Chapter 3
Microprocessors and Microcontrollers Security

Chris Shire

Abstract This chapter will consider the chip architectures used in embedded security; how they have evolved over the past three decades, the current designs, and the future trends. The chapter will consider the evolution of the microcontroller Central Processing Units (CPU) cores such as the 8051, 6805. It will look at the wide range of innovative and reduced instruction set designs, including popular off-the-shelf microcontroller designs, microprocessors, and digital signal processors. It will also consider other reduced instruction set designs, with reference to known attacks and options for protection. It will look at the vulnerability of functions within the chips such as memories and interfaces, and possible enhancements. Further security measures for different memory types will be reviewed. Enhanced security concepts using defensive designs, anti-tampering measures, and other hardware protection are discussed.

3.1 Microcontrollers and Microprocessors Security Needs

From an abstract perspective there is little difference in the function of a microcontroller and microprocessor, and in embedded applications the implementation becomes blurred as to the outside world the computing device in the system is often literally a "black box". This chapter will use the euphemism of "Embedded CPU" to cover all the options in design and integration, unless discussing a specific nuance of a design. This is because the designers or test engineers of the system are the only people likely to appreciate the difference. There are several misconceptions about the security of embedded systems. First that attacking the Embedded CPU is difficult, because it is often deep inside a complete assembly. Second there is little value in the embedded software or intellectual property. Finally, that people lack the motivation

C. Shire (✉)
Infineon Technologies UK, Bristol, United Kingdom
e-mail: chris.shire@infineon.com

K. Markantonakis and K. Mayes (eds.), *Secure Smart Embedded Devices, Platforms and Applications*, DOI: 10.1007/978-1-4614-7915-4_3,

to attack an embedded control system. Attacking a single smart meter and turning off the lights in one house seems trivial, it is only when this attack could be scaled up to a city may people worry. As a result many people consider an embedded control system is secure, because it has never been attacked. Justifying embedded security is often a major issue. In addition by definition to embed an item is "to fix something firmly in a surface or object"[1] so that it is an integral part of a larger system. It is therefore perceived that as an Embedded CPU cannot easily be physically removed without damaging the system so it is not open to abuse and must be secure. While from a physical point of view this may often be the case, it does not imply the device is secure electronically.

The first issue to clearly determine is why the Embedded CPU might be attacked, as this will likely determine the method of attack. When a thief steals intellectual property, e.g. software or chip design, then destruction of the system may be of little consequence. Alternatively the attacker may want to deny the use of the system to others for a period of time so as to gain profit directly or indirectly. The physical location of the system will have a bearing on the method of attack. If a system can be accessed remotely and even better covertly, then the potential for attack is greater. A system with strong physical security which is difficult or expensive to breach, such as the cashbox of an arcade gaming machine might be attacked via their test port. Not properly understanding the complete system's security mechanisms and potential attacker's methods can render any Embedded CPU security useless. Of course the more money spent on a security mechanism is in theory the better but there may be consequences. One example is an intrinsically valuable object, such as a royal seal stamp, could be stored in a very secure vault, but if fast frequent access is required such security might impede the signing of official documents. There has to be a balance of risk versus performance.

The detailed methods of attack are described in another chapter, but for an Embedded CPU in an enclosed system the physical connections to the outside world are often the weakest links especially those used for manufacturing tests, e.g. a JTAG (the industry standard Joint Test Action Group serial interface on many CPUs), remote programming, and peripheral connections, e.g. USB, Ethernet, etc. From a physical point of view the housing of the system can include anti-tamper mechanisms, conformal coatings, or epoxy encapsulation. It may even have some countermeasures such as one-way screws or include some "security by obscurity" such as the deletion of product identifiers on the Embedded CPU chip packages. Beyond the physical assembly the electronic architecture of the Embedded CPU and its associated components should be considered in any security analysis. To understand the strengths and weakness of an Embedded CPU it is worth considering the historical development of common architectures.

[1] Definition from Macmillan Dictionary http://www.macmillandictionary.com/thesaurus/british/embed#embed_4

3.2 Historical Development

Embedded CPUs can be broken into two broad categories: microprocessors with various peripheral components and microcontrollers which have most of its memory and peripherals on chip, thereby reducing cost and size and often designed for dedicated applications. In contrast to the personal computer and server markets, a large number of basic CPU architectures are used today; these are Von Neumann as well as various degrees of Harvard structures, Reduced Instruction Set Computer (RISC) as well as non-RISC and Very Long Instruction Word (VLIW); word lengths vary from 4 bit to 64 bits and beyond, mainly in Digital Signal Processors (DSPs) although the most typical remain 8/16 bit. Most architectures have a large number of different variants and formats, the most popular of which are manufactured by several different semiconductor companies.

Some common architectures are or were denoted by the following code numbers: 65816, 65C02, 68HCxx, 68K, 78K, 8051, 80251, ARM, C167, ColdFire, COP8, H8, MIPS, MSP430, PIC, PowerPC, SHARC, SPARC, ST6-ST32, TriCore, V850, x86, Z8-Z8000. Information on any of these architectures can be found from their entry in an Internet search engine. It can be seen that this multitude of different designs may present an obstacle to any attacker by simple obfuscation of what CPU is used in an embedded system. However, the underlying format of any design and therefore its weaknesses can be traced back to the basics of logic designs.

Since Integrated Circuit (IC) logic designs started to include arithmetic processing units with software programming, the potential for misuse either accidently or deliberately has been an issue. The earliest CPU's in the 1970's were made from off-the-shelf components such as Bit Slice Processors (BSP). BSP's used arithmetic logic units (ALUs) that typically came in 4 bit increments. These could be assembled together to make larger word lengths (8 bit, 16 bit, etc.) and were controlled using Programmable Read Only Memories (PROM). From these early beginnings developed the one chip microcontroller solutions found in every type of electronic product today, to the multi-core processors powering laptops, servers, and games machines. A direct descendant of the BSP is the Digital Signal Processor (DSP).

In 1971 Intel released its first real microprocessor, the 4004 and with it, the era of microcomputers began [1]. Microcontrollers which included some form of on chip memory first appeared in 1974 with the Texas Instruments TMS1000 with 1K Byte of masked ROM and 64 $\times$ 4 bits of RAM for use originally in calculators[2]. In 1976, Intel introduced one of its earliest microcontrollers, the 8748 and 8048. They were used extensively for computer keyboards, or programmed to perform certain data conversion operations. Other Embedded CPUs were programmed with specific stand-alone functions such as a calculator. From the 8048 came the 8051 and later the 16 bit version the 80251. These designs were extensively licensed to over 20 semiconductor suppliers both as embedded cores and later synthesisable software cores. To date it is estimated that over 5 billion embedded designs have been based on the 8051 and its derivatives, mostly in smart cards.

[2] http://www.ti.com/corp/docs/company/history/timeline/semicon/1970/docs/74tms_1000.htm

From the start two issues regarding the reliability and security of systems became evident, one internal, the other external. The internal problems were found either when initially testing devices to ensure correct operation as construction faults within multi-chip systems or faults in the PROM could lead to misbehaviour. These faults could appear later in the field, due to the semiconductor process impurities or because of the environment. Excess vibration, voltage, or electrostatic charge, either applied by accident or on purpose, might damage the devices. The other threat was to the intellectual property of the design. Clones from state-run semiconductor companies in the USSR and other eastern bloc countries appeared within a few years[3]. The need for clones was driven by the block on export of high technology by the USA and the need to produce computers for military and commercial applications. The only electrical protection at this time was often in the form of external devices that would protect the Embedded CPU from electrical damage. Hardware protection consisted of strong epoxy encapsulations to deter intruders and to ensure the component assembly stayed together when vibrated.

From the beginning there was also seen a need to protect the software in the ROM of a Embedded CPU as it represented the results of expensive software development and sometimes key intellectual property. The masked ROM of the TMS1000 protected the code of the developer by making the command to read out the ROM nominally inactive. These devices were used in simple games machines and so became, at least as a hobby, the target of attack to allow people to sell cloned or modified games. Various articles still exist (on illicit hacking websites) on how this ROM might be read out using various hardware test functions.

3.3 The Microprocessor

The microprocessor is the portion of a computer system that carries out the instructions of a computer program. This term has been in use in the computer industry at least since the early 1960s. The form, design and implementation of microprocessors have changed dramatically since the earliest examples, but their fundamental operation remains much the same.

Early microprocessors were custom designed from logic circuits as a part of a larger computer. However this has given way to the development of standard mass-produced microprocessors. This trend generally began in the era of discrete transistor mainframes and minicomputers and has accelerated with the popularity of the Integrated Circuit (IC) and the underlying technology trend sometimes known as Moore's Law [2]. Semiconductor technology has allowed increasingly complex CPUs to be designed and manufactured to tolerances in the order of a few tens of nanometers.

Early single chip CPUs supported 8 bit data and address buses such as the Intel 8080 or 16 bit as was TI's TMS 9900. There are extensive references to the history of CPU developments [3], but with regard to embedded systems there are a few

[3] http://www.cpucollection.ca/Russian_and_ussr.htm

significant steps which have driven this technology. Western Design Center Inc. introduced the Complementary Metal Oxide Semiconductor (CMOS) 65816 a 16 bit upgrade of the WDC CMOS 65C02 in 1984. This was the core of the Apple II and later the Super Nintendo Entertainment System, making it one of the first and most popular 16 bit embedded designs of all time. Intel followed a different path, and upgraded their 4 bit 4004 designs to the 8 bit 8080t and eventually the 16 bit Intel 8086, the first member of the x 86 families. Their derivatives were used in a number of embedded systems. Intel introduced the 8086 as a cost effective way of porting software from 8080 code. The 8088, a version of the 8086 that used an external 8 bit data bus, was the microprocessor in the first IBM PC, the model 5150, but also used for several early embedded applications. Following up their 8086 and 8088, Intel released the 16 bit 80186, 80286. The 8086 and 80186 had a crude method of memory segmentation, while the 80286 introduced a full-featured segmented memory management unit (MMU) which could be used to protect access to some software. The Intel family became the target of various clone manufacturers, both legitimate licensees and reverse engineered chip suppliers. This lead to several litigious incidents around chip design and intellectual property theft [4]. However it also lead to the acceptance that chip design needed further security to, if not stop, at least deter such issues in the future.

3.3.1 32 Bit Microprocessor Designs

16 bit designs had only been on the market briefly when 32 bit implementations started to appear. The most significant of the 32 bit designs is the Motorola MC68000, introduced in 1979. The 68 K, as it was widely known, had 32 bit registers, but used 16 bit internal data paths and a 16 bit external data bus to reduce pin count, and supported only 24 bit addresses. Motorola described it as a 16 bit processor, though it clearly has a 32 bit architecture. The combination of high performance, a large 16 MByte memory space, and low cost made it the most popular CPU design of its class. The Apple Macintosh designs made use of the 68000, as did a host of other designs in the mid-1980s and its derivatives such as the 68020 ever since. Other large companies designed the 68020 and its derivatives (such the 68EC020 with reduced 24 bit addressing) into embedded equipment such as laser printers Today's ColdFire [20] processor cores are derivatives of the respected 68020. The 68000 had several clones again mostly legitimate.

From 1985 to 2003, the 32 bit Intel x86 architectures became increasingly dominant in desktop, laptop and server markets and these microprocessors became faster and more capable. In 1994 Intel introduced its Smart Die™ program for its 386, 486, and Pentium products so it could supply CPU cores and peripherals for multi-chip modules, for use in embedded computers such as hand held terminals and communication equipment.

3.3.2 64 Bit Microprocessor Designs

While 64 bit microprocessor designs have been in use in several markets since the early 1990s, the early 2000s saw the introduction of 64 bit microprocessors targeted at the PC market. One example is the PowerPC a RISC architecture created by the 1991 Apple–IBM–Motorola alliance [5], also known as AIM. Derivations of this design are now found in high end embedded systems in the network communications and automotive engine management units. Several derivatives have been developed as cores for embedding in a Field Programmable Gate Array (FPGA) by various suppliers such as Altera, LSI logic, Lattice and Xilinx and as cores for various Apple products. There are a multitude of Linux-related operating systems developed with this platform for embedded applications.

Overall in the past decade or more as world trade became more open and the semiconductor technology became more complex the benefit from cloning complex CPU's became less economic. The so called "Grey" market for unofficial sales moved to remarking or repackaging lower specification original devices as high spec units. This practice continues to this day. In the past 10 years, CPU designers have now started to include hardware security functions, ranging from serial numbers to dedicated encryption engines to ensure that users can verify on-line the source of the device. This technique is effectively two factor authentication. The first step is to use the serial number or credential, which can include a check sum, to verify that it is a valid formatted serial number or credential. This will not stop copied hardware serial numbers. The second step is for a hash function of data using this identity to be verified by a remote trusted third party. This then ensures that clones cannot be active in a population of connected devices. It does not solve the problem of cloning for embedded devices with little or no remote connectivity. In these circumstances the Embedded CPU's architecture has to have further security protection integrated at the start and to be continuously active when operational. This may include hidden unique properties programmed into each individual Embedded CPU that cannot be cloned, or a public/private key pair generated on chip that can be challenged to test for authenticity of the device.

3.3.3 RISCs and ARM

A microprocessor is a general purpose product that may have many commands both logical and arithmetic to allow easy programming. Several specialised processing derivatives have followed from this basic concept. A digital signal processor (DSP) has limited instructions but often with a very large data bus to allow for high data throughput. Graphics processing units similarly in the past had limited or no general programming facilities. However, ever since the first "general purpose" CPUs were developed there has been a demand for high performance, dedicated functionality, low cost designs; so-called RISC Machines. By implementing fewer instructions,

the chip designer is able to dedicate some of the precious silicon real-estate for performance enhancing features. In addition the benefits of RISC design simplicity are a smaller chip, smaller pin count, and low power consumption. Among some of the typical features of a RISC processor are a Harvard architecture which allows simultaneous access to all the memory by having separate buses for instructions and data. The overlapping of some operations increases processing performance. Probably the most popular RISC family today is the Acorn RISC Machine (ARM) architecture which first appeared in 1985. It has since come to dominate the 32 bit embedded systems processor space due to its efficiency, the low cost licensing model, and its wide selection of system development tools. Many mobile phones include an ARM processor, as do a wide variety of other embedded products. There are microcontroller-oriented ARM cores without virtual memory support, as well as multi-core processors with virtual memory. It has been estimated that by 2011 that over 25 billion ARM cores will have been shipped [6], the vast majority into embedded systems. ARM has been licensed to over 60 commercial companies, including nearly all major IC manufacturers, and several other institutions. Only a few vendors are licensed to modify the ARM cores. This approach has lead to common design and layout, without security features making it an easy target for attack. So derivatives with security functions were requested by the smart card industry. In 1999, ARM announced it was looking at derivatives incorporating security features, called SecureCore, the first public implementation was with Samsung in 2001 with the SC100 core [21]. This core besides being a fully synthesisable design, offered randomised layout options, secure debugging, controlled development to stop reverse synthesis, plus memory protection features and anti-Differential Power Analysis (DPA) functions. From this has come the Cortex-M series of microcontrollers including the special secure core M3 (SC300) range and the ARM company has allowed further modification by at least one vendor. The SC300 has become the preferred architecture for most of the major smart card IC vendors, with various different extra security features. It provides a relatively common platform for the major software developers, i.e. the smart card vendors. This allows at least some software portability from one IC vendor to another, which has been a major hurdle to the industry in the past. In addition second sources silicon suppliers for high volume smart card designs can be provided quickly and less expensively. These derivatives can be considered for assessment to Common Criteria, with certification to EAL5 High and potentially up to EAL6 High.

In 2003, ARM announced the introductions of security extensions for its microprocessor range, marketed as TrustZone® [7] first for its ARM1176 and later found in the Cortex A5 to A15 CPU range. It provides a low cost alternative to adding an additional dedicated security core to a System-on-Chip, by providing two virtual processors backed by hardware-based access control, including a 33rd bit to identify secure commands. It offers a combination of MMU and caches uses tables to determine whether a particular level of memory exists in “Secure Mode” or “Non-secure Mode”. This enables the application core to switch between two states, referred to as “worlds” (to reduce confusion with other names for domains), in order to prevent information from leaking from the more trusted domain to the less trusted domain. This domain switch is generally orthogonal to all other capabilities of the processor,

thus each domain can operate independently of the other while using the same core. Memory and peripherals are then made aware of the operating domain of the core and may use this to provide access control to secrets and code on the device. Over 20 companies have taken licenses for the TrustZone® extensions, and variants have been used in applications as varied as MP3 players, smart-phones and payment terminals. The implementation the TrustZone® extensions are specific to each design, and some may include other hardware security features. However, it would be difficult to apply any sort of formal security certification on a device which includes both secure and insecure functions on the same chip.

3.4 Security Design of Embedded CPU Architectures

Today Embedded CPUs are at the heart of a huge range of commercial and industrial equipment [22], including domestic appliances such as microwaves, DVD players and televisions. They are used in cars for engine-control and service functions, in medical instruments, and in many other areas. The widespread availability of Embedded CPUs is a measure of their flexibility and cost when compared to a dedicated hardware function. Usually, they have a high level of input and output (I/O) device options including serial interfaces, e.g. SPI Serial Peripheral Bus (SPI), Control Area Network (CAN), Universal Serial Bus (USB), general use I/O and other interfaces. A microcontroller may minimise the number of external devices used in the system by integrating much of the external interfacing to analog signals as many of them have built-in analog-to-digital (ADC) and digital-to-analog converters (DAC), comparators and pulse width modulators (PWM).

Early Embedded CPUs had 4 bit or 8 bit internal data buses, while modern microcontrollers have 16 bit or 32 bit data buses to access external memory. Obviously, the wider the data bus, the more difficult it is to micro-probe it and reverse engineer. While the complexity, size, construction and general form of Embedded CPUs have changed drastically over the past 40 years. It is notable that the basic design and function has not changed much at all. Most modern Embedded CPUs can be described as von Neumann stored-program machines, but a few do have a Harvard architecture. The first architecture uses separate memory for instructions and data, while the latter uses a single memory structure to hold both instructions and data. From the security point of view, the Harvard architecture should offer better protection against micro-probing attacks. When attacking a von Neumann architecture, an attacker could interrupt the CPU so that it will no longer execute branch instructions or fetch instructions. The contents of the memory can be revealed by micro-probing the data bus and storing the signals. The same attack when applied to a Harvard microcontroller might reveal only the program code, whereas the data memory, which usually contains passwords and decryption keys, may not be available. If a RISC design is too simple with few instructions, it might be easier to reverse engineer it. A RISC with a complex instruction set may leave more distinctive power traces making identification of each instruction through power analysis easier.

Some Embedded CPUs derivatives have core designs with speed enhancement features. One is an instruction pipeline. Each instruction is divided into some simple subinstructions, which are executed by the CPU in step, with a pipeline controller watching the process. Hence, the Embedded CPU will not be immediately execute the code, instead it executes two or more instructions simultaneously. This makes the power analysis more difficult, because two or more instructions can contribute to the power trace. Some secure Embedded CPUs may have one or more slave crypto-coprocessors [8]. These support encryption and related processing of either various symmetric algorithms such as AES, 3DES or asymmetric algorithms such as RSA or ECC. These firmware coded coprocessors typically provide hardware acceleration of a range of functions such as multiple XORs, Galois Field multiplication/addition or modulo multiplication while storing the intermediate results in a dedicated memory of the crypto-coprocessor (SRAM). Such devices have numerous protection features that prevent unauthorised analysis of their data, or reverse engineering. A crypto-coprocessor block may include functions such as a random number generator in order to house them in the same protective environment as the encryption function. If the Embedded CPU is to be certified such random number generators have to use bit sources with a high level of entropy, for example based on two independent free running oscillators. Another common feature in many Embedded CPUs is a cache memory that stores instructions and data that requires frequent and fast access. For example, if the Embedded CPU executes a loop, then the instructions will be fetched from the cache memory rather than from the external memory thus saving time. This makes micro-probing attacks harder, as some data will not appear on the external data bus.

The demand for hardware security began with embedded systems for consumer products. Thirty years ago there was almost no protection against cloning of such devices except legal and economic forces. Often Application-Specific Integrated Circuits (ASICs) were widely used. Such ASICs were simple state machines replacing discrete logic components, thus reducing the size of the assembly and at the same time protecting against competitors with less integrated designs incurring more cost and larger solutions. These ASICs did not carry much security. Their functionality could be determined by a simple analysis of the signals using an oscilloscope or doing an exhaustive pattern analysis of their inputs and outputs. For example as the consumer demand for a clock in every room grew, digital clock ICs were heavily cloned. From the late 1970s, microcontrollers offered a very good replacement for ASIC-based designs. They not only had internal memory and useful interfaces such as LCD drivers, but some sort of security protection against unauthorised access to the internal memory contents. Unfortunately, early microcontroller's semiconductor technology did not offer non-volatile storage for large programmes or variable data, so this had to be stored in a separate chip outside the microcontroller thus allowing the attacker to access them. Games machines had ROMs made with low-cost mask technology allowing easy reverse engineering their contents. Replication of the design could involve using a microcontroller with EPROMs, which although expensive, it was economically viable if the games machine was very popular. This trend

continues today as even recently news of attacks on dongles used for "software protection" in the consumer games market has been published [9].

The next step in security design was to place an Electrically Erasable Programmable Read Only Memory (EEPROM) data storage chip next to the microcontroller inside the same plastic package or on the same die. To attack such a chip is not easy; a professional attacker would have to take apart the sample and micro-probe the chip. Such methods require equipment that cannot be afforded by a "hobbyist" attacker, and so their only hope was to exploit a software bug to get access to the data. Even today most microcontroller EEPROMs do not have any special hardware security protection, with the exception of the obscurity of the programming algorithm. In some cases the ROM read-back function is disguised, or replaced with a verify-only function. The verify-only approach can be very powerful if implemented properly, as it is in most microcontrollers used for smart cards.

Microcontrollers with on-chip program memory today often have one or more security fuses that control access to the information stored in on-chip memory. These fuses can be implemented in software or in hardware [10, 11]. Software implementation means that a password is stored in the memory or a certain memory location is assigned as a security fuse. The earliest implementation was for the fuse to be in the logic for the read-back function of the programming interface. The drawback of this design is that the size of the fuse makes it easy to locate and perform an invasive attack. For example, the state of the fuse could be reconfigured by connecting the fuse logic output directly to the supply or ground line. Another well-known example of such attacks is erasing the security fuse under a UV light. The next concept in designs was to make the security fuse part of the memory access circuit, so that any external access to the data is disabled if the fuse is set, usually the fuse is located very close to the main memory or even shares some control lines with it. If the fuse shares the same technology as the main memory array it makes it harder to locate and reset directly. One solution, used in the Motorola MC68HC705C9A microcontroller, was to place fuse cells bit-lines mixed in between the main memory cells. However, other noninvasive attacks are possible, because a fuse cell was often a one-time programmable location in the memory so the fuse may operate differently from the normal memory. As a result a combination of signals could be found under which, thus allowing the access to the information stored in the on-chip memory. Noninvasive attacks could be automated reducing time and effort. Alternatively, the attacker may try using glitch attacks to confound the security check subroutine, or using power analysis to see whether a password guess is correct or even partially correct. This is useful if the fuse is in a separate memory cell to the main memory array. For example, this was the case for early Microchip Peripheral Interface Controller (PIC) and early Atmel AVR (this is an Atmel brand not an acronym) microcontrollers. In both cases, the fuses could be easily found and disabled by one or another method. The simplest way is to check the state of the fuse on power-up, on reset, or on entering the programming mode. The state of the fuse might be changed for a short time by power glitch or laser pulse. Storing the fuse state in a register may not help, because the fuse state is checked only once and the register could be changed by fault injection. The PIC16 $\times$ 84 became popular in many hobbyist applications because it uses a

simple serial programming algorithm. It also used an EPROM memory, which was easy to erase. It also has a 64 byte EEPROM for storage of user data. The PIC16 × 84 was easily tweaked to allow hackers to read its protected contents, simple disassembly software could then reproduce the source assembly files. Microchip corrected this by introducing the PIC16F84 (and later the PIC16F84A) and discontinuing the PIC16C84.

Fuses are more secure when located within the same memory array but with separate control and signal lines. For example, fuse and main memory cells can touch each other with bit-lines, as in the Zilog Z86E33 microcontroller; or with word-lines, as in the STMicroelectronics ST62T60 microcontroller. Even if the fuses could be erased with electromagnetic radiation it is likely the main memory area would be damaged trying to erase them. At the same time, semi-invasive methods may work on some Embedded CPUs if the fuses have a separate control circuit that could be attacked without affecting the main memory. Apart from different implementations, the security fuse can be monitored in different ways. It is preferable to ensure the fuses are checked each time there is a data access. It may be more secure if the fuse state is monitored in real time and any change affects memory access. In this case, any attacker will have to disable permanently the fuse to access the information. A further improvement is the Anti-fuse [12]; this is a different kind of One-Time-Programmable (OTP) memory that uses programmable interconnection links between metal wires inside the chip. As these links are extremely small, (~100 nm wide) it is virtually impossible to identify their state and that gives an extremely high security level to the devices based on this technology.

In some early Embedded CPU architectures there were various undocumented features in their command sets, e.g. Z80, 8085, 8048 besides the occasionally obscured ROM read out command. These commands were available, depending of the particular vendor, to offer commercial advantage to special customers. They could provide test routines or to preserve compatibility with other members of the family, e.g. the 8085 with the 8086. These features varied by licensee, but it was common practice in the early developments. One apocryphal command said to exist in some CPUs was the HCF command—the Halt and Catch Fire command, [13], it offered either a hazard to hobbyist programmers or a target for hackers. The 68000 HCF command is believed to be used as a memory checker during production, as it halts the processor and reads through all memory locations as fast as possible and can only be stopped by resetting the system. Certainly some exotic commands did exist on other devices, some of which were discovered by the use of various disassembler tools that had been developed to regenerate source code. Even in later designs such as the Intel Pentium and some of its derivatives there was the so-called F00F command or a bug. This instructed the CPU exception handler to stop servicing interrupts. As a result, any Embedded CPU must be reset. This so- called “Bricking” command of an Embedded CPU can be considered a serious security flaw and can be the target of various “denial of service” attacks. Newer designs include non-executable bits to make some address space secure. ARM has instigated this feature in its TrustZone(r) concept to provide such a secure execution environment for some code.

To make invasive attacks more difficult the Embedded CPUs designs have used a final layer metal or poly-silicon mesh for some time [14]. Logic paths in this mesh are monitored for interruptions and short circuits, and cause reset or zeroing of the EEPROM memory if triggered. In addition, sensors for light, voltage, frequency and temperature maybe included to test for invasive attacks. Normally, such protection is not used in ordinary Embedded CPUs because it increases the design cost. Such sensors could be also triggered unintentionally in their environment such as in automotive applications. Ordinary microcontrollers sometimes have been seen with a fake top layer mesh, whilst this may not stop the determined reverse engineering attempt it is an effective hurdle for simple optical analysis and basic micro-probing attacks. In secure Embedded CPUs such meshes have incorporated various tamper detection mechanisms and sensors. The logic lines in the mesh are polled to check for timing interference, indicating a short circuit and the data on the lines may have randomly changing encrypted data, thus discouraging false signal injection. However, not all meshes are perfect and flaws make micro-probing attacks possible. Some semi-invasive attacks are still possible if the mesh has gaps between the wires and light/radiation or a micro probe can pass through the gap into to the active areas of the circuit. Some user programmable Embedded CPU designs have a non-standard programming interface, allowing a one-time programmable option, effectively a WORM function. In some recent Embedded CPUs further protection against micro-probing attacks is provided by bus and memory encryption. This means even if the chip is stripped down to its active layers without having access to the key materiel the sensitive information cannot be retrieved. This protection process often prevents invasive and semi-invasive attacks. In the past noninvasive attacks could still be possible if the CPU used unencrypted data. The data reaching the CPU could then be vulnerable. An example was the encrypted data stored in external program memory of the old Dallas Semiconductor DS5002FP encryption engine. Weaknesses were found in the data encryption method used in this CPU that lead to a relatively low cost attack published several years ago [23]. In a standard Embedded CPU like the PIC microcontroller, an attacker can easily trace the data bus coming from the memory to the CPU. To reduce further the chance of a micro-probe attack, various non-standard circuit layout processes have been used in secure Embedded CPUs. The standard circuit blocks used in a CPU such as the register file, ALU, instruction decoder, have been laid out in a pseudo-randomly way. This approach is sometimes called ‘glue logic layout’ and it is widely used in ASICs. Glue logic makes it very difficult to monitor the CPU information physically. Semi-invasive attacks will be difficult due to random layouts of blocks. Of course given time the probing can be automated to test every possible point and then cross-analyse the results. This approach takes a long time and may not be successful. It may be easier to attack a memory block or its control circuit as they cannot be implemented with a glue logic structure and are often physically separate.

Semiconductor process changes to facilitate faster processing and smaller lower cost production has become more expensive with each new generation of geometric feature reduction. This progress is also increasing the costs to the attackers. Ten years ago it was possible to use a laser cutter and a simple probing station to get access to

any point on the chip surface. Even today, second hand semiconductor production test equipment can be acquired quite cheaply. The original owners may have used these tools to repair devices that might have errors in the top layer metal mask, which might include the ROM of the user. These tools then provide potential for attackers, with deep pockets, to attack some chips. However, with the latest multi-metal layer semiconductor ICs, with silicon geometries measured in tens of nanometres, most potential attackers are excluded and new attack methods must be found. For example, the structure of the old Microchip PIC16F877 microcontroller was easily observable and could be reverse engineered under a microscope. The second metal layer and poly-silicon layer can still be determined even when buried under the top metal layer. This is because each mask layer in the semiconductor fabrication process follows the shape of the underlying layer. An observer may determine not only the top layer logic functions but also shapes of circuits in structure of the deeper layers. With newer technologies, for example in the Microchip PIC16F877A microcontroller, each layer is smoothed using both chemical etching, and mechanical polishing before the application of the next layer. In this way, the top metal layer does not indicate the features of the deeper layers. The result is that for an attacker to identify the circuit functions they carefully have to etch the chip layer by layer. Currently, many circuit functions are spread across several layers, the result is a three dimensional jig-saw with no big picture.

3.4.1 Security of Embedded CPU Memory

An Embedded CPU operates according to the program located in its memory. There are many different memory types and most of them are used inside microcontrollers. The majority of recent Embedded CPUs are made with CMOS technology. Embedded CPUs often have different memories on the same die. Developers can then use the appropriate memory technology for each different data functions, Static Random Access Memory (SRAM) for cache, Read Only Memory (ROM) for programs, and EEPROM for user variable data, or program updates. This has led to attackers trying to identify the memory cells and the control circuits as a focus of different sensitive data. The EEPROM is likely to hold the user credentials, the SRAM an individual session key. The ROM may hold the communication or encryption algorithms and of course the designers IP for all sorts of software processes. From the traditional security point of view, an Embedded CPU with ROM has less risk than one with EPROM memory, and in turn better than one with EEPROM or Flash memory as the number of possible attack vectors are limited. Most external memory devices are not designed with security in mind. For example, serial EEPROMs can be read in-circuit, usually via the SPI or inter-IC (I2C) bus. It is also difficult to securely and totally erase data from RAM and non-volatile memory.

Early Embedded CPUs incorporated a masked ROM or relied on external Ultraviolet EPROM for program storage and SRAM for data storage. Masked ROM is still used where large-quantity production and low cost are required.

Such microcontrollers may not be marked with their part number on the package and have only a manufacturer's logo and a ROM version number. Masked ROM offers very good performance, but cannot be reprogrammed or updated. Normally, the ROM of a standard Embedded CPU does not allow any form of external access. There are few examples where a ROM is the last layer metal mask as it is intended to be modified during production as a way of personalising the devices. In standard CMOS masked ROM the data is stored as a NOR function, this allows active layer programming; the logic state is encoded by the presence or absence of link to a transistor. Information from this type of memory is observable under an optical microscope. This type of feature was offered in Dallas Semiconductor/Maxim iButton products [24] for serialisation and the information is programmed by cutting memory bits with a laser cutter. This memory allows an attacker with sophisticated tools to change the memory contents on a nominally secure product. For semiconductor geometries smaller than 0.5 microns, further processing might be required to remove the top metal layers, which may deter observation.

With the advent of microcontrollers with integrated UV EPROM, a reprogrammable single chip embedded design became possible. In fact, there were usually two versions—one for prototyping, in ceramic packages with a quartz window allowing write and erasure of the program and another in standard plastic packages for mass production allowing a single One-Time Programming (OTP). The early devices had some disadvantages as they required high voltages for programming, which might not be available on the circuit board so in-circuit programming was not possible. This was in effect a security measure as data could only be written only one byte or word at a time, so taking a long time to program a whole chip. Some plastic packages were not 100 % UV opaque allowing OTP devices to be erased, but the time for an erase operation is around 20–30 minutes under a very intensive UV light source so it was unlikely to be attacked without some careful planning. However, attacks on devices using photographic flashgun have been known for some time [15].

George Perlegos at Intel developed Electrically Erasable PROM (EEPROM) memory in the late 1970s. The first products were discrete memory devices and it offered a great advantage over the EPROM by allowing full electrical control over both write and erase operations. Due to high manufacturing cost and complexity, it was not widely embedded in single chip Embedded CPUs until the early 1990s. Even today, many embedded system may have a small serial bus EEPROM on board to store configuration settings or transaction logs. The more recent Embedded CPUs have relied on EEPROM, which has several advantages over the UV EPROM: it can be reprogrammed electronically, in- circuit, up to hundreds of thousands of times; the high voltages are usually generated by on-chip voltage charge-pump circuits; and programming is much faster.

A further improvement of the EEPROM memory, called Flash EEPROM, is becoming the main memory storage for modern Embedded CPUs. It offers much faster programming, it can be reprogrammed in blocks saving a lot of time, and this can be repeated thousands of times. Most of the modern microcontrollers with Flash memory offer internal memory programming, thus allowing field code upgrades without expensive programming tools. Flash memory also has high density offering

3–5 times more storage capacity than the same area of EEPROM. The downside of this memory type is that it can only be erased in blocks, which are relatively large. That puts some strain on embedded software design where program updates are required. Some microcontrollers offer an alternative solution to this problem, having both new memory cells with a combined Flash and EEPROM type behaviour. Flash EEPROM has many different layouts and structures; every IC manufacturer normally has its own design process. The structure is made up of a floating gate memory with either a NOR or a NAND structure. From the security point of view, all floating-gate memories offer very good protection against invasive attacks, because of the very small electrical charge used during programming, which is buried deeply inside the memory cell, so it cannot be detected directly.

Another memory type uses a ferroelectric function to store the data. So called FRAM, has been promoted as an alternative for EEPROM and Flash memories. FRAM has a very fast write cycle and does not require internal high voltage generators, so could also replace some of the functions as SRAM used in an Embedded CPU. FRAM has a two-transistor cell with nonlinear capacitors, which are polarised depending on the applied electric field; the cell will keep its state even when unpowered. FRAM has a disadvantage in that the Read operation destroys the contents of the cell so that a refresh Write is required. However, FRAM offers very good security because its logic state cannot be detected either optically or with probes. Micro-probing of the memory data bus is of course still possible, unless the information is encrypted. However, current FRAM has a limited number of read/write cycles, the cell size is 3–5 times larger than a Flash cell and its fabrication technology is more complex, so there are very few areas where FRAM-based memories are used.

Attacks on the regular layout areas of memory on a chip have forced chip designers to introduce additional protection. For example, modern secure Embedded CPUs may have a default setting of a one-time bootstrap software loader located in the Flash memory that overwrites itself during initialisation. This eliminates any possible access to the information, unless disabled by the system designer with a password. The password can be stored at a certain address location in non-volatile memory. For example, in the Texas Instruments MSP430F112 microcontroller, the read-back operates only with the correct 32 bytes password. Although such protection seems to be more effective than previous offerings, it is open to low-cost noninvasive attacks such as timing attacks and power analysis. If the security code is sampled from the memory during power-up or reset, it could present an attacker the chance to identify the password. This could be done using a combination of brute force attack and power glitches, or by trying to force the checking circuit to get the wrong state of the memory.

Another hardware security issue for all types of memories is data remanence. Remnants of stored data may exist and be retrievable from devices long after nominally being erased and with the power removed, which could be useful to obtain program code, temporary data, crypto keys, etc. In many modern Embedded CPUs, a monitoring circuit is usually implemented, causing a reset of the hardware programming interface or preventing any write/erase operations below or above certain voltages, frequencies etc. Some system designers have assumed that the erased data

will disappear. In reality, some traces of the data may be left behind. Even in SRAM, after power removal, have shown examples of data remanence, as when frozen some SRAM cells retain information for hours [16]. To retrieve the trace of the data is not easy, but for example during the chip erase, operation if the security fuse was deactivated, the memory may be accessed normally. Then each transistor inside the memory array has to be checked by micro-probing the internal memory bus. In general, SRAM memory offers a very good level of protection by placing sensors into the circuit to avoid low-temperature attacks.

3.4.2 Security of Embedded CPU Interfaces

Almost every modern piece of assembly equipment in a factory with an electronic control unit is connected to a network. These networks carry information that controls production flow, transfers manufacturing data, and provide remote equipment management. It may be possible to diagnose and repair many failures, if the unit equipment is connected to a network. Thus avoiding expensive on-site service calls, and reducing production down time. It may be required to provide secure access to the control system in a number of situations:

- To interrogate an industrial Embedded CPU for data, even when the manufacturing machine is switched off.
- Reboot a controller station remotely.
- Ensure an operator panel is safeguarded with latest health and safety policies, without halting operations or operator intervention.

All of these scenarios and more represent possible threats. An embedded system designer must consider that attackers will attempt to access the Embedded CPU and consider the following points of attack:

- Software programming interface
- Hardware programming interface
- Third party unverified protocols
- Read-back functions
- Hardware security fuses
- Software security fuses
- Discrete memory separate from the on chip memory
- Shared memory control lines
- Shared bit-lines
- Password locations
- Verification checks at power-up
- Permanent real time monitoring

Some Embedded CPU manufacturers intentionally leave a side channel access to the code for testing or programming purposes after fabrication. Normally, the

information on these test protocols is kept secret by the manufacturers. The programming interface allows writing, verifying, reading and erasing of data in on-chip memory. It could be implemented either in hardware such as a JTAG state machine, in a proprietary interface, or in software (e.g. Mask ROM or Flash bootloader). Before initial programming and a fuse has been set, some microcontrollers offer a software controlled boot loader for in-system programming. Others offer a fast hardware interface for mass production programming. For example, an Embedded CPU may have in-circuit serial programming via a synchronous interface (e.g. SPI, JTAG), fast industrial parallel programming, and a software boot loader via an asynchronous serial interface (e.g. USB). The JTAG (IEEE 1149.1) interface maybe an Achilles' heel of the system. JATG can provide a direct interface to the internal registers of Embedded CPU and so has become a common attack vector. A JTAG interface to USB test harness can be bought or self-assembled with a few low cost commonly available components, allowing automated attack routines to set easily set up. Removing JTAG functionality from a device is difficult. System designers usually disguise links, cut traces, or blow fuses. However, a determined attacker can easily repair most of them. Such test lines are used in smart card ICs only during the initial wafer manufacture. These lines are routed into the sawing corridors of the die during chip layout. These lines are then destroyed during chip separation. This technique when used with the combination of fuses make micro-probing for the lines useless.

The In-Circuit-Emulator (ICE) is a commonly adopted tool as a software program debugging technique. The GUI interface debugging software can help a legitimate user to debug easily. If freely available, these tools may reduce the time taken to attack an embedded CPU design. One solution for embedded system designers that need to protect their embedded software, from competitors and counterfeiters, is to use a secure Embedded CPU as an in-system software authentication device. To protect embedded software from cloning a challenge is sent at random intervals from the secure Embedded CPU. The response to the secure Embedded CPU is then compared to the expected response. By providing a large number of challenges and placing those in unique areas, the source code can be relatively well protected. This makes it extremely difficult for anyone to reverse engineer the source code. This added difficulty will make it more cost effective for attackers to develop an entirely new system rather than modify the existing source code.

3.5 Advanced Chip Design

Advanced embedded designs may use more than a single chip CPU, and as part of an ASIC or other VLSI design. Synthesisable logical blocks such as DSPs and RISCs and CPUs have been available for nearly 20 years. Their first development in the early 1990s, was to reduce the system cost by using the minimum functions needed for an application and to improve the system size or performance or security. Implementation tools are often optimized for a specific FPGA family or ASIC library

and for a given range of clock frequencies. When designing a synthesised Embedded CPU, a few important aspects must be taken into account; area usage, performance in terms of throughput, and added value. To protect what maybe a non-secure hardware platform various techniques have been employed to protect the design from an illicit observer, such as introduction of random or spurious logic blocks but these may impact performance or power consumption. Synthesis tools have led to multi-core designs for embedded applications incorporating two or more CPU's with DSP functions. Safety and security features are sometimes included such as error correction on the memory, parity checking on some interfaces and interrupt registers, redundancy checking functions, and advanced memory lock protection. In addition as previously described the software in an Embedded CPU-based system may be protected often by a mixture of encryption and fuse protection, against unauthorised attacks. Although this will provide a barrier to any reading of the memory optically or by micro-probing the data bus, this data normally has to be decrypted somewhere, often in or by the main CPU and stored in a SRAM cache pipeline ready for operation. The focus of the attacker may then try to detect any plaintext on the data bus close to the CPU, or better still the key to decipher the ROM. It may be possible by stopping the clock and literally freezing the circuit to read the contents of the SRAM with impunity. As a further precaution, Embedded CPU chip manufacturers offer an enhanced verify-only approach. In this case, a hash value of the content of memory is compared to a secured value and a single-bit response in the form of pass/fail sent back. The verification process can take place both in hardware or in software. It may be impossible to verify the whole memory in one go, so the process is split into blocks with their size limited by the available SRAM buffer or hardware register. The result of the verification is either to stop the Embedded CPU on detection of the first incorrect memory block, or to flag the status in a register. Of course, as described ever more sensors can be included to test for invasive attacks and tighter geometry meshes added to deflect probing or reverse engineering.

A radical departure to this concept has been developed by Infineon in the past few years, given the name "Integrity Guard™" [17] it consists of three features; error detection, full data encryption, and a new type of mesh shield. The concept is focused on not, as in earlier generations, to add more sensors and protective devices to the periphery of the Embedded CPU but to concentrate on protecting the data at all times, so no plaintext is ever used. The concept includes a mesh that uses a new shielding concept combined with intelligent secure wiring. The electrical signal lines inside the chips are rated concerning their relevance, and on the basis of this classification they are automatically routed and checked. An intelligent shielding algorithm checks the chip's layers, providing the final so-called "Active I^2-shield". The error detection is based on a microcontroller with a dual CPU allowing error detection in real time, even while processing. Both CPUs deliver their operational results independently from each other. A comparator detects whether an operation was performed the same, or if an erroneous operation was made. In the case of an error, an alarm is issued. Even the cache is an active part of the error detection, which is essential, as cache-based attacks will become a major threat for embedded security in the near future. In addition the concept applies full encryption over the

complete core and memories, leaving no plain text on the chip. The dual CPUs utilise full hardware encrypted operation, with different secret keys used in each of the CPUs. All memories are completely encrypted: for the memory buses and blocks of RAM, ROM, EEPROM, and FLASH, a strong block-cipher hardware encryption engine has been utilised. Data is enciphered from the memory encryption system to the encrypted CPU without exposing plaintext. Peripheral buses are protected using dynamically changing keys, and some peripherals work in encrypted modes. For example, the new crypto coprocessor "SCP" (a Symmetric Crypto Processor for Triple-DES and AES) utilises internal, dynamic encryption — just like the encrypted CPUs. This prevents the presence of plaintext inside important parts of the chip. This type of enhanced digital security is required as advanced attacks are developed, such as micro coil-based localised Differential Electromagnetic Analysis (DEMA). This concept is being applied to both traditional 16 bit Embedded CPU architectures and to modified ARM designs. As with all innovation it can be expected that other Embedded CPUs will also start to incorporate such concepts, but the cost of such developments and the cost of such technology has to be balanced against the threat risks.

3.6 Conclusion

The impact of embedded devices is huge. Overall, it is usually estimated that for every desktop computer chip sold, 100 microcontrollers are sold for embedded systems. Techniques for creating secure high-reliability embedded systems have focused historically on safety-critical markets, e.g. the aerospace, medical, and automotive industries. In these sectors system failures can have fatal consequences. These applications remain important, but embedded microprocessors and microcontrollers now also have an enormous impact in much broader areas of product development, such as consumer applications as diverse as simple washing machines and as sophisticated as games consoles. Manufacturers need to maximise the reliability, and the security, of the key components in embedded systems in order to reduce the cost of warranty repairs, minimise product recalls and ensure continued business. In summary, it is clear that Embedded CPUs have become more sophisticated and more secure in the past 40 years. A typical modern average house may contain 10–20 devices with an Embedded CPU, mostly independent of each other. The average citizen may carry 5–10 portable devices with Embedded CPUs as smart cards or in smart phones etc. The average mid-range car may have over 50 Embedded CPUs with over 50 % networked together. A factory employing 500 operators may have over a 1,000 networked devices. While most of these Embedded CPUs may have little access to the outside world, and little information of worth, the few such used for payment or access rights may present a target for theft either physical or virtual. However, as the issues of the "Semantic Web" [18] become reality there will be need to increase security as threats from "denial of service" or malware attacks on the user or the network provider will become more attractive. With the increasing

complexity of software and the possibility to provide remote updates there will be need for remote authentication, and integrity management of an Embedded CPU's status. It is likely that self-checking of systems will have to move past parity checks or error correction and look at frequent hardware and software image verification. The need for embedded hypervisor Embedded CPU elements will increase. This idea has been seen in the Trusted Platform Modules (TPM) for notebook computers, and the various secure elements in mobile phones, engine management systems, gaming consoles and smart meters. If the current trend [18] continues then by 2020 the current level of machine-to-machine communications could have more than quadrupled. There is little doubt that as the value of these communications increases, so will be the need for embedded security.

References

1. Ed Glynnis Thompson Kaye "Intel: Innovator of the information revolution", Published by Intel Communications Dept 1984 [Online Available] http://www.intel.com/Assets/PDF/General/15yrs.pdf
2. Gordon Moore: "Cramming More Components Onto Integrated Circuits" Electronics Magazine, 1965, [Online Available] http://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf
3. Mike MALONE, "The Microprocessor - A Biography", Springer-Verlag 1995, 0–387-94342-0, [Online Available] http://www.computerhistory.org/
4. An early example :Article 10 of 31, Article ID: 8901130503, Published on February 16, 1989, San Jose Mercury News (CA), "Start-Up used Stolen Trade Secrets, Intel Charges".
5. Charles R Mooore & Russell Stamphill: "The Making of the PowerPC", Association of Computing Machinery Communications of the ACM, June 1994, 37(6), [Online Available] http://zmoore.net/CACM%20PPC%20Alliance.pdf
6. Mark Hachman, ARM: "We'll Own over Half of the Mobile PC Market by 2015": PC Magazine May 31st 2011 [Online Available] http://www.pcmag.com/article2/0,2817,2386209,00.asp
7. ARM PLc; "Building a Secure System using TrustZone Technology", ref PRD29-GENC-009492C, [Online Available] http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf
8. Helena Handschuh: "Smart Card Crypto-Coprocessors for Public-Key Cryptography" 2000 The Pennsylvania State University [Online Available] http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.9288
9. Ronald Huizer, "Don't Give Credit: Hacking Arcade Machines", Immunity Inc, May 2011 [Online Available] http://www.immunitysec.com/infiltrate/presentations/Arcade_Attacks.pdf
10. Smart card handbook By Wolfgang Rankl, Wolfgang Effing Smart card Handbook 2003 Wiley, ISBN 0-470-85668-8 page 535.
11. US patent: 5083293, "Prevention of alteration of data stored in secure integrated circuit chip memory", Gilberg, Robert C. (San Diego, CA), Moroney, Paul (Cardiff-By-The-Sea, CA), Shumate, William A. (San Diego, CA), January 1992.
12. Inventors Cutter, Douglas J. (Boise, ID), Beigel, Kurt D. (Boise, ID), Ong, Adrian E. (Santa Clara, CA), Ho, Fan (Boise, ID), Mullarkey, Patrick J. (Meridian, ID), Luong, Dien S. (Boise, ID), Debenham, Brett (Meridian, ID), Pierce, Kim M. (Meridian, ID) United States Patent 5631862 "Self current limiting antifuse circuit", Micron Technology, 20 May 1997.
13. HCF was first mentioned in conjunction with the Motorola 68000, in BYTE Magazine, Vol 2, December 1977.

14. Inventors Gilberg, Robert C. (San Diego, CA), Knowles, Richard M. (San Diego, CA), Moroney, Paul (Cardiff-by-the-Sea, CA), Shumate, William A. (San Diego, CA) US patent:4933898, Secure integrated circuit chip with conductive shield, 1990.
15. Hinds, D.J. Stokoe, J.C.D. British Telecom Research Laboratories, Ipswich, UK, IET Electronics Letters: June 20 1985 Volume: 21 Issue: 13, On page(s): 553–554 ISSN: 0013–5194.
16. Peter Gutmann. "Secure Deletion of Data from Magnetic and Solid-State Memory", 6th USENIX Security Symposium Proceedings, San Jose, California, July 22–25, 1996.
17. Peter Laakmann, Markus Janke "A New Security Concept For The Next Decade" Secure Magazine Issue 2 2006. www.infineon.com/integrityguard.
18. Tim Berners-Lee, James Hendler and Ora Lassila: "The Semantic Web" Scientific American Magazine (May 17, 2001). [Online Available] http://www.scientificamerican.com/article.cfm?id=the-semantic-web
19. Harbor Research: 2010–2014 M2M & Smart Systems Forecast Report 2010 M2M & Smart Systems Report Brochure.
20. Motorola Press release High Performance Embedded Systems Division today announced the ColdFire(TM) MCF5200D Developer's Chip at the Embedded Systems Conference, September 12–15, 1995 in San Jose, CA. [Online Available] http://www.thefreelibrary.com/MOTOROLA+ANNOUNCES+COLDFIRE+MCF5200D+DEVELOPER'S+CHIP-a017376831
21. Samsung to use ARM core with security accelerator in 32-bit smart card chips 10/09/2001 [Online Available] http://www.eetimes.com/electronics-news/4105054/Samsung-to-use-ARM-core-with-security-accelerator-in-32-bit-smart-card-chips
22. Designing Embedded Hardware, 2nd Edition By John Catsoulis Publisher: O'Reilly Media Released: May 2005 Print ISBN:978-0-596-00755-3| ISBN 10:0-596-00755-8 Ebook ISBN:978-0-596-55662-4| ISBN 10:0-596-55662-4.
23. Pirate War Causes Shut-Off of Battery Cards article by David Lawson Scrambling News. 1996 [Online Available] http://www.mycal.net/Group42/hack/tv/sat/scnews/news0306.htm
24. Dallas/Maxim datasheet DS1427 17.02.1998 [Online Available] http://datasheets.maximintegrated.com/en/ds/DS1427.pdf