In Proceedings of the second international conference on Embedded networked sensor systems (SenSys 2004), Baltimore, Maryland, USA, November 2004.

Poster Abstract: Flexible Hardware Abstraction of the TI MSP430 Microcontroller in TinyOS*

Vlado Handziski† handzisk@tkn.tu-berlin.de
Joseph Polastre‡ polastre@cs.berkeley.edu
Jan-Hinrich Hauer† hauer@tkn.tu-berlin.de
Cory Sharp‡ cssharp@eecs.berkeley.edu

† Technische Universität Berlin
Telecommunication Networks Group
Sekr. FT 5, 10587 Berlin GERMANY

‡ University of California, Berkeley
Computer Science Department
Berkeley, CA 94720 USA

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and embedded systems*

General Terms

Design, measurement, experimentation

Keywords

Wireless sensor networks, TinyOS, TI MSP430

1. INTRODUCTION

Wireless sensor networks (WSNs) promote energy-efficiency as the main design criterion. This introduces rather conflicting requirements for the hardware adaptation layer. Maximizing the efficiency requires that the presentation must closely mimic the underlying hardware model. On the other hand, increasing the level of abstraction simplifies the development, but at the cost of *lowered* efficiency because it obstructs the link between the application and the hardware. As new microcontrollers and radios are introduced for use in WSNs, applications must be able to effectively use new low power features and peripherals.

The MSP430 family of microcontrollers by Texas Instruments is specifically designed for ultra-low-power applications. It incorporates a 16-Bit RISC CPU, peripherals and a clock system. The MSP430F149 is one of the most popular members of the family. As shown in Fig. 1, it has 60 KB Flash, 2 KB of RAM and a flexible clock system sourced by an internal digitally controlled oscillator (DCO) and/or two external oscillators. It also contains a 12-Bit A/D Converter, two independent timers and two USARTs.

2. DESIGN APPROACH

TinyOS and nesC [2] enable the application developer to adaptively choose the "right" level of abstraction by "wiring" only selected subparts of the hardware adaptation functionality.

Yet the success of the model squarely depends on the way that the functionality is divided between the components. TinyOS has no rules or restrictions on how applications interface with the physical hardware. We had the freedom to reinvent and reorganize the hardware presentation and abstraction layers for the MSP430. Since TinyOS was initially written for the Atmel family of microcontrollers, it did not effectively abstract the capabilities required by sensor network applications, rather the interfaces exposed the capabilities of Atmel specific hardware.

In the design of the hardware adaptation components, we followed a two phase strategy:

- Following the established TinyOS tradition, our Hardware Presentation Layer (HPL) components are stateless and provide direct access to the hardware via the MSP430 registers. The HPL maps each TinyOS function to the hardware registers and interrupts.
- The components above the HPL expose the full capabilities of the underlying hardware and maintain state to perform common operations, similar to MIT's Exokernel work [1]. Components at this layer provide more suitable abstractions for applications and perform arbitration between higher layer services.
- To maintain compatibility with the existing TinyOS hardware APIs, additional wrapper components have been developed that downcast the functionality of the underlying components to the "traditional" TinyOS interfaces.

This approach enabled us to provide the full capabilities of the MSP430 to the applications that need them. At the same time, the compatibility wrappers ensure that existing TinyOS applications can be used without any modifications.

3. IMPLEMENTATION

The implemented MSP430 platform in TinyOS exposes most of the functionality contained in the peripheral modules depicted on Fig. 1. In the following we will provide a short overview of the most important components.

3.1 ADC

The ADC peripheral is represented by three modules: In the HPLADCM module, all capabilities of the hardware are exposed through the ADC registers and flags.

*This work has been partially supported by the EC under the contract IST-2001-34734 (EYES), the National Science Foundation, and the DARPA "NEST" contract F33615-01-C1895.

The MSP430ADCM module provides interfaces to use the full functionality of MSP430's ADC, arbitration between multiple outstanding requests, and maintains a state machine of the ADC's operation. Our MSP430ADC implementation is more expressive then the standard TinyOS interfaces `ADC` and `ADCControl`. For compatibility, we introduced a wrapper component, ADCC, that maps a subset of the MSP430ADCM module to ADC and ADCControl interfaces.

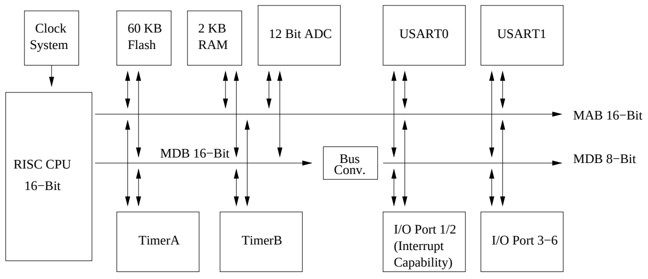

Figure 1: MSP430F149, Functional Block Diagram.

3.2 USART

The MSP430F149 is equipped with two USARTs. Analogous to the ADC there is a HPLUSART for each USART exposing the hardware's capability. Because some applications only need to use either UART- or SPI-mode, there are additional wrappers (HPLSpiM, HPLUARTM) that offer this functionality. The different wrappers access the same hardware through the HPL components. To prevent bus collisions, we introduced arbitration components to TinyOS that guarantee exclusive access to each UART.

3.3 Power Management

On the MSP430 platform we chose a new approach to deal with power management. The standard TinyOS method has an HPLPowerManagementM module actively called by components as they shut down to adjust the power management level. Since the MSP430 wakes up in less than 6 μs, our implementation checks each of the hardware peripherals for use of the internal DCO in the `TOSH_sleep()` function that is invoked by the scheduler whenever the task queue is empty. If the DCO is not in use, the microcontroller enters sleep mode from `TOSH_sleep()`. The advantage of this approach is that it keeps power management transparent, i.e. errors created by components not calling adjustPower() are avoided.

3.4 Timer

Timers are supported through an HPL component that exposes the two MSP430 Timers and their corresponding capture/compare registers. As each capture/compare generates an interrupt, the interrupts are dispatched as *alarms* to higher layer components. Above the HPL MSP430 Timer implementation is a wrapper that uses a single Timer alarm to implement the TinyOS notion of "Timers"–a one-shot or periodic timer that can generate events with millisecond granularity. In addition to the TinyOS TimerC wrapper, we also provide jiffy (30.5 μs) granularity alarms and a dedicated low latency jiffy timer for precise uses, i.e. controlling a radio or performing low jitter high frequency data sampling.

4. SUPPORTED PLATFORMS

4.1 Telos

Telos is the latest mote in a line of devices designed by the University of California, Berkeley. Telos is designed to minimize the power consumed during the three phases of low power operation–sleep, wakeup, and active modes. To achieve the lowest power profile in sleep and wakeup modes, the MSP430 microcontroller was chosen. In active mode, we chose a CC2420 wireless transceiver operating at 250kbps and compatible with the IEEE 802.15.4 standard. By significantly increasing the wireless data rate, Telos can return to sleep quickly after sending or receiving data. A Telos node shares many of its resources; for example, the radio and external flash share the same bus. To support concurrent operation of each device, TinyOS drivers reconfigure the hardware USARTs and use double buffering to efficiently communicate with the devices. Telos also takes advantage of the Timer system. A single timer capture/compare register is used to control the CSMA backoff functions of the radio, but during transmit it is reconfigured to capture a timestamp of the start of frame delimiter from the radio. The design of the MSP430 platform components allow Telos to complete operations more efficiently and use less resources in the process than in previous mote generations.

4.2 EYES

EYES is a European research project on self-organizing and collaborative energy-efficient sensor networks. Two platforms based on the MSP430 have been developed in the framework of the project. The EyesIFX platform, developed by Infineon, uses the TDA5250 radio transceiver that supports ASK/FSK modulation with speeds up to 64 kbps. It also provides a high-precision temperature sensor and a light sensor on-board. Similar to the original Mica mote, the EyesNEDAP platform, developed by NEDAP, uses the RFM TR1001 transceiver that supports OOK/ASK modulation with speeds up to 115.2 kbps. Both prototypes are targeted at extremely low-powered, low-datarate WSN applications. The EyesNEDAP node capitalizes on the fast switching capabilities of the simpler transceiver, while paying the price in high noise and interference sensitivity. The TDA5250 radio of the EyesIFX is much more robust, making it a good choice for more demanding RF environments.

Parameter	Mica2	Telos	EyesIFX
Program Memory (kB)	128	60	60
RAM (kB)	4	2	2
Data Rate (kbps)	38.4	250	64
Minimum Voltage	2.7	1.8	2.1
Active Power (mW)	24	3	3
Sleep Power (μW)	75	6	6
Wakeup Time (μs)	180	6	6
Receive Power (mW)	29	38	31
Transmit Power (mW)	42	35	25

5. REFERENCES

[1] D. R. Engler, M. F. Kaashoek, and J. O'Toole, Jr. Exokernel: an operating system architecture for application-level resource management. In *Proceedings of the ACM SOSP-15*, pages 251–266, 1995.

[2] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003*, pages 1–11, 2003.