Intelligent and passive RFID tag for Identification and Sensing

Authors:
Dipl El. Ing. FH Sven Keller
Michael Schweizer (student)
Prof. Dr. Marcel Meli,

Contact:
Prof. Dr. Marcel Meli,
ZHAW, Institute for embedded Systems
Technikumstrasse 9, Winterthur
CH-8400, Switzerland
Marcel.Meli@zhaw.ch,

Abstract

In this work, we present a design that can be used to emulate RFID tags, or to measure parameters and deliver the results in a form that is compatible with an RFID protocol. The data delivered can be an identity, a measured parameter, or a combination of both. Thanks to the use of a new ultra low power microcontroller, the power needs of the tag are kept low. The intelligence and flexibility associated with microcontrollers can be used for pre-processing of the sensor data, implementation of a protocol, or even an extension of the protocol to suit the application. The tag does not require the use of batteries, and is powered by the field of the RFID reader. After stating the challenges to overcome, we discuss the considerations that guided our design and the results of some preliminary tests.

1 Introduction

In the last years, there have been attempts to build sensors that can be read using an RFID reader and integrate them in RSN (RFID Sensor Networks) [5,6]. This opens up a new range of applications, especially in cases where it is impossible or not practical to use batteries as power sources, or in situations where size, power consumption or cost of a Wireless Sensor are not acceptable. In the case of RFID tags, cost, size and power consumption are kept low, thanks to the simplicity of these devices. They typically require a state machine that reads the contents of non-volatile memory and uses it to modulate its antenna load. The idea behind the RFID sensor is basically to add to the sate machine a part that will read the sensor data and insert it the tag ID that is sent to the reader. This keeps the device cheap, but also simple and low power, and somehow dedicated to its application. In order to open that field to many more sensors and to introduce flexibility in the collection and processing of the data, it is wished to use a microprocessor in the place of a simple hardwired state machine. This will basically deliver a Wireless Sensor without the complexity, cost and power needs of the associated radio transceiver. The solution will be a device between a classical Wireless Sensor and an RFID tag, with the advantage of a software programmable controller. The main problem in realising such a device is to reduce the power needs of the whole tag to levels that are acceptable and yet to have the needed elements to meet the protocol requirements. The recent design of the WISP [7,8] has shown that it is possible. The WISP relies on a low power MSP430 microprocessor, and can emulate a UHF RFID tag. In this work, we use a new microcontroller that requires less power than the MSP430, to build

an intelligent passive RFID tag. The platform should be the basis for the emulation of several tags. In this paper we show that the characteristics of the new device are good enough for such an application. We also show the result of some preliminary tests.

2 Motivation.

One of the main advantages of passive RFID tags is that they do not require an own power supply. They derive their energy from the electromagnetic field generated by the RFID reader. The amount of power harvested is dependent on several factors such as the power of the reader (needs to fit the local regulations), the geometry of the antenna, the distance of the tag to the reader, the type of field used.

For near field applications (magnetic field), the distance is usually very small (a few centimetres), and frequencies such as 125/134 KHz (LF), 13.56 MHz (HF) are used.

For important distances (several meters), the electric field is used, in frequency bands such as 868 MHz, 920 MHz. The data rate is higher than for LF and HF tags.

Tags differentiate themselves in the following ways:

- The type of modulation and coding scheme that they use
- The implementation of anti-collision schemes which gives the possibility to have several tags at the same time in the reader's field
- The type of memory they use to hold their data/identity, etc.
- The capacity to receive, interpret and respond to readers commands
- For special applications, there is a variety of other possibilities (such as dual frequency tags)

Very simple tags will just read their identity from a non-volatile memory and continuously sent it as soon as the reader is active (power on).

There are several reasons for emulating RFID tags. They might be related to the development, test or deployment of RFID tags, or to applications that require the integration of intelligent tags in RFID networks. In both cases, it is of interest to have a solution that can cover different protocols.

FPGA or CPLDs have been used to emulate tags or sub functions of tags. They are fast and the HDL design can be used as a basis for a tag IC. However, CPLDs/FPGA need a lot of energy, and therefore an own supply. For protocol development work this might not be a big issue, but it difficult to use them in tests or applications where passive tags are required. FPGA and CPLD also tend to be expensive and have an important footprint.

Low power microcontrollers on the other hand are very cheap (some of the popular 8-bit devices cost well under 1$ in quantity), but slower than FPGA/CPLD. By using them to emulate RFID tags, the following could be achieved:

- Emulation to test new algorithms on a larger scale
    - New algorithms that fit the processing resources can quickly be programmed and tested.
    - Tests with large number of tags are easier and cheaper (compared to an FPGA solution)

    - Complex crypto algorithms can be implemented, improving the security of some RFID applications.
- Sensors in RFID network
    - Flexible solution that will allow several type of sensors to be integrated in the application
    - Faster to develop
    - Possibility for software pre-processing of the measured parameters
- Use for intelligent pairing
- Implanted sensors (where no battery can go!) (medical applications, monitoring of bridge structures, ...)

Additionally, microcontrollers have a small footprint, and integrate FLASH memory which opens up the possibility of implementing a kind of software update. This is a very useful feature in development of protocols or applications and in the improvement of a product.

3 Microcontroller specifications.

The most important requirement in choosing hardware element for a passive tag emulator is low power. This can be translated directly into low voltage, low current, fast start up.

- Fast start up usually means that the microcontroller should be able to run on an internal oscillator (which also keeps power consumption low). Most popular microcontrollers have this feature. An internal oscillator can be calibrated when the device is manufactured, but its precision may have to be taken into account when fast data transfer is required.
- Low voltage
  Many microcontrollers will work down to 1.8 Volt. The lower, the better. This will make it easier to design the energy harvesting circuit and add range to the tag.
- Low current consumption
  Active, standby and power down current consumption are important, since the microprocessor may find itself cycling through these modes in the course of the tag emulation.
- Efficient instruction set
  This has to do with how much work can be done using a given amount of energy. It is important, and yet difficult to use for comparison, since processors have different architectures and thus different instruction sets. In case high level language is used, the efficiency of the compiler is also important.
- Good timer facilities to adapt to the many RFID protocols
- Analog peripherals for sensor interfacing.

For this work, we used a new microcontroller of the CoolRISC family, the EM6819. Information about this device from EM Marin can be found on the firm's website. Some of the features relevant to this work are mentioned below.

- Based on the CoolRISC core developed by the CSEM (Neuchâtel, Switzerland)
- Low voltage (can run from 3.3 Volts down to 0.9 Volt).
- Integrate a software controllable step up to generate the necessary voltage for external peripherals.

- Low current consumption (we typically measured less than 100uA at 1MIPS and 2V, room temperature). It is important to note that due to its inner working, the current consumption under 2V is higher that at 2 V, for the same CPU frequency.
- 4 Timers and several prescalers that can be used to determine the CPU frequency and control the peripherals.
- Harvard architecture with RISC like instruction set. Each instruction requires 2 clocks including branch operation (branch taken or not taken). (This is slightly changed in very special interrupt cases)
- Digital and analog peripherals such as SPI, ADC, comparator, Op Amp, to allow and simplify the interfacing of low cost sensors.
- Flash memory for fast development and updating of program, with EEPROM emulation.
- The device also sports several oscillator features (internal and external) and can run at up to 15 MHz in the temperature and voltage range (7.5 MIPS).
- On the fly change of the CPU speed is possible, and very useful for power saving.

The EM6819 is not the only device with a CoolRISC core. The firm Semtech (Nechâtel, Switzerland) also develop a class of high quality devices for instrumentation, associated with a CoolRISC core. Those devices are also suitable for low power, and some can run as low as 1V.

Near the CoolRISC, there are other microprocessors that could be used for this purpose. A first estimation based on the parameters in the datasheets and the measurements that we made showed that the EM6819 needs less energy. Timing with the CoolRISC is also more predictable, making it easier to write critical software parts. All this led us to give first priority to the EM6819 for this work.

4 Emulation procedures

The way the emulation is done is very important in saving power. There are in principle, 3 important parts.

- Receiving an interpreting commands from the reader
- Preparing the data
- Sending the data.

The sending of the data is always present, and the other 2 parts depend on the tag to emulate and on the application.

- For a very simple read only tag where one just needs to send the identity, the processor will only have to read a pre-coded identity and use it to modulate the antenna load. The data can be sent to a port pin, using immediate addressing. It will hardly be necessary to use a microprocessor just for that.
- A more realistic case is when sensor data are integrated in the data to send. Part of the identity can be kept, and the other part filled with the sensor data. Since the sensor data are not known in advance, the information need to be prepared, for instance by computing and adding the proper parity bits or CRC. Here also, the computing power needed to send the final data is directly related to the data rate. The time needed to prepare the data depends on the sensor type, the sampling rate and the pre-processing required by the application (eg. Averaging of A/D values). It is an

advantage if one can change the CPU speed on the fly. Going up when more processing is required, going down when there is less to do.
- In an even more complicated case, one might need to receive commands from the RFID reader, interpret and execute them. The CPU processing power should be such as to allow the proper reception of commands and their execution in the prescribed delays. Power consumption during the reception of commands will be related to the uplink data rate.

Depending on the frequency of operation and the precision required, one might have to work in synchronous or asynchronous mode.

Asynchronous mode: The CPU uses its own local oscillator to send the tag's data. It does not directly synchronise to the reader's clock. A sequence of operations might look like the following:

- Initialise ports and other peripherals
- Initialise CPU frequency to fit reader's data rate
- Initialise read pointer to read tag data
- Give out data read from data memory (do this until all the bits for the identity are given out)
    - Header, first block, second block, ..., CRC
- End of data block
    - Read sensor data
    - Insert sensor data in reserved part of data block
    - Compute parity
- Give out data

Advantage: No extra HW is needed for clock extraction.

Disadvantage: The phase difference that can lead tag and reader to be misaligned to a point where the data cannot be properly interpreted. Because of the short length of the RFID data and the relatively low data rate, this should not be a problem.

Synchronous mode: In this case, the CPU will wait for an edge of the clock signal sent by the microcontroller before sending a bit. The clock signal of the reader is isolated, and used to start an event or an interrupt. The associated routine will then give out the next bit, and then send the CPU in a mode where it consumes less power (than the active mode)

Hybrid mode: Here, synchronisation is done only done for some bits.

5 Some implementations, tests and results

We verified some of our concepts using different RFID devices. In this paper, we will limit ourselves to the emulation of a 125 KHz device (Near field) and a 868 MHz devices (Far field). The block diagram of the tag is shown in Fig. 1. The appropriate antenna has to be selected for the different frequency bands. A central element to deal with the intermittent nature of the energy provided by the RFID reader is the dual limit low power comparator. It allows a certain amount of energy to be accumulated before the microcontroller is started.

Once started, it is known how much energy is available, for the task at hand. The limits of the comparator can be set to take advantage of the low voltage feature of the microprocessor.

5.1 Emulation of a 125 KHz RFID tag.

The chosen device is the EM4102 from EM Marin. The device is used for animal tagging and industrial applications [1].

The EM4102 is a simple RFID device that can be manufactured in versions with Manchester, Biphase or PSK coding (mask option). The data speed can also optionally be set at manufacturing time. “The EM4102 contains 64 bits divided in five groups of information. 9 bits are used for the header, 10 row parity bits (P0-P9), 4 column parity bits (PC0-PC3), 40 data bits (D00-D93), and 1 stop bit set to logic 0.”

1	1	1	1	1	1	1	1	1	9 header bits
8 version bits or customer ID				D00	D01	D02	D03	P0	
				D10	D11	D12	D13	P1	
				D20	D21	D22	D23	P2	
32 data bits				D30	D31	D32	D33	P3	
				D40	D41	D42	D43	P4	
				D50	D51	D52	D53	P5	
				D60	D61	D62	D63	P6	
				D70	D71	D72	D73	P7	
				D80	D81	D82	D83	P8	
				D90	D91	D92	D93	P9	10 line parity bits
				PC0	PC1	PC2	PC3	S0	

4 column parity bits

Header, 9 bits	Data + parity(5 Bits)	Data + parity(5 Bits)	….	Data + parity(5 Bits)	Data + parity(5 Bits)
111111111	D00 … D03, P0	D10 … D13, P1	……	D90 … D93, P9	PC0 … PC3, S0

9 bits Header + 10 groups of 5 bits (data + parity) + 4 bits row parity, + S0(0) = 64 bits

Manchester Code

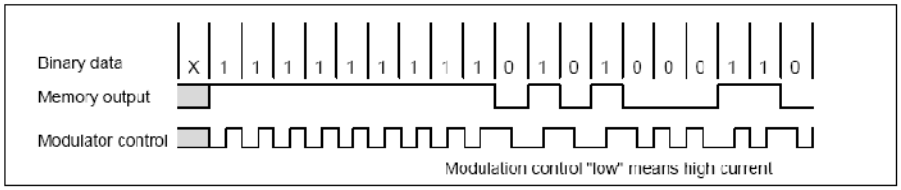


Fig. 6

Biphase Code

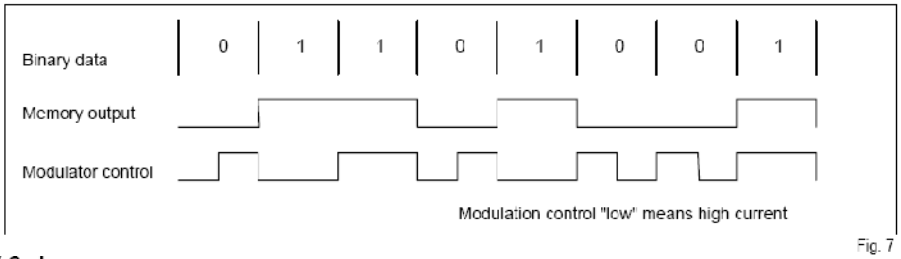
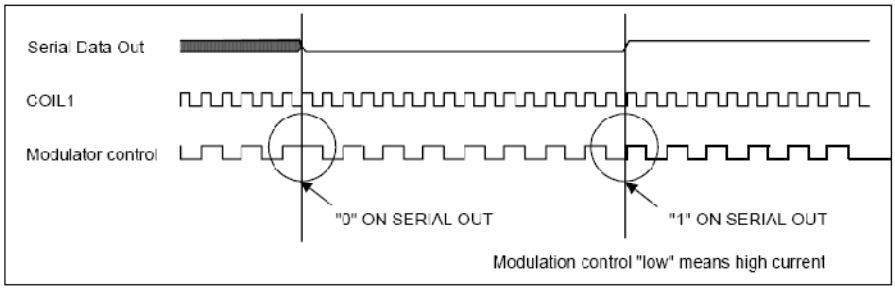


Fig. 7

PSK Code



By taking all the options into consideration, we arrived to the conclusions shown in the table below. To facilitate calculations, a reference frequency of 128 KHz is considered (instead of 125 KHz).

Coding	version	Clocks per bit	Data rate	Phases per bit(max)	Phases for 64 bits	Phase freq.	CPU clock
Manchester	A6	64	2 Kbit/S	2	128	4 KHz	16 KHz
Manchester	A5	32	4 Kbit/S	2	128	8 KHz	32 KHz
Biphase	B6	64	2 Kbit/S	2	128	4 KHz	16 KHz
Biphase	B5	32	4 Kbit/S	2	128	8 KHz	32 KHz
PSK	C4	16	8 Kbit/S	2	128	16 KHz	64 KHz

The emulation can be done by using the CPU to read the data in the ID array and writing it to a port pin. On this case, the following can be achieved.

- By efficiently using the instruction set of the CoolRISC, it is possible to read data from RAM using indirect addressing, and write it to the needed port. This will require 2 instructions, thus 4 CPU clocks for each phase. For 2 phases we have 8 CPU clocks. A bit can thus be read and transferred to the RFID reader using 8 CPU clocks.
- For PSK encoding, this means that a CPU clock of 64KHz will be enough.
- For Manchester or Biphase encoding and 64 clocks per bit, the CPU could run at 16 KHz.

- In case of Manchester or Biphase encoding and 32 clocks per bit, the CPU could run at 32 KHz.

Because of the need to check the end of the array, extra instructions are required, and the clock was doubled in each case.

At the frequencies mentioned above, the microcontroller will consume less that 20UA, meaning a factor of about 10 compared to the consumption of the EM4201. By using optimisation and taking advantage of some of the digital blocks, one can reduce the CPU clock and thus the current consumption.

In one of the simplest program version, the device just gives out a code loaded in its volatile data memory. The use of the volatile memory allows sensor data to be integrated in part of the Tag ID, and sent together to the reader. This was done in a second step, using the on chip temperature sensor. The device temperature was regularly measured, the corresponding parity calculated, and a new data set generated. The set was then sent to the RFID reader.

As reader, the EM4095 kit from EM Marin was used to generate the RFID field, decode the data from the tag and display the read ID on a PC.

It was possible to read the tag ID added with the integrated sensor data, and display them on the PC screen.

5.2 Emulation of 900 MHz RFID Tag.

The emulated Tag is the EM2222 from EM Marin. Some information about that device cannot be publicly made available at the moment. The following features can be mentionned:

- Factory programmed 64 bit ID number
- High data rate: Up to 256 kbit/s
- Frequency independent: Typically used at 869 MHz, 902 - 960 MHz (versions 001 to 099),
- 2.45 GHz (versions 101 to 199)
- On-chip oscillator
- On-chip rectifier

The data stream consists of a preamble, ID data, CRC for a total of 96 bits. Using the same strategy as described for the 125 KHz tag, we could emulate this device with a CPU frequency of 1 MHz (500 KIPS). In this case, data is also kept in volatile memory to allow the integration of sensor data. The CPU speed can be reduced by 2 if the data is hard coded. We calculated that a further reduction by 2 is possible by using some optimisation techniques, leading to a CPU speed of 250 KHz. The tolerance of the CPU clock matches that of the local oscillator of the emulated tag.

We implemented a first algorithm running at 1 MHz, and could emulate the tag in continuous mode at up to 10 cm (directly supplied). Using the dual limit comparator to control the power buffering, (high limit at 2.6V) we could achieve over 50 cm read range. The reader used is a demonstration reader from EM Marin (EMDB412 with SkyeModule M9 UHF reader). Reader power was set at 27dBm. We used a monopole antenna for the tag and 5 stage multiplier. The antenna and the energy harvesting stage are not yet optimized for good power transfer.

Conclusions and thanks

It is possible to emulate different tags using the new low power microprocessor, from LF to UHF. The appropriate front-end for energy harvesting and data and clock shaping must be provided. The resources of the Microcontroller are appropriate to emulate at least parts of complex protocols.

We will further develop the platform to optimise the energy harvesting and extend the rage of the UHF tags. We will also develop the layer needed for more popular protocols such as Gen2.

Thanks to EM Marin for providing several devices and of the UHF RFID reader.

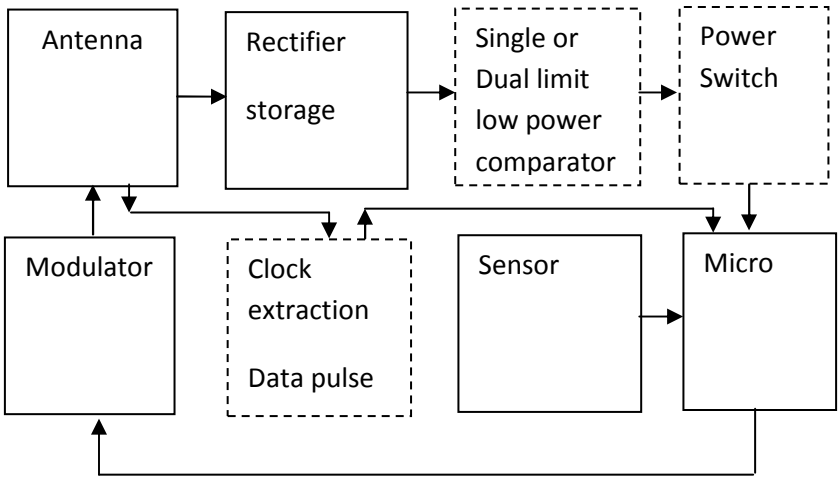


Fig.1 Block diagram of the passive tag

References:

[1] Datasheet EM4102, www.emmarin.ch

[2] Datasheet EM6819, www.emmarin.ch

[3] Factsheet EM4222, www.emmarin.ch

[4] Datasheet EM4223, www.emmarin.ch

[5] Self-powered wireless temperature sensors exploit RFID technology
K. Opasjumruskit et al. Pervasive IEEE Volume 5, Issue 1, Jan.-March 2006

[6] Oki Electric Wiselink Sensor

[7] A Wirelessly-Powered Platform for Sensing and Computation.
Joshua R. Smith et al. Ubicomp 2006

[8] Revisiting Smart Dust with RFID Sensor Networks
Michael Buettner et al Seventh ACM Workshop on Hot Topics in Networks (HotNets-VII), Alberta, Canada, Oct 6-7 2008.