Energy Efficient Data Restoring After Power-Downs for Wireless Sensor Networks Nodes with Energy Scavenging

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

Abstract— **The article focuses on the new problem, which became topical for Wireless Sensor Networks (WSN) with energy harvesting possibility and which is in the realization of system and user data restoring for nodes after unexpected power-downs due to the temporal unavailability of harvested energy. Different possibilities for the creation of the restore point and data restoring after power-downs are described and evaluated in the article, which include the usage of microcontroller internal RAM and Flash memory, the usage of external EEPROM and NVRAM chips or the radio interface with joint node data storing. The evaluations for the article were made using presented Restorable Event-Based Operation System (REBOS) which is intended to solve the problem of OS and user application automatic restoring point creation and restoring after power-downs. The evaluation of energy consumption for different restoring methods is included.**

Keywords-wireless sensor network; microcontroller; restore; back-up; operation system

I. INTRODUCTION

The recent technology improvements, which allowed the integration of low power consumption with adequate processing power, allowed the introduction of the new applications for Wireless Sensor Networks (WSN). One of the most interesting possibilities for the usage of these technologies lies in the field of WSN with environment energy harvesting. During recent years, numerous demonstrations have been presented which showed the possibility of collecting and utilizing the energy from different environment elements, like: light [1,2,3,4,5], temperature difference [1,3,6], vibration [1,3,7,8], water, air or gas flow [1,3,4] or electrical and magnetic fields [1,3].

Such technologies allow the development of new types of systems, which do not require any service after deployment and which can fulfill their tasks autonomously for extremely long periods of time. Nevertheless, the usage of energy harvesting also brings some special features for such WSN. One of the most important of them from the point of practical implementation, which comes out of energy scavenging usage, is the construction of the node power supply system. The usage of energy scavenging demands from the node power supply system to have high amount of possible recharge cycles and the possibility to charge the system using low energy, collected from environment. All that leads to the fact that the usage of standard accumulators with energy scavenging becomes impossible, which leads to usage of some sort of capacitance for storing gathered from environment energy for most presented WSN nodes with energy scavenging[5, 6, 7]. Although such systems allow to get numerous recharge cycles and can be charged using existing energy scavenging technologies, but usually the energy storing capacitance for these systems is quite small[5, 6, 7]. This makes it impossible for such systems to support long operation time for the case, when none or only small amount of scavenged energy is available. The other problem, which appears with using energy scavenging technologies is the availability scenario of harvested energy, as in many cases it is impossible to predict when the scavenged energy would become unavailable and when it will be possible to start using it again. This fact leads to the problem, which differentiates WSN nodes with energy scavenging from the nodes with static supply, which is that power supply for systems using energy scavenging can switch on and off quite often depending on when the collected energy would become available or would stop being available.

Until now, to the best of our knowledge, in the most WSN application programs, the disappearance of power for the node would result in the full system reboot with full reinitialization after power would become available. This usually leads to lose of all collected data and made settings, which are not saved in power independent memory. It can be affordable, when the system uses static power supply or battery and for which power downs should not happen during normal operation but, due to described above specialties, for the systems with energy scavenging temporary power downs are usually a part of normal system operation. That is why, in this article we evaluate possible ways for program and data backup implementation and present Restorable Event Based Operation System (REBOS) which is intended for use in WSN with energy scavenging. The system was developed and tested for Texas Instruments MSP430F2274 low-power microcontroller, which has been recently used in many energy harvesting demonstrations [5,6,7].

II. METHODS FOR STATE RESTORING IN WSN

To have the possibility for OS or any other data restoring after a power down, the current working mode and all required data for all running processes should be periodically saved. To define the possible possibilities for storing required data, it is necessary to overview the structure of a contemporary most-general WSN node, which is presented on Fig.1.

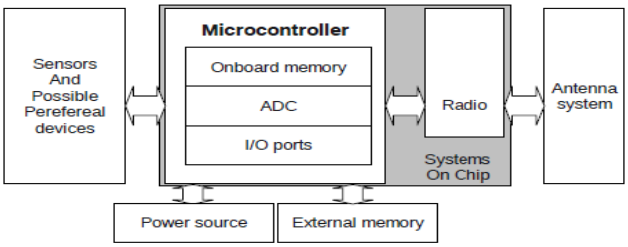

Figure 1. The typical structure of a WSN node.

From Fig.1, it is possible to see that for storing the data for OS or for any other program for a WSN node it possible to use one of the following methods:

- Storing data in microcontroller RAM
- Storing data in microcontroller Flash
- Storing data in external memory
- Storing data in the memory of other WSN node using a radio interface.

Below we will review the advantages and disadvantages for each of these opportunities in more details.

A. *Usage of Microcontroller RAM for State Restoring after Power Down*

The main advantage of RAM, which defines its usage for program data storing is that it can be accessed for both reading and writing by single bytes and that operations with RAM can be made very fast and are usually the most efficient from the point of power consumption [9, 10]. Usually the main limiting factor for using RAM is the size of RAM, which is much smaller than microcontroller Flash or external memory. The other problem is that RAM is a volatile memory and is unable to store data for long periods of time without power supply. To estimate the possibility of using microcontroller RAM for storing data without available power supply, we have conducted the experiments using the MSP430F2274 microcontroller of eZ430-RF2500 development kits.

From Fig.2 it is possible to see that during short periods of power unavailability (up to 5 seconds), the data in microcontroller RAM will high probably retain in memory on power up and can be reused for restoring the system state.

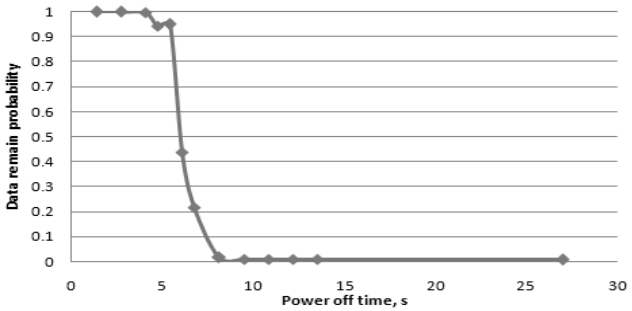

Figure 2. Probability of RAM data retaining over time in power off mode.

B. *Usage of Microcontroller Flash for State Restoring after Power Down*

The part of memory, which is usually intended for storing the microcontroller program and all data when the power supply is not available, is Flash. The main advantage of using Flash, except that it retains the data without power supply, is the size of Flash, which is usually bigger that of RAM.

Meanwhile, the usage of Flash has several disadvantages. The main of them is the complicity of writing to Flash. Before writing to some Flash part, where there is already some data, it is necessary to clear the old data and the clearing of data in Flash is done in blocks which makes it impossible to rewrite a single byte in a Flash. The second problem is that although reading from Flash is usually very fast, but writing to it and clearing of Flash are much slower and therefore consume much more energy, than writing to RAM. Thirdly, as the microcontroller program and interrupt vectors are located usually in Flash, writing to Flash would stop program execution till the write end, which can result in missing or delay in processing for some events. Finally, writing to Flash often requires higher voltage, than operations with RAM or normal-mode program execution. According to [11], the minimum voltage for microcontroller Flash writing is 2.2 V while the microcontroller works for voltage above 1.8 V [11] (1.5 V according to [10]).

C. *Usage of External Memory for State Restoring after Power Down*

The third option, which can be utilized for data restoring, is the usage of external from microcontroller memory for saving the state during power down. It is possible to use different sorts of external memory chips, like Flash, Electrically Erasable Programmable Read-Only Memory (EEPROM) or Non-Volatile Random Access Memory (NVRAM) and the advantages or disadvantages for it would be defined by the specific memory chip architecture.

Nevertheless the usage of external memory chip would have some common features, e.g., comparing with internal memory, it requires having some additional energy wasted for communication with the memory chip during reading or writing to memory. The second impact, that is the same for using any external component with a WSN node, is that addition of external memory chip would increase the size, the price and, in most cases, the sleep mode power consumption of the node.

D. *Usage of Cooperative Data Storing and Radio Interface*

The final possibility is the usage of the radio channel for transmitting the required data to other WSN nodes and the usage of other nodes memory for data storing. Although this method has significantly higher power consumption, than all other methods and adds some overhead for radio communication and requires to have reliable (errorless) radio communication, it does not require any resources from a node for on-node data storing and allows to have multiple places for restoring information storing (depending on the amount of nodes in the transmit range of a node). Meanwhile, the sensibility of this method usage also depends on the structure of the network, e.g. if the network has homogenous structure and consists of only of the nodes, using environment energy

scavenging and the energy for all nodes can become unavailable at the same time – the usage of the described method would be hardly effective.

III. Implementation of System Restoring for Restorable Event Based Operation System (REBOS)

In II were discussed four possible opportunities for the implementation of state restoring, which can be used for WSN. To automate the process of restore point creation and system restoring and to improve the solution portability, it was implemented as OS, which would support restoring both for OS itself and for running user threads.

To minimize the power consumption the OS was made event-driven – it switches to the active mode only for event processing and stay in low-power mode with minimum power consumption otherwise.

The work of the whole developed OS is based on four tables, the size of which can be adjusted depending on available resources and application: the table for OS state, the table for all active devices that can create the expected by launched threads events, the table for active threads that are waiting for events and the waiting query for called threads. The structure of the system kernel is presented on Fig. 3.

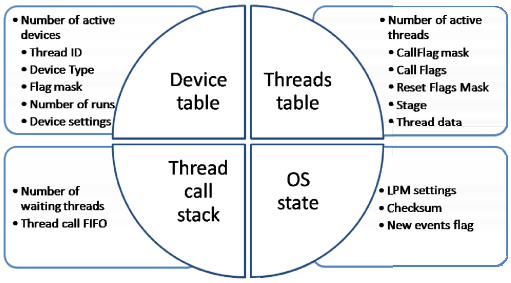

Figure 3. The structure of REBOS kernel.

For every user, function at first a thread at the thread table is initialized. This table defines for which event which user function should be called and contains the pointer to required function user data. After thread initialization, the device initialization for that thread is done. As soon as all required threads and devices are initialized, the microcontroller switches to the low-power mode, which is defined by the settings in OS state table and stays in the low power mode until some event would happen. Once some event would be detected, during interrupt processing the necessary flags for corresponding to device thread is set. After the interrupt processing, the OS kernel is activated and the flags for all active threads are checked and the thread functions that qualify the calling conditions are added to a thread call query. Later the threads from the query are being called until all the required threads would be processed, after which OS automatically sends the microcontroller to the low power mode again.

Such simple structure allowed to make the minimum requirements for OS kernel as low as 1kbyte of Flash and 30 bytes of RAM. The OS with the application for eZ430-RF2500 boards that use radio communication required only around 3 Kbytes of Flash and around 60 bytes of RAM.

A. State Restoring Realization in REBOS

The restoring system was realized as the library for REBOS and implements all the possibilities, which were discussed in II and allows the tuning in the restoring system depending on the current application requirements. For the most general case, the restoring system is intended to implement 2 levels for restoring: so called "fast" and "reliable" restoring.

"Fast" restoring is based on the microcontroller RAM specialties, which were discussed in II.A and is intended for restoring the system after possible short time power-off switches. During fast restoring, the contents of microcontroller RAM, where the OS, device and user threads data should be located, is checked using the checksum, which is calculated and stored by OS on every "fast" backup point creation. In the case, if the calculated and saved checksums are equal, the OS and user data remaining in RAM is considered correct. After that, OS, using the data of the device table, reinitializes all required peripheral devices to last known state and launches the normal work of OS kernel. In the case, if the data in original OS locations has not retained, it is possible to use one or several of other methods, described in II.

The first possibility – is the usage of the second block in microcontroller RAM for storing the restore point. The main difference between this option and the one, used during "fast" restoring is that in this case a special block in RAM for storing the data is used, which allows the creation of the restore point that would not be affected by OS work.

The second option is the usage of Flash memory for storing the backup copy. If this possibility is used, OS uses the memory blocks at the end of microcontroller code memory for creating the restoring points. To avoid the loose of restoring data, if the power off situation would happen during new restoring point writing to Flash, was realized the possibility of using two different memory blocks with keeping two restoring points and sequent switching between the blocks. This method, although it requires more memory, allows the ensuring that at least one of restoring points in Flash would be valid for any circumstances.

The third possibility, which can be used for restoring, is external memory. Up to now REBOS has been tested with ST M95512and ST M48Z35memory chips. ST M95512 is the EEPROM memory with communication to microcontroller over SPI interface. The main disadvantages for using this type of memory are similar to using the microcontroller in-build Flash memory – the memory does not support the access by a byte and it has limited amount of write cycles. The main advantage of the memory is that it is capable for long storing the data without using any sort of power supply.

ST M48Z35 presents the other type of memory – a Non-Violate RAM (NVRAM) and contain integrated supply source – a lithium battery that allows the memory to retain the data also when the supply voltage is not available. This chip has all the advantages of RAM, e.g. every byte can be accessed separately although the usage of this chip significantly increases the size and the weight of the node and except that the battery, which is used for storing the data in memory chip, has limited capacity. The other limiting factor is defined by the architecture of the NVRAM. Typical NVRAM chips have parallel pinout which requires to use numerous microcontroller pins. For example, M48Z35 with 32 Kbytes of memory requires having 15 address pins, 8 data pins and 3 pins for control signals. So, to make it possible to use the M48Z35 with eZ430-RF2500 board it was required to increase the number of

available General Purpose Input/Output (GPIO) pins, which was done using two STMPE8018-bit port expander chips.

The final possibility is the usage of the radio interface for data storing. If this option is used, OS transmits or includes into data transmission the required restoring data. The neighbor nodes are excluding received restore data from the packet and, if they have enough resources, store the data in memory. To prevent the loose of previous restore points in case if the power down situation happens when the node is transmitting the information over radio, the receiver nodes firstly write the received data to a buffer and check its completeness and only after that rewrite the previous restoring point. For security considerations, the restore point information state before radio transmission can be encoded.

The restoring points can be created by either automatically by the OS, or by the command from a user thread, which allows to adjust the work of restoring system basing on available system energy and required reliability. Additionally two different restore point creation mechanisms were realized: the full and the compressed restore point creation. Full restore point stores all the REBOS and user functions memory to the restore point location, while during compressed point creation the data is analyzed and only the data required for restoring is copied. Although, the usage of compressed points allows the reduction of the required memory for the restore point storing, it also requires additional time and energy for data processing.

B. Evaluation of Energy Consumption for Using Different State Restore Methods in REBOS

To evaluate the efficiency of different suggested methods WSN node energy and time consumption measurement during restore point creation and system restoring have been made. The measurements were done using developed REBOS implementation on eZ430-RF2500 boards and the resulting data is presented in Table I.

TABLE I. ENERGY AND TIME CONSUMPTION FOR USING DIFFERENT STATE RESTORE TECHNIQUES WITH REBOS.

Restore methode	Restore point creation				
	Preparation		*Writing to Media*		
	Time, ms	*Energy, μJ*	*Time, ms*	*Energy, μJ*	*Minimum Vcc, V*
"Fast" restore[a]	0,29	3,55	-	-	1,8
internal RAM[a]	1,22	15,12	-	-	1,8
internal Flash	1,84	29,57	1,36	26,08[b]	2,2
NVRAM (M48Z35)	1,23	15,80	24,4	1045,7	4,75
EEPROM (M95512)	1,23	15,50	3,66	155,56[b]	1,8
Radio	1,23	15,20	4,21	224,03	1,8

Restore methode	OS Restoring				
	Reading from Media			*OS Restoring*	
	Time, ms	*Energy, μJ*	*Minimum Vcc, V*	*Time, ms*	*Energy, μJ*
"Fast" restore[a]	-	-	1,8	0,35	2,59
internal RAM[a]	-	-	1,8	0,43	3,23
internal Flash	-	-	1,8	0,43	3,41
NVRAM (M48Z35)	23,1	1106,0	4,75	0,43	3,87
EEPROM (M95512)	1,24	54,30	1,8	0,43	3,32
Radio	7,77	336,96	1,8	0,43	3,78

a. Can be used for restoring after short power-downs (up to 5 seconds)
b. Including Flash sector erase

Table I presents the data for the case, when the system was running using 8 MHz clock frequency with the supply voltage of 3.6 V (for external NVRAM – 5V). OS had one active user device and one active user thread with 12 bytes of user data (the overall restore point data size was 56 bytes).

As it can be seen in Table I, the time and energy required for using "fast" restore method are much lower than for other methods. The reason for it is that using the "fast" method there is actually no data copied to memory – during the restore point creation only the checksum calculation is done over the necessary data and during restoring – both the checksum calculation and required device re-initialization are done. The main disadvantage of this method is that whenever there will be OS or user data change, the checksum should be recalculated; otherwise during restoring the restore point would be defined as corrupted. Thus, restoring point creation using "fast" restore method should be made before going to the low-power mode and the method can be efficiently used only for restoring after short power-downs that happen during system stay in low-power mode.

The creation of the restore point in microcontroller RAM required almost 5 times longer time comparing with using "fast" restore technique due to necessity of actual coping the OS and user data from its initial location to restore point location. The required for restore point creation energy can be lowered by approximately 30% in the case if instead of the "compressed" the "full" restore point creation function is used. However, it would increase the size of the restore point from 56 to 100 bytes. Unlike "fast" restore, the usage of this method allows the system restoring after short power-downs regardless on the microcontroller state at power-down as the change of the restore point is done only by the special command. Also, this method can be used in the case if the system needs to make later backup to some earlier state during its work.

The energy and the time consumption for the restore point creation in microcontroller internal Flash were three to four times higher that of using microcontroller RAM. The other requirement for using this approach was that writing to microcontroller internal Flash required higher supply voltage than minimum required voltage for normal operation. Nevertheless, the main profit for this method is that microcontroller Flash can be used for the system restoring regardless of power-down length. Usage of "full" restore point creation in this case increases the energy consumption for almost 1.5 times comparing with "compressed" point creation. From presented data shows that this method has the lowest energy consumption among all the methods that do not have limitations on power shut-down time.

Usage of external M48Z35 NVRAM appeared to be the most ineffective from the point of energy and time consumption. Although, the memory operations were quite fast and did not require much energy (for writing the memory consumed around 17 μJ and 0.25 ms, for reading – around 70 μJ and 1.6 ms), the main energy was consumed by using GPIO extender chips, which are required for setting the address to every operation with memory chip. The other problem that complicates the usage of this type of memory, especially for WSN nodes with energy scavenging is high supply voltage which is required for operations with NVRAM (for M48Z35 it was equal to 4.75 V).

The usage of external M95512 EEPROM during the state restore, increased the energy consumption of the restore point creation approximately three times comparing with using internal microcontroller Flash memory, but it also allowed to decrease the limitation on minimum supply voltage during writing to Flash and did not required to stop microcontroller program execution during restore point creation, as it was while internal microcontroller Flash was used. Nevertheless restoring using external Flash memory required much higher energy than the use of internal Flash memory as microcontroller firstly had to copy the restoring data from external memory to internal, while using internal Flash microcontroller can access the required restoring data directly.

As it can be seen from presented data, for creating a restore point using the radio interface, the node consumed approximately 50% more energy than while using external Flash and approximately four times more than while using microcontroller internal Flash. Here it should be marked, that for the tested case all the required restoring data was send in one radio packet. Restoring using the radio connection required to have both transmission and reception phases (during testing the restore point version control was not used) which lead to energy consumption 100 times higher comparing with using the restoring point in microcontroller internal Flash.

Table I presents the values for energy consumption for active restoring operations, but usage of external memory chips would lead to system current consumption increase also during the low-power mode. Indeed, the current consumption of eZ430-RF2500 WSN node in low-power mode is around 1.2 μA [12], while the usage of M95512 EEPROM would increase it by 1 μA and usage of M48Z35 with GPIO expanders would demand around 7 μA additional consumed current. Although, this increase would not be significant for energy harvesting systems, that are unable to store the energy (e.g. [6,8]), but for the systems that can collect and store the energy for further usage (e.g. [5]) potential increase of low-power mode current consumption would be undesirable.

IV. Discussion and Conclusions

This work, to the best of our knowledge, is the first one that addresses a new problem of OS and user data restoring that become more important with the development of WSN with energy scavenging. In the article were described and evaluated different possibilities for WSN OS and user data after power-down restoring realization. The evaluation of different restoring techniques was made using REBOS – Restorable Event-Based Operation System, which was presented in the article. REBOS is the first OS, to the best of our knowledge, which is intended for use in WSN and which allows to automate OS state and user application data back-up procedures. Although presented results and methods are intended for usage in WSN with energy scavenging, they can be as well applied for WSN or other automotive systems with static power supply which require data restoring.

As it was shown in the article, contrary to widespread opinion, the data stored in microcontroller RAM is able to retain short power shut downs. From presented results it can be seen that after power supply shut downs up to several seconds OS can be efficiently restored using retained in RAM data. This method requires minimal amount of additional data processing and has minimal energy consumption which is around 20 times less than the minimum required for using non-volatile memory.

For the system restoring after longer power supply cut offs, the non-volatile memory, e.g. microcontroller internal Flash or external memory chips, should be used. As it was shown, the usage of internal microcontroller Flash during restoring allows minimizing energy consumption among all non-volatile memory types, but it requires higher supply voltage than minimal voltage required for WSN operation. The usage of external Flash memory allowed to reduce the required supply voltage, but increased the energy consumption during restore point creation and especially during restoring due to communication between microcontroller and memory chip. External NVRAM, although having low energy consumption for operations with memory itself, required very high additional energy for using GPIO expanders which are required for interfacing NVRAM chip with parallel pinout and WSN microcontroller. The usage of WSN radio interface for the system restoring, except increasing the network traffic and demanding the resources from other nodes, also lead to high energy consumption, especially during the node restoring.

As the continuation of this research, we plan to study the possibility of described restoring methods implementation for other used in WSN OS and to compare it with achieved using REBOS results, Additionally, the influence of restoring on the whole WSN operation requires further evaluation.

References

[1] C. Mathuna, T. O'Donnell, R. Martinez-Catala, J. Rohan and B. O'Flynn, "Energy scavenging for long-term deployable wireless sensor networks", Talanta, vol. 75-3, pp.613-623, May 2008.

[2] A. Hande, T.Polk, W. Walker and D. Bhatia, "Indoor solar energy harvesting for sensor network router nodes", Microprocessors and Microsystems, vol. 31-6, pp.420-432, September 2007.

[3] C. Knight , J. Davidson and S. Behrens, "Energy Options for Wireless Sensor Nodes", Sensors, vol. 8, pp. 8037-8066, December 2008.

[4] R.Morais, S. Matos, M. Fernandes, A.Valentea,S. Soares P. Ferreira and M. Reis,"Sun, wind and water flow as energy supply for small stationary data acquisition platforms", Computers and Electronics in Agriculture, vol. 6-2, pp.120-132, December 2008.

[5] eZ430-RF2500-SEH Solar Energy Harvesting Development Tool (SLAU273), Texas Instruments, Dallas, Texas, USA.

[6] TE-Power NODE datasheet, Micropelt GmbH, Freiburg, Germany.

[7] P. Mitcheson, E. Yeatman, G. Rao, A. Holmes and T. Green,"Energy Harvesting From Human and Machine Motion for Wireless Electronic Devices", in Proc. IEEE, vol. 96-9, pp.1457-1486, September 2008.

[8] Random Vibration Joule-Thief datasheet, AdaptivEnergy RLP, Hampton VA, USA.

[9] K. Mikhaylov and J. Tervonen, "Improvement of Energy Consumption for "Over-The-Air" Reprogramming in Wireless Sensor Networks", in Proc. ISWPC 2010, pp. 86-92, May 2010.

[10] K. Mikhaylov and J. Tervonen, "Optimization of Microcontroller Hardware Parameters for Wireless Sensor Network Node Power Consumption and Lifetime Improvement", in Proc. ICUMT 2010, Oct. 18-20 2010, pp. 1150-1156

[11] MSP430x22x2, MSP430x22x4 Mixed Signal Microcontroller, Texas Instruments, Dallas, Texas, USA.

[12] eZ430-RF2500 Development Tool User Guide (Rev. E) (SLAU227E), Texas Instruments, Dallas, Texas, USA.