Firmware Protection and Attacks Against the ATMega Microcontroller Series

Dionisio Perez-Mavrogenis

March 7, 2014

Abstract

Give paper topic, objectives and conclusions reached.

1 Introduction

mention abbreviation mcu=microcontroller unit.

1.1 Problem Statement

Perhaps need to classify attackers : e.g. home hacker << (semi-)professional crackers << funded organisations. also give potential problems(financial and security implications) with stolen software (IP theft etc), firmware tampering etc. Also separate microcontrollers (designed for high security or ordinary use)

1.2 Objectives of Paper

- Review the ATmega series (architecture, applications and security features)
- Review current attacks against MCUs in general
- Review Countermeasure to these attacks or how to make them more difficult
- Review attacks applicable to the ATmega (give the easiest possible attack applicable)
- Describe a way to make the mega more secure
- Evaluate if it's worth going the extra mile for security
- Have an overview of the paper in the conclusion

Give paper structure at some point.

2 The ATMega MCU Series

2.1 AVR Predecessors to the ATmega Series

apparently sergei has cracked them already maybe talk about support(libraries/tools) here ?? [3]

2.2 ATMega Architecture and Features

compiler is AVR-GCC (library is avr-libc) and has own libraries for interfacing with AVR

By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip,the Atmel ATmega644 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.The ATmega644 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2.1 Background and General Features

The ATmega series of MCUs is a relatively large family of microntroller units. However this paper focuses only on the ATmega644 and ATmega1284, [reasons for focus here]. The only difference between then 1284 and the 644 is that the 1284 has got more memory available and a summary of (some) of the features of the two units is give in Table 1[[1],[2]].

Both MCUs are an enhanced RISC Harvard architecture 8-bit CPU. Figure 2 shows the conceptual difference between a Von Neuman (most modern PCs) and a Harvard architecture, where the key distinction lies in the separation of application code and program data into different memory sections (Harvard) and tasking the CPU with distinguishing between code and data that lives in the same memory region (Von Neuman). The 644/1284 implement a Harvard architecture for both power and computational efficiency, as they are able to access more than one registers simultaneously (due to the physical wiring of the CPU) and hence are able to execute an instruction per cycle as once an instruction is executing the next one is pre-fetched and decoded. Their operating voltages can vary between 1.8V and 5.5V (maximum operating frequency 20 MHz).

2.2.2 Memory Organisation

The 644/1284 are equipped with an EEPROM, flash memory, SRAM, a large number of general purpose registers and a large number of I/O registers (in order to be able to perform I/O) and all memory (including I/O memory mapped images) is linear, i.e. it follows the flat memory model.

The flash memory is separated into two regions, the bootloader section and application code section. The boundary between the two sections can be configured by programming the appropriate fuses, and the page size can also be configured that way as well. Both sections hold code, however code residing in the bootloader section can execute a special instruction (`SPM`[1]) which allows the bootloader code to write to *any* section in the flash memory and hence possibly modify itself(designed for purposes such as firmware upgrades). The bootloader code can be triggered by a direct jump from the application section or by programming the reset vector via the reset fuse to point to the appropriate section of the bootloader code.

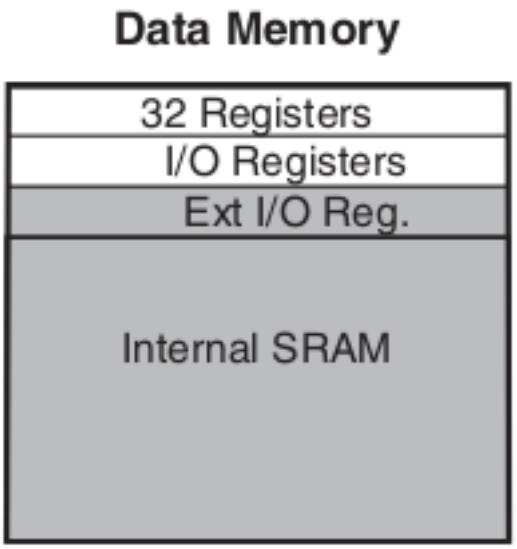


Figure 1: SRAM layout for the ATmega 644 and 1284.

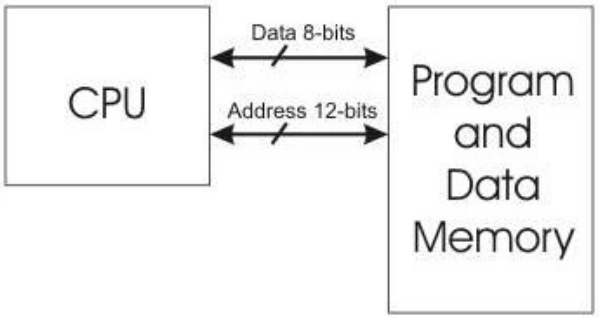
The EEPROM is memory for data that needs to persist between reboots of the MCU and hence it is (widely) used to hold configuration variables and other non-temporary preferences the application code (or the bootloader) may need, having an average lifespan is 100,000 write cycles per page.

The SRAM is volatile storage and is used as the stack and heap for the software (either application code or bootloader code) as well as for storing the Register File (i.e. the 32 GP registers) I/O and Extended I/O Memory. The reserved register locations exist in order to support the use of peripheral units as well as hold program status information (e.g. the Stack Pointer can be found in one of the GP registers).Figure 1 gives an overview of the SRAM hierarchy, which is slightly different (in terms of region sizes) for the 644 and the 1284 as the 1284 offers more SRAM.

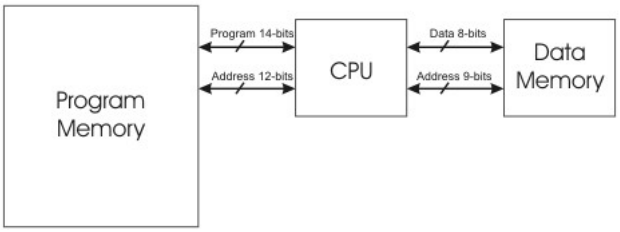
[1]SPM = Store Program Code

Model	EEPROM (Kb)	SRAM (Kb)	Programmable Flash(Kb)	GP Registers	SPI Port	JTAG Interface
ATmega644	2	4	64	32	Yes	Yes
ATmega1284	4	16	128	32	Yes	Yes

Table 1: Specification overview for the AVR ATmega644 and ATmega¡¿



(a) Schematic of a Von Neuman architecture.



(b) Schematic of a Harvard architecture.

Figure 2: A comparison of different machine architectures.

2.3 ATMega Security Features

The AVR ATmega644/1284, even though they are not meant to be trusted or secure hardware, posses certain security features. In particular, each board provides six Lock bits (which can be programmed or unprogrammed) and which are responsible for controlling or preventing different memory portions of the board to be modified or read by the other parts (e.g. prevent code executing from the bootloader section to read/write the application code section via the `SPM` instruction). The prevention however is not permanent, as that would limit the usefulness of the MCU and therefore one has the option to bring these lock bits back to State 1 (i.e. unprogrammed, having no protection scheme enabled) by issuing a Chip Erase command, which has the effect of completely erasing the Flash, EEPROM and Lock bits.

The erasing is performed with the sequence of events presented above and this is important, as one does not want to remove the access protection before removing all sensitive data and hence the Lock bits are set to 1 only after the whole program memory has been erased. Even though the flash memory has an average lifespan of 10,000 write cycles (as well as programming being relatively expensive as an operation) this approach makes sense as the ultimate goal is to preserve the intellectual property on the board rather than the board itself.

Table 2 provides an outline of the available Lock bits provided by the ATmega series. The functionality of the BLB1 group is to control access and modification of the bootloader section, group BLB0 bits control access to the application code section and group LB bits are responsible for controlling modifications on the EEPROM and Flash.A detailed explanation of their functionality and how to use them is given in [2] and [1].

Lock Bit Byte	Bit Number	Description	Default Value
BLB12	5	Boot Lock Bit	1
BLB11	4	Boot Lock Bit	1
BLB02	3	Boot Lock Bit	1
BLB01	2	Boot Lock Bit	1
LB2	2	Lock Bit	1
LB1	1	Lock Bit	1

Table 2: Security lock bits offered by the ATmega644 and ATmega1284.

3 Current Attacks

3.1 Introduction

mention attacks may be passive(observing input->output mapping) or active (tamper with the aforementioned mapping in some useful way)

3.2 Non-Invasive Attacks

3.3 Semi-Invasive Attacks

all memory types are linear(as well as memory mapped IO) - related to memory scanning attacks by glitching and power faults (stop making call or jump instructions)

3.4 Invasive Attacks

overview of attack categories [each one to the category that it corresponds above]

- microprobing
- side-channel attacks
- software attacks (exploit communication protocols or crypto implementation and such)
- reverse engineering of hardware
- fault generation (power/clock glitches)

* for each category discuss budget/tools/skillset/time required

4 Countermeasures to known attacks

- overview of most popular techniques
- benefits and how they improve the situation/approach the problem
- added cost for this investment (in terms of hardware and money, transparency to the developers, runtime overhead etc)

perhaps review some popular secure chips ??

5 Securing the mega

5.1 Working attacks against the ATmega

5.2 Motivation

5.3 Current Attack Vectors

5.4 Protective Steps

* feasible? * added cost (in terms of $$, extra hardware and software implementation penalties/overhead)

6 Evaluation

not sure if the subsections are needed here

6.1 Attacks and Solutions overview

6.2 Conclusions

References

[1] AtmelCorporation. *Atmel ATmega644 data sheet*, 2012. URL `http://www.atmel.com/Images/doc2593.pdf`.

[2] AtmelCorporation. *Atmel ATmega1284 data sheet*, 2013. URL `http://www.atmel.com/Images/Atmel-8272-8-bit-AVR-microcontroller-ATmega164A_PA-324A_PA-644A_PA-1284_P_datasheet.pdf`.

[3] Dr. Sergei Skorobogatov. Breaking copy protection in microcontrollers, 2000. URL `http://www.cl.cam.ac.uk/~sps32/mcu_lock.html`.