Embedding a Microchip PIC18F452 based commercial platform into TinyOS

Hans-Jörg Körber†
hj.koerber@hsu-hh.de

Housam Wattar†
wattar@hsu-hh.de

Gerd Scholl†
gerd.scholl@hsu-hh.de

Wolfgang Heller‡
Wolfgang.Heller@enocean.com

† Helmut-Schmidt-University
Electrical Measurement Engineering
Holstenhofweg 85
22043 Hamburg, Germany
+49 4065412638

‡ EnOcean GmbH
Kolpingring 18a
82041 Oberhaching, Germany
+49 89673468934

1. INTRODUCTION

For embedding radio communication modules into low power devices unconstrained by   expensive or obstructive wiring an efficient network operating system working under stringent energy conditions is vital. TinyOS is specially designed for wireless sensor networks (WSNs) and thus faces many of the design criteria of WSN such as severe memory, minimized code size requirements, fine-grained power management and flexible event handling and task scheduling. As both the interest in the programming environment TinyOS/nesC [1] and its user community are rapidly increasing a steady rise of the number of supported platforms (motes) can be simultaneously observed. The most recent platforms which have been implemented into TinyOS source tree [4] are the Telos and Eyes motes [2], which are both based on a TI MSP430 microcontroller (MCU).

Comparable with the MSP430 Microchip offers with the PIC18F family a bundle of MCUs especially designed for low-power devices. To tap the full potential of PIC-based low-power wireless sensors and actuators already on the market we implemented the PIC18F into TinyOS. However, at present there is no GCC tool chain available, which is required by TinyOS. Although there is an open source project [7] with the goal to make GCC available for the PIC the GCC support has not reached his maturity. Instead an alternative method had to be found to introduce the PIC into TinyOS.

2. EnOcean TRANSCEIVERMODUL

The respective hardware platform embedded into TinyOS is the transceiver module TCM 120 developed by EnOcean [6]. The TCM 120 was originally intended to operate as a wireless modem with the capability of processing and relaying all of the different types of EnOcean radio messages generated by self-powered wireless modules like radio switches and sensors. The TCM 120 is controlled by a Microchip PIC18F452 MCU (32 KB ROM, 1.5 KB RAM) which offers all of the common features like UART, ADC, Timer, SPI, I^2C, etc. The MCU is driven by a 10 MHz crystal. For clock rate adjustment a phase-locked loop (PLL) is employed leading to effective 10 Mega instructions per second (Mips). This enables fast clocking of the transmit data into the radio part so that a radio data rate of 120 kbps is achieved. The radio part which is split into a discrete very low-power transmitter and an Infineon TDA 5200 receiver operates at a center frequency of 868.3 MHz. The modulation type is ASK, the maximum transmission power is 10 dBm.

The performance characteristics of the TCM 120 in comparison to alternative hardware platforms are given in "Table 1". If the energy per bit and mW radio output power is considered this comparison clearly shows that the TCM 120 has a very good performance. Moreover the high radio data rate minimizes both the radio message length and energy consumption. This is advantageous if especially network traffic and network lifetime have to be considered.

Table 1. Platform comparison

Parameter	EnOcean TCM	Crossbow Mica 2	Telos	EyesIFX
MCU	PIC 18F452	ATMEL 128 L	TI MSP 430	TI MSP 430
Current Active Mode (mA)	13.40	8.90	1.70	1.40
Current Sleep Mode (µA)	8.00	27.70	3.30	2.90
Tx Current (mA)	9.90	15.50	19.40	11.90
Rx Current (mA)	15.80	10.70	21.10	14.80
Supply Voltage (V)	4.75	2.70	1.80	2.10
Data rate (kbps)	120.00	38.40	250.00	64.00
Energy per bit - Tx (µJ)	0.92	1.72	0.15	0.44
Energy per bit and mW (µJ/(bit*mW))	0.09	0.52	0.15	0.31
Tx Output Power (dBm)	10.00	5.00	0.00	1.50
Tx Input Power / Tx Output Power	4.70	13.23	34.92	17.67
Energy per bit - Rx (µJ)	1.16	1.38	0.16	0.53
Rx Sensitivity (dBm)	-95.00	-98.00	-94.00	-95.00

3. EMBEDDING PROCESS

In the case of hardware platforms like Mica 2 [8] or Telos the standard TinyOS compilation process is as follows: the nesC-compiler NCC, which is an extension to GCC, is started and generates an executable for the respective platform as well as a huge C-file called "app.c" in which all nesC source files are preprocessed and collected. Right after that a binary form of the executable is written into the program memory of the respective mote. Hereby the code is pushed through a programming interface like JTAG into the program memory of the MCU. Each type of MCU requires its own GCC-compiler libraries, which are to be embedded into the TinyOS compilation process.

Due the PIC's missing GCC option an alternative way had to be found in order to make the TinyOS nesC source files running on the PIC18F452 based EnOcean platform.

Like in the Wisenet project [3] our method of making the TinyOS nesC-files to match with the Microchip PIC18F452 and the

associated Microchip C18 C-compiler is build upon the "app.c"-output-file of the nesC-compiler (see "Figure 1"). We had to include the PIC into the appropriate sections of the TinyOS- make process. Additionally we created a platform directory for all PIC specific nesC source files. In that way we were able to invoke NCC with our platform as target. After this step we used a Perl-script to modify the "app.c" such that it can be compiled by the PIC specific C18 C-compiler. Hereby the respective Perl-script of the Wisenet project served as template. In contrast to [9] we did not have to reduce the TinyOS inherent function call depth since the 31-level call stack of the PIC18F452 is sufficient.

To test the feasibility of this approach we first implemented the PIC specific interrupt service routine (ISR) with its appropriate syntax in the "app.c"-file of the very simple TinyOS "Blink"-application which periodically toggles a LED.
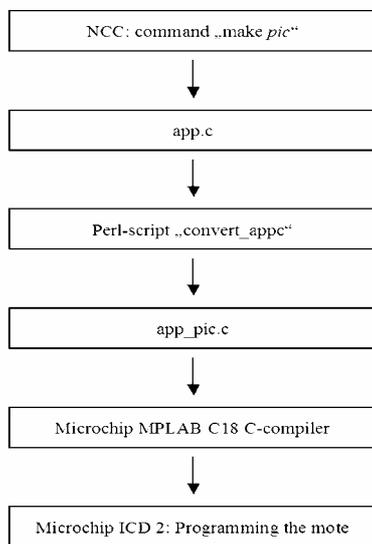

Figure 1. Compilation process

4. TinyOS IMPLEMENTATION

By porting TinyOS to the EnOcean TCM 120 it was our intention to expose most of the peripheral modules offered by the MCU. Thereby it was our goal - in order to maintain compatibility with the existing TinyOS applications - just to modify the hardware abstraction layers (HPL) and the lower presentation layers as much as necessary. Our TinyOS software package for the standard TCM120 hardware, which is available for download on the TinyOS sourceforge site [5], contains the following modules:

- Radio Interface module, which handles both the radio transmitting and receiving and provides a received signal strength indicator (rssi)

- Timer module, which is driven by the main oscillator and generally offers a resolution of 1.024 milliseconds with a precision of +/- 25.6 µs

- UART module, which runs with a default serial data rate of 57.6 kbps

- ADC module (10-bit successive approximation a/d)

- Power Management, which forces the mote into low power sleep mode with a wake-up time of 2 ms plus 1024 clock cycles and the MCU internal watch dog timer serving as wake-up source

5. CONCLUSION

The presented embedding of the commercial Microchip PIC18F452 based platform EnOcean TCM 120 into TinyOS showed the feasibility of porting TinyOS to a MCU without GCC support. Our platform shows very good performance when compared with the other TinyOS platforms. In the current design state our platform can serve as gateway between self-powered wire-and batteryless ambient energy sensors form EnOcean, which are already available on the market, and the steadily increasing TinyOS world.

In order to reduce power consumption to a level such that the energy for the transceiver modules can be entirely sourced from the primary process or the environment we will replace the PIC18F452 by the new PIC18F4620 with *nanoWatt Technology* which will simultaneously extend the RAM (4 KB) and ROM (64 KB) resources significantly. A first test showed that power consumption of the new PIC in sleep mode will be reduced by a factor of at least 2.5 which will bring the TCM 120 to the level of the MSP430 platforms.

Additionally we will reorganize the routing and pinning to make both SPI and I^2C useable and an asynchronous timer option available, which is required for enabling a controlled sleep mode while maintaining the system wide time-reference. Moreover we will develop a base platform on which the modified TCM will be mounted. This base platform will contain additionally hardware such as a 32 kHz crystal, a low power reference voltage supply, and an EEPROM. Beside it offers slots for a sensor and a power module, in which the latter is currently developed within a master thesis.

6. REFERENCES

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

[2] Handziski, V., Polastre, J., Hauer, J.-H., and Sharp, C., "Poster Abstract: Flexible Hardware Abstraction of the TI MSP430 Microcontroller in TinyOS," presented at the SenSys'04, Baltimore, Maryland, USA, 2004

[3] http://cegt201.bradley.edu/projects/proj2003/wisenet/index.html

[4] http://cvs.sourceforge.net/viewcvs.py/tinyos/

[5] http://cvs.sourceforge.net/viewcvs.py/tinyos/ tinyos-1.x/contrib/hsu/

[6] http://www.enocean.com

[7] http://www.gnupic.org

[8] http://xbow.com

[9] Lynch, C., O'Reilly, F., PIC-based TinyOS Implementation, In proc. 2nd European Workshop on Sensor Networks, Istanbul, Feb 2005, pp. 378-385, 2005