Evaluation of Power Efficiency for Digital Serial Interfaces of Microcontrollers

Konstantin Mikhaylov and Jouni Tervonen
RFMedia Laboratory
Oulu Southern Institute,University of Oulu
Vierimaantie 5, 84100, Ylivieska, Finland
Email: {konstantin.mikhaylov,jouni.tervonen}@oulu.fi

***Abstract*—Over the recent years, novel low-power microcontrollers have been introduced. This has allowed the development of various applications, which can operate over long periods of time and fulfill their tasks having very limited amount of energy, such as e.g., Wireless Sensor Networks (WSN). Nonetheless, for fulfilling their tasks such devices in addition to the central processor often have to include several microcontrollers or some peripherals, such as sensors, external memory chips or other application-specific hardware. All those peripherals, except the energy for actual operation also require some energy for communicating to the central processor. In the paper we are investigating and comparing the energy consumption for three most widely used embedded systems' digital serial interfaces, namely I^2C, SPI and UART. The presented results have been obtained using the real-life microcontroller (PIC18F family from Microchip) and reveal the energy consumption for different interface implementation methods (hardware vs. software) and various scenarios (idle interface, different data transmit or receive scenarios). The presented data can be valuable as well for researchers and engineers and allow to choose the most energy efficient communication interface and its implementation method to be used in the most energy-critical applications.**

***Keywords*-digital serial interface; energy consumption; embedded systems; power efficiency; wireless sensor networks**

I. Introduction

The technology advances of the recent years allowed to develop wide range of novel embedded systems with extremely low power consumption. Nowadays, the low-power embedded processors are widely used as core processing systems in various portable consumer electronics, including communication devices, computers and medical devices [1], [2]. Nonetheless, although the embedded processors themselves often have quite high power efficiency, the peripheral components (e.g., sensors, external memory chips or cards, graphical user interfaces hardware) that are required in many applications could significantly increase the energy consumption for the whole device. The energy, that is consumed by those peripherals consists as well of the energy for their actual operation and the energy for peripheral communication with core processor. As has been revealed e.g., in [3], sometimes the energy consumption for the communication with a peripheral appears to be even higher than the one for this peripheral operation. Especial importance the problem of energy efficiency overall and energy efficiency for communication in particular achieves for applications, where continues autonomous operation is required, such as e.g., many real-life applications of Wireless Sensor Networks (WSN) [4] or personal communication devices with multiple processors [5]. Although some disparate scraps of information can be found in the works focusing the energy efficiency of the WSN platforms and other portable communication devices (see e.g., [3], [5]), the actual problem of energy efficiency for the most widespread embedded systems' communication interfaces, to the best of our knowledge, has not been considered previously at all. Therefore, in this paper, we *evaluate the most widely used digital interfaces and compare them from the point of view of energy efficiency*.

This paper has been accepted for publication in the proceedings of WSN-ADT Workshop that had been held in conjunction with NTMS 2012.

II. Serial Interfaces

As revealed in [6] and [7], out of the wide range of existing communication interfaces, nowadays the most widely used are the analog interface and I^2C, SPI, UART and 1-wire digital interfaces. Each of these interfaces has some specifics, that influence the data rate, energy consumption and available additional interface's features (e.g., robust communication or device identification support). Many of those interfaces are nowadays already implemented by the hardware modules that are integrated into the majority of contemporary microcontrollers [8].

A. UART

The interfaces based on Universal Asynchronous Receivers/Transmitters (UARTs) (later in the paper those will be addressed as "UART interfaces") are often used for implementing such standards as Electronic Industries Alliance (EIA) Recommended Standards RS-232, RS-422 and RS-485. UARTs are nowadays one of the most commonly used method for communication between an embedded system and an external device [9]. As Fig. 1 revels, UARTs provide full-duplex asynchronous serial peer-to-peer communication. The data transmission over UART usually starts with a start bit(S), that alerts the receiver that a word of data is about to be sent and allows the receiver to synchronize clocks with the transmitter. After the start-bit, the actual data is transmitted starting with the least-significant bit. Once a data word has

978-1-4673-0229-6/12/$31.00 ©2012 IEEE

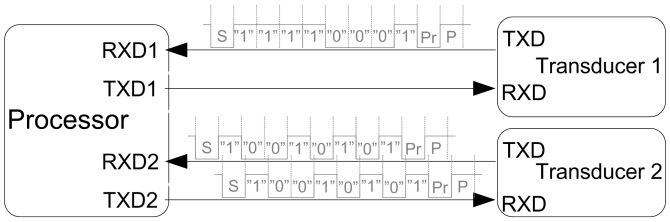

Fig. 1: UART interface and data format

been sent, transmitter may issue the parity bit(Pr) to provide a simple error checking and/or a stop bit(P) to signalize the end of data word transmission.

B. SPI

The Serial Peripheral Interface (SPI), that has been popularized by Motorola, is a synchronous serial interface that operates in full duplex mode [10]. The typical method for connecting several SPI slave devices to a master is presented in Fig. 2. As the figure shows, SPI bus utilizes three common lines for all slave devices: clock (SCLK); master output, slave input (MOSI); master input, slave output (MISO); and a separate chip select ($\overline{CS}$) line for each slave device. Therefore, before starting the communication, the SPI master device pulls down $\overline{CS}$ line of the required slave device to select it. The SPI specification does not define neither any maximum data rate (for existing devices it reaches dozens MHz) nor any particular addressing scheme or acknowledgment mechanism.

C. I^2C

The developed by Philips Inter-Integrated Circuit (I^2C) interface and its data format are presented in Fig. 3. As this figure reveals, the I^2C interface uses two common physical lines for clock (SCLK) and data (SDA), which are pulled-up with resistors (Rp) [11]. The I^2C interface supports multiple slave and master devices; therefore, the master device starts the communication by sending the start bit(S) and the unique slave device address (which usually consists of 7 bits, although addresses of 10 bits are also defined in recent I^2C standard revisions). Together with the slave device's address, the master transmits the 1-bit Read/Write ($R/\overline{W}$) for defining the communication direction, which would be used until the stop bit (P) closes the current session. The I^2C communication protocol implements per-byte acknowledgments ($A/\overline{A}$). The standardized data rates for I^2C devices are 10 kbit/s, 100 kbit/s and 400 kbit/s, although most recent I^2C revision [11] also supports the rates of 1 Mbit/s and 3.4 Mbit/s.

The results of comparison for different serial interfaces are summarized in Table I.

III. Experiment Setup

For evaluating the energy consumption for I^2C, SPI and UART interfaces in real-life conditions we have used the testbed build around two PICKit 3 development boards [13] from Microchip with PIC18F45K20 microcontrollers onboard

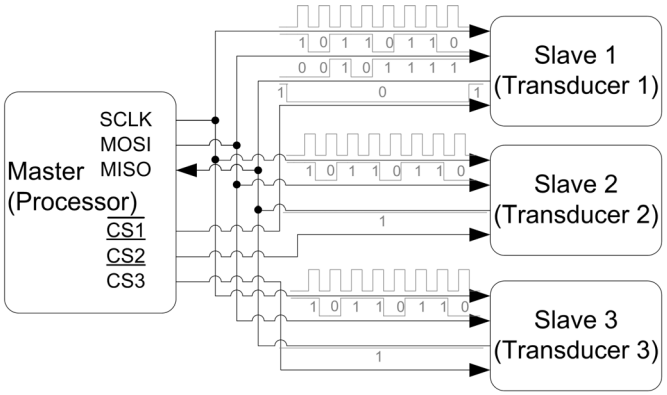

Fig. 2: SPI interface and data format

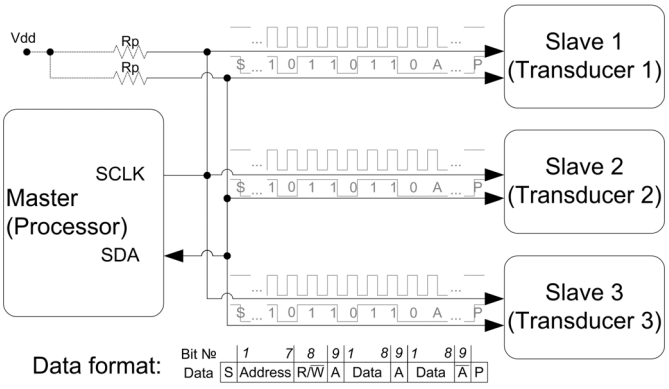

Fig. 3: I^2C interface and data format

[14]. The microcontroller of the first board was programmed to transmit the data over the serial interface (TX board), while the microcontroller of the second board was used to receive the data (RX board). The pins of the microcontrollers that were required for implementing the serial interfaces have been connected together using 10 cm long wires.

For implementing the required serial interfaces we have used two different options: interfaces were implemented using the available hardware modules of the microcontroller (Master Synchronous Serial Port (MSSP) and Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) modules [14]) and using General-Purpose Input Output (GPIO) pins with complete interface implementation by microcontroller software (i.e. bit-bang implementation). For implementing the interfaces was partially used PIC18 MPLab library from Microchip [15].

The structure of the developed testbed is presented in Fig. 4. As revealed in Fig. 4, the developed testbed was using current shunt method for consumed energy measurement [16]. The consumed by testbed current, that was later used to estimate the consumed energy (see (1)), was estimated by measuring the voltage drop (V_{shunt}) over measurement period from T_0 to T_{meas} on known shunt resistor ($R_{shunt} = 4.7Ohm$) using the oscilloscope. For each implementation option and each interface at least ten measurements for E_{cons} were made

TABLE I: Comparison of serial interfaces

Parameters	UART	SPI(4 wires)	I^2C
Number of lines:	2	3 + 1 per slave	3 with pull-ups
Data transfer:	asynchronous duplex	synchronous duplex	synchronous simplex
Communication model:	peer-to-peer	master-to-slave	master-to-slave
Number of master devices:	-	multiple	multiple
Number of slave devices:	-	multiple	119[a]or 1024[b]
Maximum data rate, Mbit/s:	50[c]	70[d]	3.4[e]
Interface features:			
Robust communication mechanisms:	parity check[f]	-	acknowledgment
Device identification mechanisms:	-	-	address
Overhead per packet, bits:	2 per byte	0	11 + 1 per byte

[a] using 7-bit addresses
[b] using 10-bit addresses
[c] according to [9]
[d] according to [12]
[e] according to [11]
[f] optional

TABLE II: General testbed and experiment parameters

Parameters	UART		SPI		I^2C	
	HW	SW	HW	SW	HW	SW
Required testbed resources[a]						
Flash, bytes (TX)	1151	1215	1132	1261	1122	1395
RAM, bytes (TX)	274	281	272	281	274	281
Flash, bytes (RX)	1227	1187	1045	1010	1125	1190
RAM, bytes (RX)	291	282	288	278	290	278
Experiment parameters and settings						
Pull-up resistors, kOhm	-	-	-	-	100	100
Data rate, kbit/s[b]	15.71	15.82	15.66	15.84	15.94	15.8
Maximum data rate, kbit/s[c]	125	65	166.7	74.6	222.2	38

[a] using nominal optimization settings of IDE
[b] the data rate for interface during experiment
[c] using 8MHz core clock frequency and 3.0 V supply voltage

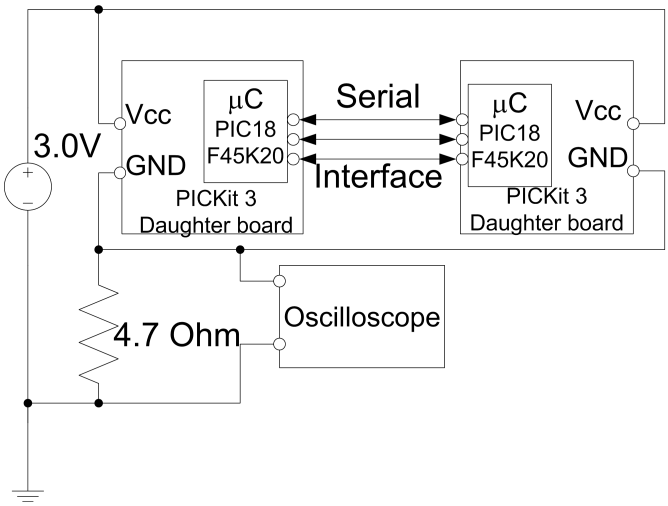

Fig. 4: Experiment setup

and the average value (E_{exp}) for all the measurements has been calculated. During all the measurements, the testbed was supplied from the same laboratory DC power source with 3V voltage (V_{cc}) and nominal clock frequency of microcontrollers in active mode both for RX and TX boards was 8 MHz. The maximum error for the measurements is estimated to be in the order 5 μs for time and 4% for energy consumption measurements.

$$E_{cons} = \int_{T_0}^{T_{meas}} \frac{V_{cc} \cdot V_{shunt}(t)}{R_{shunt}} dt \qquad (1)$$

The energy consumption measurements have been done in three stages. First of all, energy consumption for TX and RX boards has been measured when the serial interfaces on both boards have been initialized, but no transmission was ongoing. The second and third measurements were made when the transmission was ongoing: during the second measurement the RX board was selected as data target, during the third measurement - RX slave board was not selected as target (for UART third measurement was not made as it is impossible to deselect the RX). The energy consumption during the second and third stages has been measured for two cases: single byte transmission and transmission of nine data bytes (T_0 - start of packet transmission; T_{meas} - end of packet transmission) in single packet (all required service operations, e.g., start/stop bits transmission or slave address transmission for I^{2}C are also accounted).

To have the possibility to compare the results of the measurements, all interfaces have been tested at the data rate of around 15.8kbit/s (the measured real-life data rate (estimated basing on the time for transmitting single bit in the middle of data byte transmission, all service operations excluded) is presented in Table II). The main reasons for the difference between the data rates for various interfaces and implementations are different mechanisms for clock generation for various interfaces and microcontroller clock instability. During second and third stages, the transmitted data (in hexadecimal) was 0xAA for single byte transmission and 0x000103070F1F3F7FFF for nine-byte data packet.

IV. Results

The general testbed and experiment parameters and the results of energy consumption measurements for implemented interfaces are summarized in Tables II and III respectively. To be able to compare the results for different interfaces and their implementation methods, the values of energy for data

transmission have been normalized using Eq. 2 to the common data rate value (15.8 kbit/s). In Eq. 2 $Datarate_{exp}$ stands for the experimentally defined data rate in kbit/s (see Table II), E_{exp} - for the experimentally measured energy consumption for transferring one or nine data bytes respectively(see Table III).

$$Enorm = \left(\frac{15.8 - Datarate_{exp}}{15.8} + 1 \right) \cdot E_{exp} \qquad (2)$$

As the presented data (see Table II) reveal, surprisingly, the complete interface implementation in software required almost same amount of resources as the implementation of interfaces using hardware modules. Besides, for some cases (slave receivers for UART and SPI), the resource requirement for bit-bang implementation was even slightly lower than for hardware-based one. The programs that have been used for estimating those resource requirements had the same functionality and used Microchip PIC18 MPLab library [15] for implementing interfaces using hardware modules and proper assembly-based code for implementing the interface in software. Nonetheless, as revealed in Table II, the maximum achieved data rate (i.e. the maximum achieved data rate for error-less communication between the boards working at 8 MHz microcontroller clock frequency) for hardware implementation is significantly higher than the one for complete software implementation.

Table III reveals, that the energy consumption for data transmission over the interface implemented in software is significantly higher than the one for the same interface implemented using inbuilt microcontroller hardware modules. This is especially notable for SPI interface, for which the difference in energy consumption is almost 3 times. The major reason for this difference is the possibility for using for hardware-implemented interfaces the interrupts on data transmission/receival complete. Those allow to keep the microcontroller in sleep mode most of the time thus significantly lowering the power consumption. This is especially important for low-duty cycle applications such as e.g., many WSNs.

As one can see from Table III, the most efficient interface from the point of energy consumption was SPI. This is not surprising, as SPI is the only interface among tested ones, that does not have any overhead data transmission. If to compare the measurements for software implementation (this is the most rational as interface implementations in hardware required to use different sleep modes due to different sets of required peripherals such as timers and clock systems), the UART required 22.5% and I^2C - 26.6% more energy for transferring 9 data bytes. Also, as can be seen from presented data, the difference in energy consumption between UART and I^2C for single byte transmission was around two times (as I^2C had to transmit one extra byte with slave address) and for 9-byte transmission this difference almost vanished and was only 4%. This allows to expect that for packets with over 10 data bytes the I^2C interface will be more energy-effecient than UART.

V. Discussion and Conclusion

In this paper we have discussed and evaluated the most widespread digital serial interfaces (namely I^2C, SPI and UART), that are used by different embedded systems and have evaluated real-life energy consumption for different scenarios and implementations of those interfaces.

As the data presented in the paper reveal, of all tested interfaces SPI had the lowest power consumption both with inactive interface and during data exchange for the case, when all interfaces were using the same data rate. Nonetheless, SPI interface does not support any standardized mechanism for device identification or robust communication implementation. The presented data reveal that SPI slave receiver implementation required the minimum amount of resources comparing to other tested interfaces, which is caused by very simple data format in SPI. Also this was the reason, why complete software SPI implementation allowed to get maximum data rate among all tested interfaces.

The interface based on UART required at least 22-23% more energy for transferring the same amount of data comparing to SPI. This is mainly caused by the necessity to have some overhead for communication via UART - namely start and stop bits for each byte. The implementation of UART interface for slave device required more resources than implementation of SPI interface due to higher complicity for data receival procedure without any specific clock signal availability. Unfortunately, the most widespread UART-based interfaces support only peer-to-peer connection, which makes it impossible to connect multiple UART devices to single UART port of the microcontroller. The UART interface usually can use a parity check bit for each data byte, but has no standardized mechanisms for device identification support.

The single byte transmission using I^2C interface required more than two times more energy comparing to SPI, although for longer packets the efficiency of I^2C interface increases and for packets over 9 bytes it becomes more efficient than UART. The reason for it is the necessity for I^2C to start each data packet with 1 byte containing slave device's address. The software implementation for I^2C interface required rather high resource consumption, which is caused by high complicity for I^2C communication procedures (e.g., use of start, restart and stop bits and acknowledgment). This is also the reason why complete software implementation for I^2C interface had almost two times lower maximum achievable data rate than the other interfaces. Surprisingly, hardware implementation for I^2C had the maximum achievable data rate of all tested interfaces, which should be caused by the features of interface hardware modules implementation in PIC18 microcontrollers. The I^2C has the support for device identification (using 7-byte addresses) and uses per-byte acknowledgments during data transmission.

The data presented in paper reveal, that although implementation of serial interface in software requires approximately same amount of resources as implementation in hardware, it causes significantly higher power consumption (20%-300%

TABLE III: Energy consumption for evaluated serial interfaces

Parameters	UART		SPI				I^2C			
			RX selected		RX unselected		RX selected		RX unselected	
	1 byte	9 bytes	1 byte	9 bytes	1 byte	9 bytes	1 byte	9 bytes	1 byte	9 bytes
Interface implementation in hardware										
Power consumption with inactive interface, mW	6.06	6.06	4.66	4.66	4.66	4.66	6.13	6.13	6.13	6.13
Communication:										
Required time, ms	0.68	5.78	0.52	4.6	0.52	4.6	1.26	5.91	1.26	5.91
Required energy, μJ	7.31	53.31	2.5	23.53	2.42	22.52	8.11	31.71	8.01	30.23
Normalized energy[a],μJ	7.27	53.01	2.48	23.33	2.4	22.32	8.19	31.99	8.08	30.5
Interface implementation in software										
Power consumption with inactive interface, mW	11.04	11.04	11.17	11.17	11.17	11.17	11.23	11.23	11.23	11.23
Communication:										
Required time, ms	0.64	5.72	0.51	4.62	0.51	4.61	1.23	5.85	1.23	5.86
Required energy, μJ	9.00	80.06	7.26	65.25	7.26	65.25	17.45	82.84	17.44	83.6
Normalized energy[a],μJ	9.01	80.16	7.28	65.42	7.28	65.42	17.45	82.84	17.44	83.6

[a] see Eq. 2

depending on the interface) and allows to achieve lower maximum communication data rate (2-6 times lower comparing to implementation in hardware).

Although all our experiments have been executed using only one hardware platform (namely PIC18F45K20 microcontrollers), we assume that obtained results can be expanded to some extent also for other hardware platforms. We believe that the data presented in the paper provide valuable information, that can help to choose the most energy efficient communication interface and its implementation method to be used in the most energy-critical applications. One of major target application groups, that can benefit from the presented data utilization, are the Wireless Sensor Networks. Indeed, as has been revealed e.g., in [7], nowadays the SPI and I^2C interfaces are very widespread among different digital sensors (e.g., among the available digital temperature sensors I^2C bus is used by 57% and SPI - by 10%). Therefore, for obtaining high energy efficiency e.g., in WSN one should consider not only the actual parameters of a sensor, but its communication interface as well and the data presented in the paper provide the background information for it.

In the current paper we have evaluated the serial interfaces for one particular data rate and estimated the maximum data rates that can be achieved for various implementation options. In future we are planning to make similar tests for other hardware platforms and other data rates which would allow to understand better the influence of serial interface's parameters on the power consumption of the whole system. This will allow to develop, in future, a model that could predict the power consumption for communication over serial interfaces between various components of a portable electronic device for specified interface settings and environment conditions.

ACKNOWLEDGMENT

The authors wish to thank all parties who have financially supported this study. This work has been supported by European Regional Development Fund, Council of Oulu Region, the Sub-regions of Ylivieska and Nivala-Haapajarvi, Town of Ylivieska and Kerttu Saalasti Foundation.

REFERENCES

[1] B. Munsey. (2011, Aug.) New developments in battery design and trends. House of Batteries. [Online]. Available: http://www.houseofbatteries.com/documents/New%20Chemistries%20April%202010%20V2.pdf

[2] K. Mikhaylov, J. Tervonen, and D. Fadeev, "Energy efficiency aware application development using programmable commercial low-power embedded systems processors," in *Embedded Systems - Theory and Design Methodology*, K. Tanaka, Ed. Rijeka, Croatia: InTech, 2012, pp. 407–430.

[3] G. Mathur, P.Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low power data storage for sensor networks," in *Proc. IPSN 2006*, 2006, pp. 374 –381.

[4] M. Kuorilehto, M. Kohvakka, J. Suhonen, P. Hamalainen, M. Hannikainen, and T. Hamalainen, *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*. Hoboken, NJ: John Wiley & Sons, 2007.

[5] N. Sklavos and K. Touliou, "A system-level analysis of power consumption and optimizations in 3G mobile devices," in *Proc. NTMS 2007*, May 2007, pp. 225–235.

[6] K. Mikhaylov, J. Jamsa, M. Luimula, J. Tervonen, and V. Autio, "Intelligent sensor interfaces and data format," in *Intelligent Sensor Networks: Across Sensing, Signal Processing, and Machine Learning*, F. Hu and Q. Hao, Eds. London, UK: Taylor and Francis LLC, CRC Press, to appear in 2012.

[7] K. Mikhaylov, T. Pitkaaho, and J. Tervonen, "Plug-and-play mechanism for plain transducers with digital interfaces attached to wireless sensor network nodes," *submitted for publication*.

[8] G. Gridling and B. Weiss, *Introduction to Microcontrollers*. Vienna, Austria: Vienna University of Technology, 2007.

[9] H. Yuan, J. Yang, and P. Pan, "Optimized design of UART IP soft core based on DMA mode," in *Proc. ICIEA 2010*, June 2010, pp. 1907 –1910.

[10] *SPI Block Guide v. 03.06*, Motorola Semiconductor Products Inc. Std. S12SPIV3/D, 2003.

[11] *I2C - bus specification and user manual rev.03*, NXP Semiconductors Std. UM10 204, 2007.

[12] D. Johnson, "Implementing serial bus interfaces using general purpose digital instrumentation," *IEEE Instrumentation Measurement Magazine*, vol. 13, no. 4, pp. 8 –13, August 2010.

[13] "PICkit 3 Programmer/Debugger Users Guide," Microchip Technology Inc., Chandler, AZ.

[14] "PIC18F46J50 family data sheet," Microchip Technology Inc., Chandler, AZ.

[15] "MPLAB C18 C compiler libraries," Microchip Technology Inc., Chandler, AZ.

[16] Z. Nakutis, "Embedded systems power consumption measurement methods overview," *MATAVIMAI*, vol. 2, no. 44, pp. 29–35, 2009.