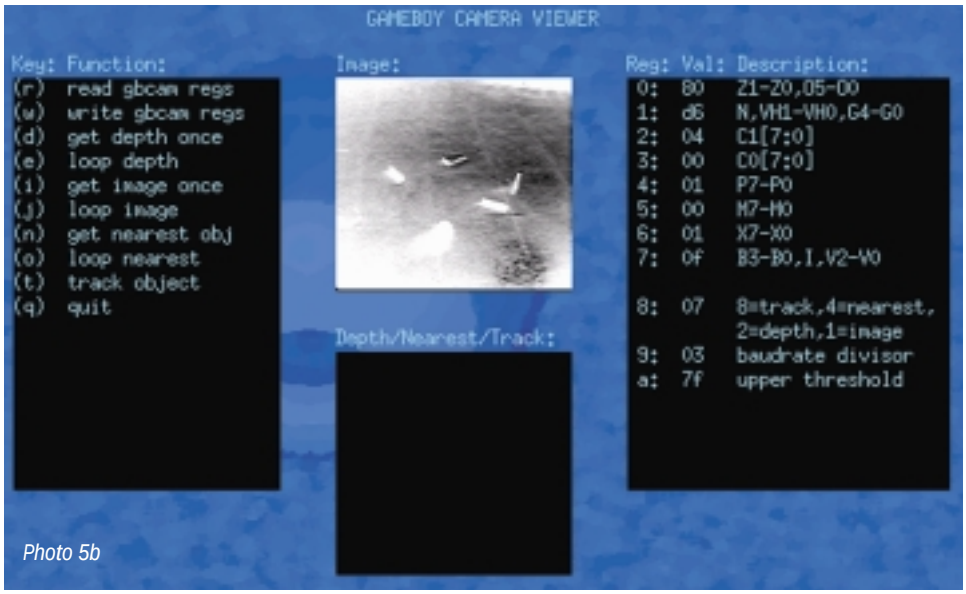


Photo 5b

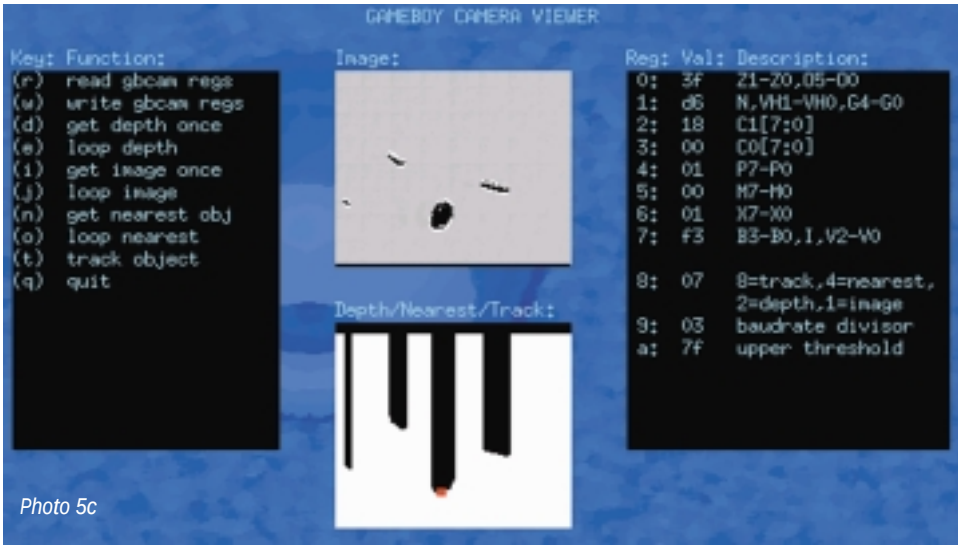


Photo 5c

to Photo 5 for screen shots of the user interface. Although I completed all of the development for this project in Linux, it should be fairly easy to port to a Windows environment.

There are two versions of the AVR assembly code. The pan-tilt program controls the pan-tilt camera head, and the chase program controls the mobile robot. The pan-tilt code has the added overhead of communicating with the PC through the UART. In addition, it requires the use of the pan-tilt C program on the PC. This adds significant delays to the operation of the pan-tilt camera.

If you want to get the maximum performance out of the pan-tilt camera, you can hard-code the register settings and remove the calls to the UART communication routines. The chase code is already optimized for the fastest possible frame rate. Either of the AVR programs can be modified to allow for higher-level behavior by adding calls to new routines in the main loops.

Enlargements

One way to enhance the output of the tracking system would be to mount a low-power laser pointer on the pan-tilt head. Then, as long as the field of view is kept away from humans and highly reflective surfaces (in addition to other appropriate safety precautions), the robot could alternate between strobing the laser pointer and tracking an object. This would let you see a pulsating red dot, signaling where the robot is actually focusing its attention.

You could also use the laser and a cylindrical lens to generate a flat, fanned-out beam (visible as a horizontal line on any objects in its path). This line would be visible only at points in the field of view where an object is obstructing the beam (T. Ward, "Scope Out the Laser Range Finder," Circuit Cellar 46). Therefore, the camera would have a much greater ability to detect low-contrast objects against the floor background. Additionally, this approach would also help in finding low-contrast walls.

If a high-speed parallel connection to a PC were used instead of the serial one, then the PC could attempt to perform pattern recognition in real time, comparing the edge-extracted image to an edge database of known objects.

A similar improvement would be to use two cameras in tandem without pan servos. A microcontroller could send both edge-extracted images via a high-speed parallel connection to a PC, allowing the PC to compare the two images and attempt to find matching patterns of edges. Assuming that the pan angles of the two cameras were fixed, the matching pattern locations would then allow the PC to determine the distance to certain objects based on the distance between the two cameras and the difference in pan angles.

Atmel Raises the Bar on Tire Pressure Monitoring Systems

By: Markus Levy, Convergence Promotions

How many people check their tire pressures on a regular basis? If you walked outside and checked the air pressures in your vehicle's tires, you'll probably find that at least one is under-inflated. That's what the U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) discovered in a recent study. Tires under-inflated by as little as 3psi can adversely affect vehicle handling, as recent Goodyear and NHTSA tests show, when front tire pressure's are low, a vehicle wants to go straight, thus requiring more steering wheel input to make the curve. Furthermore, under-inflated tires change the shape of the tire's footprint and the pressure it exerts on the road, degrading the tire's ability to transmit braking force to the road surface. With one or more tires on the majority of vehicles being nearly 8psi below the recommended inflation pressure, the chances of hydroplaning and overheating also come into play, not to mention the drop in fuel economy and the excessive wear on tire edges.

The NHTSA has proposed a new safety standard to warn the driver when a tire is significantly under-inflated (the current industry controversy is in the definition of 'significantly under-inflated' and that the standard will not provide a timely warning). The proposal requires manufacturers to install a four-tire Tire Pressure Monitoring System (TPMS) that is capable of detecting when a tire is more than 25 percent under-inflated and warning the driver. Beginning by September 1, 2005, the new Federal Motor Vehicle Safety Standard would apply to passenger cars, trucks, multipurpose passenger vehicles, and buses with a gross vehicle weight rating of 10,000 pounds or less, except those vehicles with dual wheels on an axle. Atmel is at the forefront of this application with its offering of all the components needed to design a direct measuring sensor TPM system (TPMS sensor modules and car module base stations).

Indirectly Measuring Tire Pressure

In general, today's TPM systems allow just a general prewarning ("something is wrong") without precise conclusions about what happened and where. Future TPM systems will have monitoring dedicated to both driving and parked situations, depending on different temperature and loading scenarios.

TPM is a vehicle-embedded system detecting the tire pressure by analyzing different wheel speeds or by measurement of pressure and temperature. Fundamentally, the TPMS uses a transmitter with a tire pressure and/or heat sensor located inside the each tire's air valve stem or inside the wheel itself.

There are two primary forms of TPMS: indirect and direct. The indirect method is based on technology that is already more than a decade old. This method is very low cost, but is unacceptable due to its many disadvantages. Specifically, indirect TPMS doesn't measure actual tire pressure, instead these systems monitor the rotational speed of the individual wheels and interpolate that data to warn a driver of tire pressures below the acceptable margin. Basically the way an indirect TPMS works is when a tire loses air pressure its effective diameter gets smaller and the rotational speed increases in relation to the other tires. The vehicle's on-board computer and/or a chip in the indirect TPMS device(s) picks up on the differences in wheel speeds and signals the driver there's a problem. Some indirect systems tie in directly to the vehicle's anti-lock braking system's (ABS) wheel-speed-sensors to help reduce implementation costs. Reading the same signal used to modulate ABS systems, the vehicle manufacturers have programmed another function into the vehicle's onboard computer to warn the driver when a single tire is running at a reduced inflation pressure compared to the other three.

Despite the cost benefits of implementing an indirect tire pressure monitoring system, there are several shortcomings to this approach. First, indirect systems don't tell the driver which tires are low on pressure and won't warn the driver if all four tires are losing pressure at the same rate (as occurs during the fall and winter months when temperatures turn colder). Indirect systems can also generate frequent false warnings, such as when the vehicle is driven around a long curve that causes the outside tires to rotate faster then the inside tires, or when the tires spin on ice and snow-covered roads.

Rolling With Direct TPM Systems

Direct tire pressure monitoring systems measure, identify and warn the driver of low pressure. Direct systems have a sensor in each wheel and can generate more accurate warnings if the pressure in any one tire falls below a predetermined level due to rapid or gradual air loss. Many direct systems use dashboard displays that provide the ability to check current tire pressures from the driver's seat.

The least expensive, and most basic of the direct TPMS systems, are nothing more than a valve stem cap that changes color when the tire pressure drops below a preset pressure to warn the driver a tires needs airing-up. Alternatively, a pressure sensor with a transmitter can be banded to the inside of the wheel before the tire is mounted, or via a pressure monitoring device that attaches to or replaces the conventional valve-stem or valve stem cap. This more sophisticated and accurate method of direct tire pressure monitoring is based on a UHF receiver in the vehicle and 4 sensor modules mounted on the wheel rim/valve to sense data, to

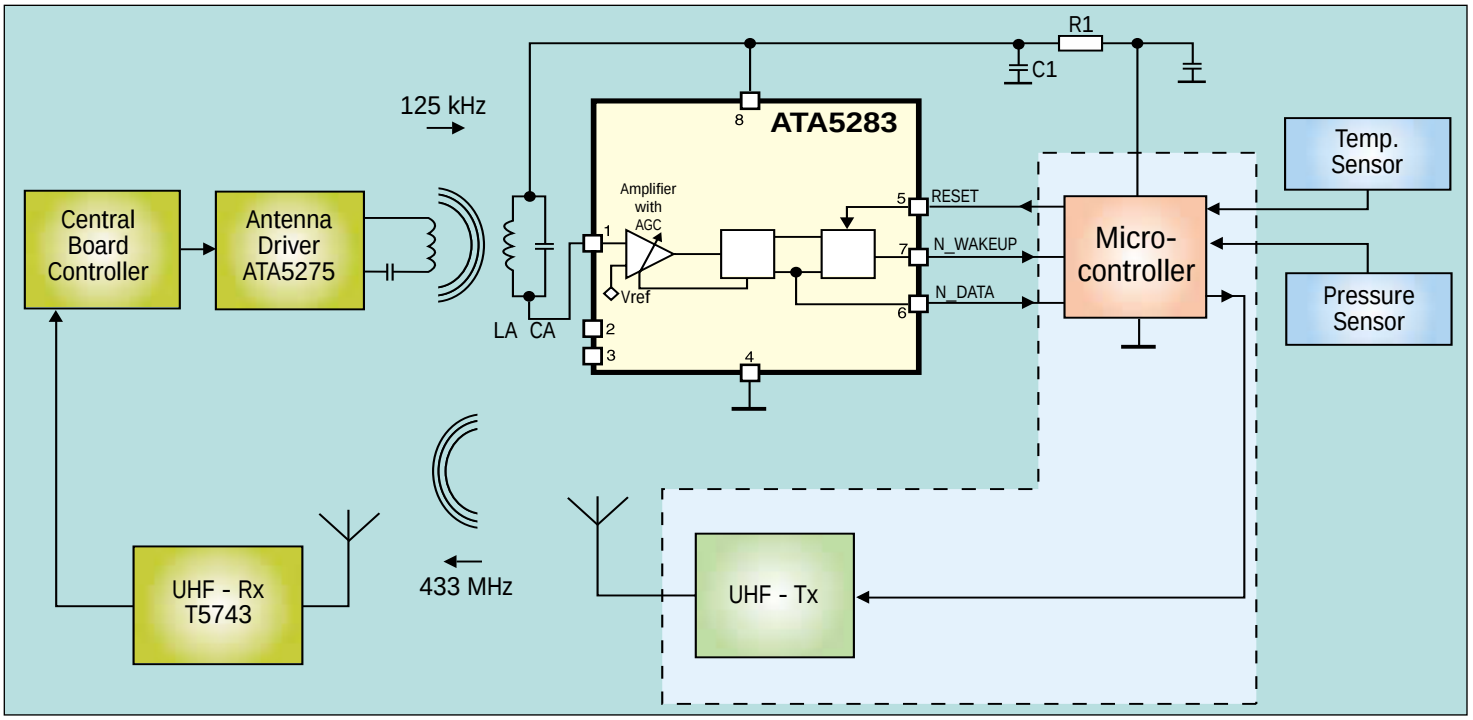


Figure 1: Atmel's TPMS product portfolio includes UHF transmitter and receiver ICs as well as highly integrated microcontrollers

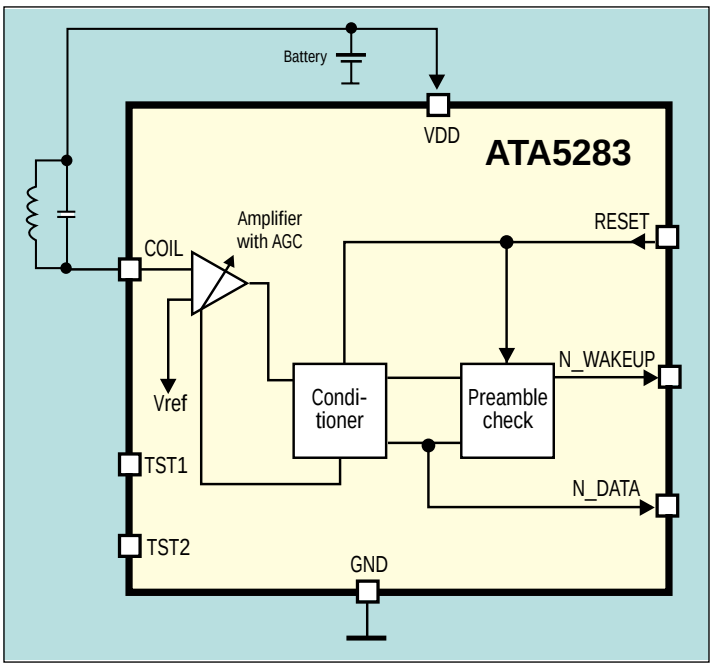
calibrate pressure vs. temperature, and to organize the data transmission to the car body and driver display. Atmel's TPMS product portfolio includes UHF transmitter and receiver ICs as well as highly integrated microcontrollers (Figure 1). All of these products operate in the -40 to 125 degree C to comply with automotive standards.

The direct TPMS approach can be implemented using Atmel's integrated ATAR862 (a 4-bit microcontroller combined with a UHF ASK/FSK transmitter) and ATA5283 (interface IC for 125 kHz wake-up function). The ATA5283 (Figure 2) can wake up sensor modules in defined duty cycles to improve the TPMS' battery lifetime. In addition to its tolerance for a wide range of temperatures, to comply with automotive manufacturers' standards, a TPM module must have a minimum lifetime of 10 years. Since the integrated battery in such modules cannot be replaced, low current consumption is a must for TPM applications. The LF receiver ATA5283 is constantly in standby listening mode and monitors for a signal originating from the car or from a base station. The device switches off the TPM module when it is not needed and wakes up the module only if a valid preamble has been detected. The ATA5283's current consumption in listening mode with a sensitivity of 1 mV rms is as low as 1 microampere (typical), while competing products have current consumption at least 3 times higher.

Another key requirement in TPM applications is small outline and low weight. Since the high speed of a car wheel causes enormous centrifugal forces, a TPM module needs to be very small and may not exceed a weight of 35 grams. Up to now, the smallest possible 125 degree C TPM products were only available in TSSO16 packages, the ATA5283 is available in a TSSOP 8-lead package.

For correct signal demodulation, an appropriate internal signal amplitude is needed. To control the input signal, the ATA5283 has a built-in digital automatic gain control (AGC) to amplify the input signal from the coil. The AGC regulates the internal signal amplitude according to the reference value. With its dynamic range of 60 dB, the AGC can handle both very large and very small input signals from 1 mV rms to 1000 mV rms. To prevent the circuit from unintended operation in noisy environments, a valid preamble signal with a minimum of 192 periods is necessary to activate the AGC circuit. It requires up to 512 carrier periods for the settling. Short interrupts, which are suppressed by the signal conditioner, are tolerated. The ATA5283 supports ASK modulation (PWM, Manchester and Bi-Phase) with a data baud rate up to 4 kbit/s.

The technology available from Atmel today, along with the direct TPMS method will be implemented in automobiles for the next 3-5 years. The next phase is battery-less systems where no battery is used in the sensor module, instead, a sensor tag is placed inside the tire to allow the merging of RFID and pressure measurement functions. Regardless of the technology, we recommend that your tire pressures are checked monthly because even good tires lose about one-pound of air per month through natural seepage.



Code Patch

Setting the UNIX/Linux Environment to Develop an AVR Application

By: Daniel Widyanto, ITB DSP Lab

Features

- Setting GNU Toolchain
- Setting Debugger and Simulator

Introduction

This application note will guide the UNIX/Linux user in building an AVR development environment.

Setting GNU Toolchain

The GNU toolchain consists of GCC (GNU Collection Compiler), binutils, and library. These packages are usually installed in your Linux box, but usually it is configured only to build host architecture software.

To make it compile AVR binaries, one must download and reconfigure these toolchains.

Package Name	Description	Download Location
Binutils-2.14 `binutils-2.14.tar.bz2`	Binary utilities	http://freshmeat.net/projects/binutils/
GCC-3.3.1 `gcc-3.3.1.tar.bz2` **OR** `gcc-core-3.3.1.tar.bz2` `gcc-g++-3.3.1.tar.bz2`	C compiler	http://gcc.gnu.org/mirrors.html
AVR-libc-1.0.4 `avr-libc-1.0.4.tar.bz2`	AVR C libraries	http://savannah.nongnu.org/download/avrlibc/
AVR libc - User Manual `avr-libc-user-manual-1.0.4.tar.bz2`	AVR C libraries user manual	http://savannah.nongnu.org/download/avrlibc/

Table 1. GNU Toolchain

The GNU Toolchain is a combination of GCC, binutils, and avr-libc versions. Not all of these combinations work. Please take a look at http://winavr.sourceforge.net/package.html for a working combination.

To start the installation, make a separate directory first. A separate directory makes it easy to remove.

```
$ su
$ mkdir /usr/local/atmel
```

Binutils-2.14

Extract binutils, configure, and install using UNIX/Linux default installation. Option --prefix is used to arrange its output directory.

```
$ cd /temprorary/working/directory
$ tar -xvjf binutils-2.14.tar.bz2
$ cd binutils-2.14
$ ./configure --target=avr --prefix=/usr/
    local/atmel \
   --disable-nls
$ make
$ make install (as root)
```

GCC-3.3.1

Only GCC above v.2.95 is available for AVR, except GCC-3.3.3, which is not work for AVR. GCC-3.3.1 is the only stable version of GCC when this application note was made. C++ is still in development in this version.

```
$ cd /temprorary/working/directory
$ tar -xvjf gcc-core-3.3.1.tar.bz2
$ tar -xvjf gcc-g++-3.3.1.tar.bz2
```

OR

```
$ tar -xvjf gcc-3.3.1.tar.bz2
$ mkdir gcc-build
$ cd gcc-build
$ ../gcc-3.3.1/configure --target=avr \
  --prefix=/usr/local/atmel \
  --enable-language=c,c++ --disable-nls
$ make
$ make install (as root)
```

AVR-libc-1.0.4

AVR libc has to be compiled within a new directory outside its source code directory. Otherwise, some of the AVR type library won't be compiled.

```
$ cd /temprorary/working/directory
$ tar -xvjf avr-libc-1.0.4.tar.bz2
$ mkdir/temprorary/working/directory/avr-libc
$ cd avr-libc
$ export PATH=/usr/local/atmel/bin:$PATH
$ ../avr-libc-1.0.4/configure --
    prefix=/usr/local/atmel
$ make
$ PATH=/usr/local/atmel/bin:$PATH
    make install (as root)
```

AVR-libc-1.0.4 User Manual

AVR-libc usually come with built-in documentation. You need to install Doxygen to build these documents. This

package contains only AVR-libc documentation in HTML format, just in case you don't have Doxygen to build the AVR-libc documents. Extract it to any place that you like.

```
$ tar -xvjf avr-libc-user-manual-1.0.4.tar.bz2
    -C /usr/local/atmel (as root)
```

Setting Environment

After these toolchains are installed, you need to add the path in your environment shell. Assume you're using BASH, add this in your `.bash_profile` file

```
PATH=$PATH:/usr/local/avr/bin
export PATH
```

Setting Debugger and Simulator

Debugger and simulator is useful to verify one's design. There are many free AVR debuggers and simulators around. This application note uses gdb and SimulAVR.

Package Name	Description	Download Location
GDB-6.1 gdb-6.1.tar.gz	GNU Debugger	http://ftp.gnu.org/gnu/gdb ftp://sources.redhat.com/pub/gdb/releases/
SimulAVR-0.1.2.1 simulavr-0.1.2.1.tar.bz2	AVR Simulator	http://savannah.nongnu.org/download/simulavr/

GDB-6.1 GDB

(GNU Debugger) is GNU's official debugger. GDB must be combined with SimulAVR to simulate AVR.

```
$ cd /temprorary/working/directory
$ tar -xvzf gdb-6.1.tar.gz
$ cd gdb-6.1
$ ./configure --target=avr --
    prefix=/usr/local/atmel
$ make
$ make install (as root)
```

SimulAVR-0.1.2.1

SimulAVR provides an AVR emulator in your computer. The AVR registers and data as provided by SimulAVR. You can read these registers and data using GDB.

```
$ cd /working/directory
$ tar -xvjf simulavr-0.1.2.1.tar.bz2
$ cd simulavr-0.1.2.1
$ ./configure --prefix=/usr/local/atmel
$ make
$ PATH=/usr/local/atmel/bin:$PATH
    make install (as root)
```

Code Patch

Using STK500 Under UNIX/Linux Environment

By: Daniel Widyanto, ITB DSP Lab

Introduction

This application note is a guide to use the STK500 features under a UNIX/Linux environment. To be able to use STK500 in Linux, one needs additional software. There are two software packages that can be used with STK500: AVRdude (formerly known as AVRprog) and UISP. It is recommended that you use UISP, since it can handle all of STK500 features.

AVRdude-4.3.0

Installation

AVRdude can be downloaded from http://savannah.nongnu.org/download/avrdude. To install it, extract the source first.

```
$ tar -xvjf avrdude-4.3.0.tar.bz2
```

Then, configure, compile, and install.

```
$ cd avrdude-4.3.0
$ ./configure
$ make
$ su
$ make install
```

Programming Devices on STK500

The typical command to use STK500 with avrdude is :

```
$ avrdude -c stk500 -p <device> \
  -U <memtype>:r|w|v:<filename>[:format]
     [other_option]
```

Option explanation :

-c : Use to identify the programmer that is used. Currently, AVRdude supports STK200, STK500, avrisp, AVR910, butterfly.

-p : AVR device. Currently these AVR <device> are supported :

t15	AT Tiny15	1200	AT90S1200
2313	AT90S2313	2323	AT90S2323
2333	AT90S2333	2343	AT90S2343
4414	AT90S4414	4433	AT90S4433
4434	AT90S4434	8515	AT90S8515
8535	AT90S8535	m163	ATMEGA163
m169	ATMEGA169	m128	ATMEGA128
m103	ATMEGA103	m16	ATMEGA16
m8	ATMEGA8		

-U : Memory operation. Multiple -U can be used to operate on multiple memories at single command.

<memtype> : memory type (flash or eeprom).

r | w | v : (r)ead memory to <filename>, (w)rite <filename> to memory, (v)erify memory against <filename>

[:format] : <filename> format. The option is:

- i - Intel-hex format
- s - Motorola S-record
- r - Raw binary, little endian byte order
- a - Auto
- m - Immediate, the value is entered directly

Reading Flash/EEPROM

To read flash from a device on STK500, use the following command (assume you are using AT90S8515):

```
$ avrdude -c stk500 -p 8515 \
    -U flash:r:8515_flash.hex:i
```

The AT90S8515 flash memory will be saved in 8515_flash.hex. This file is in Intel-hex format. To read EEPROM, change flash to eeprom.

Writing Flash/EEPROM

To write flash to the device on STK500, use the following command (assume you are using AT90S8515):

```
$ avrdude -c stk500 -p 8515 \
    -U flash:w:test.hex:i
```

AVRdude will automatically erase the device before it is programmed and verify the result. The test.hex file is in Intel-hex format. To write the EEPROM, change flash to eeprom.

Signature, Fuses, and LockBits

You must read the datasheet : fuse programming, lockbit programming, and signature reading section to understand the magic number that is used to program the signature, fuses, and lockbits.

NOTE: Not all AVR devices support fuses/lockbits read/write in ISP programming mode.

Signature, fuses, and lockbits are located in special memory section (depend on AVR type). To see, where they are located, type (assume you are using ATMega16):

```
$ avrdude -c stk500 -p m16 -v
```

You will see something like this :

(see table top of next page)

Memory Type	Paged	Page Size	Size	#Pages	MinW	MaxW	Polled ReadBack
eeprom	no	512	0	0	9000	9000	0xff 0xff
flash	yes	16384	128	128	4500	4500	0xff 0xff
lock	no	1	0	0	0	0	0x00 0x00
lfuse	no	1	0	0	0	0	0x00 0x00
hfuse	no	1	0	0	0	0	0x00 0x00
signature	no	3	0	0	0	0	0x00 0x00

It means that the signature is located in signature memory, fuse in lfuse and hfuse memory, and lock in lock memory. It maybe different on different AVR devices. Usually, this name corresponds to the datasheet (eg. lfuse to describe *low-byte fuse* in ATMega16).

Programming Fuses/LockBits
To program lockbits/fuses, use lock/lfuse/hfuse as memory type, replacing flash/eeprom keyword in -U option. For example, to activate mode 2 BLB1 (Boot Lock Bit) in ATMega16, type (refer to AT Mega16 datasheet page 259)

```
$ avrdude -c stk500 -p m16 -U lock:w:0x2F:m
```

Notes: with m-modes, all values are located in filename section and are entered directly in the lockbit memory.

UISP
UISP is far more advanced than AVRdude. UISP can be used to set advanced features in STK500 (setting STK500 oscillator, VTARGET, AREF). It also can be used to program STK500 in high-voltage mode.

Installation
UISP can be downloaded from http://savannah.nongnu.org/download/uisp/. UISP only needs typical installation (extract, configure, compile, and install).

```
$ tar -xvjf uisp-20040311.tar.bz2
$ cd uisp-20040311
$ ./configure
$ make
$ su
$ make install
```

Programming STK500
Typical command to program STK500 is:

```
$ uisp -dserial=/dev/ttyS0 -dprog=stk500 \
  -dpart=ATxxx --[upload|download|verify] \
  --segment=flash|eeprom|fuse \
  if=<input_hex_file>
      [of=<output_hex_file>]
```

Option explanation:

-dserial :location where your STK500 is attached. / dev/ttyS0 for serial port 1, serial port 2 is / dev/ttyS1, and so on
-dprog :selecting STK500 board
-dpart :device name. Use complete device name (eg, -dpart=atmega16)

Reading Flash/EEPROM
To read flash, use (assume you are using AT90S8515):

```
$ uisp -dserial=/dev/ttyS0 -dprog=stk500 \
    -dpart=AT90S8515 --download --segment=
      flash \ of=8515_flash.hex
```

The flash will be saved in 8515_flash.hex. This is Motorola S-record format file. To read eeprom, replace --segment=flash with -- segment=eeprom

Writing Flash/EEPROM
To write flash, use (assume you are using AT90S8515):

```
$ uisp -dserial=/dev/ttyS0 -dprog=stk500 \
  -dpart=AT90S8515 --upload --segment=flash \
  if=test.hex
```

UISP can accept Intel 16-bit hex file or Motorola S-records (S1 and S2). The file type is automatically detected by UISP. It also erase the chip automatically.

Programming Fuses/LockBits
Under UISP, fuses and lockbits programming is far more easy than AVRdude.

NOTE: Not all AVR device support fuses/lockbits read/write in ISP programming mode.

There are several additional option to program fuses and lockbits:

--rd_fuses : Read all fuses and print values to screen / stdout
--wr_fuse_l=byte : Write fuse low byte
--wr_fuse_h=byte : Write fuse high byte
--wr_fuse_e=byte : Write fuse extended byte
--wr_lock=byte : Write lock bits. Argument is a byte where each bit is:
Bit5 -> BLB12
Bit4 -> BLB11
Bit3 -> BLB02
Bit2 -> BLB01
Bit1 -> LB2
Bit0 -> LB1

Here's an example of how to program mode 2 BLB1 (Boot Lock Bit) in ATMega16 (refer to AT Mega16 datasheet page 259):

```
$ uisp -dserial=/dev/ttyS0 -dprog=stk500 \
    -dpart=atmega16 --wr_lock=0x2F
```

STK500 Additional Features
STK500 has many additional features, such as high-voltage programming, on-board oscillator, VTARGET, AREF software controllable.

High Voltage Programming
Refer to page 3-13 of the STK500 User Manual for high-voltage programming setup.

Add option -dhiv to uisp, eg:

```
$ uisp -dserial=/dev/ttyS0 -dprog=stk500 \
    -dpart=AT90S8515 -dhiv --upload \
    --segment=flash if=test.hex
```

AREF setting
Refer to page 3-17 of the STK500 User Manual for AREF jumper setting.

Reading AREF
Use option: --rd_aref

NOTE: Due to a bug in the STK500 firmware, the read value is sometimes off by 0.1 from the actual value measured with a volt meter.

Writing AREF
Use option: --wr_aref=<value>

Valid values are 0.0 to 6.0 volts in 0.1 volt increments. Value can not be larger than the VTARGET value.

VTARGET setting
Refer to page 3-16 of the STK500 User Manual for AREF jumper setting.

Reading VTARGET
Use option: --rd_vtg

NOTE: Due to a bug in the STK500 firmware, the read value is sometimes off by 0.1 from the actual value measured with a volt meter.

Writing VTARGET
Use option: --wr_vtg=<value>

Valid values are 0.0 to 6.0 volts in 0.1 volt increments. Value can not be smaller than the AREF value.

Oscillator setting
Refer to page 3-20 STK500 User Manual for AREF jumper setting.

Reading oscillator
Use option: --rd_osc
The frequency is displayed in Hertz

Writing oscillator
Use option: --wr_osc=<value>
Set the oscillator frequency in Hertz, from 14.06 to 3686400.

CryptoMemory®

The World's ONLY Secure Serial EEPROM

With CryptoMemory you get a low cost, high security solution for any embedded application that requires data protection. These devices have many security features not found in any other product on the market today.

Atmel's CryptoMemory family was developed after years of designing chips for the smart card market, and those same security techniques are now available for the embedded market.

CryptoMemory is available from 1 Kbit to 256 Kbits, operating in a synchronous 2-wire protocol. A proprietary cryptographic algorithm encrypts data, passwords and checksums, providing a secure place for storage of sensitive information within a system. With its tamper-protection circuits, this information remains safe even under attack.

CryptoMemory is available now…only from Atmel.

Check out www.atmel.com/CryptoSecureMem/ for more information on CryptoMemory

Features	Applications
Four Independent Levels of Security	Authenticates Subassemblies
Symmetrical Dynamic Mutual Authentication	Securely Stores System Configuration Data
Stream Encryption	Protects Networked Systems

Everywhere You Are℠

Finding the building blocks for your next embedded design isn't child's play.

That's why for applications from Automobiles to Zigbee protocols, companies that drive global innovation choose Atmel.

Atmel is a global leader in researching, designing, manufacturing and marketing advanced semiconductors, including microcontroller, nonvolatile memory, logic, secure, mixed analog/digital,radio frequency and sensor integrated circuits (ICs).

These functions are marketed as standard products (aimed at a wide range of applications for many customers), ASSPs (a single application for a limited number of customers) and ASICs (implementing a specific application for a single customer).

These functions are marketed as standard products (aimed at a wide range of applications for many customers), ASSPs (a single application for a limited number of customers) and ASICs (implementing a specific application for a single customer).

Atmel ICs are fabricated in its own manufacturing facilities using its proprietary industry-leading process technologies that are fine-tuned to the requirements of its products and customers.

This gives Atmel's system designers' flexibility of choice in terms of matching device performance to their product requirements, time-to-market, development cost and unit price in volume.

Start your journey today towards a successful design at: www.atmel.com

Everywhere You Are™

LIT #3362D-CORP-VOL-4-15M