ADAPTING AN IP MC6805 CORE FOR MULTIPROCESSING AND MULTITASKING

Guillermo A. JAQUENOD
Fac. Ingeniería, UNCPBA, ARGENTINA.
chipi@netverk.com.ar

Horacio A. VILLAGARCÍA
CICPBA – Fac. Informática, UNLP, ARGENTINA.
hvw@info.unlp.edu.ar

Oscar N. BRIA
CONICET – Fac. Informática, UNLP, ARGENTINA.
onb@info.unlp.edu.ar

Marisa R. DE GIUSTI
CICPBA – Fac. Informática, UNLP, ARGENTINA.
marisadg@volta.ing.unlp.edu.ar

ABSTRACT

The availability of high-density field configurable devices provides the opportunity for designing highly integrated solutions (SOPC: System On a Programmable Chip).

Among the SOPC solutions, a case is the integration of an embedded single processor equipped with a multitasking operating system. As an alternative to a single processor the embedding of various processors on a chip, even heterogeneous and with multitasking capacity, may be considered.

A distinctive characteristic of a SOPC device is that the tasks to be performed are well known before the design starts. That feature is opposed to the traditional multiprocessing and multitasking systems in which general purpose applications are adopted during design. The benefit of this knowledge is that hardware as well as software can be adapted to fit the application's requirements.

This paper presents the hardware modifications performed on an microcontroller embedded core, to allow its inclusion as a multitasking device in a "multiprocessor on a chip", through the addition of a hardware task manager (scheduler) and communication channels among processors.

RESUMEN

La disponibilidad de dispositivos de Lógica Programable de alta densidad de integración permite buscar soluciones integradas en un dispositivo SOPC (System On a Programmable Chip).

Un tema de creciente interés son los procesadores empotrados, siendo usual un único procesador y un sistema operativo con capacidad de multitarea.

Sin embargo, debe considerarse como alternativa insertar varios procesadores, no necesariamente idénticos, que pueden a su vez atender varias tareas. En un SOPC, como diferencia fundamental con los casos tradicionales de multiprocesamiento y multitarea, las tareas a realizar son conocidas antes de comenzar el diseño, por lo tanto hardware como software se pueden configurar a medida de la aplicación, combinando la velocidad propia del primero, con la versatilidad del segundo.

Este artículo describe las modificaciones de hardware realizadas al núcleo IP (Intellectual Property) de un procesador, de modo de permitir la inclusión de un administrador de tareas por hardware y de canales de comunicación interprocesadores.

ADAPTING AN IP MC6805 CORE FOR MULTIPROCESSING AND MULTITASKING

Guillermo A. JAQUENOD
Fac. Ingeniería, UNCPBA, ARGENTINA.
chipi@netverk.com.ar

Horacio A. VILLAGARCÍA
CICPBA – Fac. Informática, UNLP, ARGENTINA.
hvw@info.unlp.edu.ar

Oscar N. BRIA
CONICET – Fac. Informática, UNLP, ARGENTINA.
onb@info.unlp.edu.ar

Marisa R. DE GIUSTI
CICPBA – Fac. Informática, UNLP, ARGENTINA.
marisadg@volta.ing.unlp.edu.ar

ABSTRACT

The availability of high-density field configurable devices provides the opportunity for designing highly integrated solutions (SOPC: System On a Programmable Chip).

Among the SOPC solutions, a case is the integration of an embedded single processor equipped with a multitasking operating system. As an alternative to a single processor the embedding of various processors on a chip, even heterogeneous and with multitasking capacity, may be considered.

A distinctive characteristic of a SOPC device is that the tasks to be performed are well known before the design starts. That feature is opposed to the traditional multiprocessing and multitasking systems in which general purpose applications are adopted during design. The benefit of this knowledge is that hardware as well as software can be adapted to fit the application's requirements.

This paper presents the hardware modifications performed on an microcontroller embedded core, to allow its inclusion as a multitasking device in a "multiprocessor on a chip", through the addition of a hardware task manager (scheduler) and communication channels among processors.

1. INTRODUCTION

The design of a computer processing system [8][10][12] strongly depends upon the exact knowledge of the characteristics of the problems to solve:

- When the tasks are unknown and diverse, the solution is to use a general-purpose processor, e.g., a personal computer.
- When the system will be used to compute specific but yet undefined tasks (e.g., image processing), it is worth choosing specialized processors as DSP´s with a large amount of memory or particular I/O features.
- When the application is totally known before design starts, the pertinent approach is to use the best adapted hardware resources, and in such a case even to use an ASIC (Application Specific Integrated Circuit).

The System On a Chip (SOC) solution is the answer to the actual demand for the integration of full systems in small spaces, with a short time to market effort. The design methodologies based on SOC can take advantage of libraries of IP blocks that have been already designed and verified. Actually, the reusability of IP blocks allows the design of new SOCs attending to the space and time demands [6][7][9][11].

Moreover, in the field of programmable logic devices, the trend is moving towards SOPC (System On a Programmable Chip) alternatives. Besides, there is a growing interest in the literature in presenting IP blocks for specific functions [4] [15].

The leading companies are already offering some commercial products including a single processor, a real time operating system (RTOS) with multitasking capabilities, and a set of programmable resources:

- ATMEL is offering an 8-bit RISC processor (AVR), with suitable amount of RAM and ROM memory, and a 10K to 40K gates in a programmable block.
- TRISCEND is offering a 32-bit ARM7DMI, with internal cache memory, interfaces to external memory, peripheral devices (timers, UART´s, interrupts), and a programmable matrix with an equivalent complexity of 40K gates.

- ALTERA is offering a softcore alternative called NIOS [13], with configurable data bus width. A hardcore alternative, belonging to the Excalibur family, offers three ARM922T models and three MIPS32 4Kc models [14].
- XILINX has announced a 32-bit softcore alternative called MicroBLAZE, which includes UART, timer, parallel I/O, interrupt controller, multimaster arbitrator, FLASH memory interface, and different RAM types.

All the above solutions are based on a unique powerful processor, their own peripheral devices, and interconnection resources with a programmable logic array.

As an alternative to the above-proposed single-processor solutions, it is possible to include several processors [1][2] on a chip. Moreover, every processor can be different from each other and devoted to specific tasks, in an structure called MPOC (Multi-Processors On a Chip).

The key difference of this approach is related to the knowledge of the tasks to be performed:

- In traditional multiprocessing / multitasking designs, the features of the tasks are knows 'a posteriori' because they are oriented to general-purpose applications.
- Unlikely, in the MPOC design, the tasks are known 'a priori', then the hardware as well as the software can be tuned to meet the requirements of the specific applications.

This paper describes the hardware modifications performed into the IP core of an 8-bit MC6805 processor, to include a hardware multitask scheduler, as well as interprocessor communication channels.

2. THE MPOC PROPOSAL

The MPOC (**MultiProcessors On a Chip**) proposal is oriented to low cost applications [5], where a structured methodology is suggested for the building of multitasking / multiprocessing applications. In this proposal tasks are assigned to processors according to the type of processes and the inter-processes communication rate. As a consequence, the use of multiple (no necessarily identical) processors can reduce the latencies and overheads of a monoprocessor RTOS:

- Tasks attending the same type of processes can reside on the same processor. With the same criteria, different types of tasks can reside on different processors; choosing for each task the best suited processor.
- Tasks with a large rate of information interchange can communicate between using high bandwidth resources (e.g., shared memory areas or FIFO´s). Meanwhile, lightly coupled tasks can use simpler channels (e.g., serial channels, such as TLINK´s [16]).

To operate in a MPOC environment, a processor should have the following characteristics:

- When attending a predefined number of known tasks, the hardware & software overload for task management and context switching has to be minimum.
- When interacting with other processors, the hardware required for the communication facilities has to be as reduced as possible.

Based on those requirements, an MPOC can be seen as a hierarchical structure composed by processors, tasks, channels, and I/O ports.

Figure 1 shows a schematic MPOC, as it has been presented in [5]. In that system several processors attend several tasks (some of them just one and other more than one), and communicate among them using point-to-point channels or some broadcasting facility. Many of the tasks can communicate with external world using I/O lines, while other ones are just internal processing tasks.

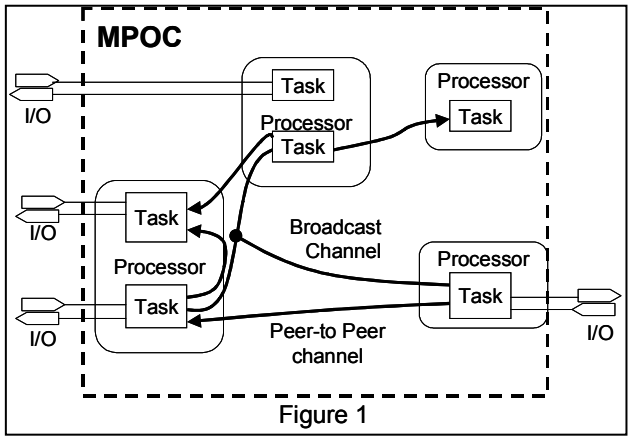

Figure 1

As an example, consider the design of a car computer. In this case there are contextually different tasks:

- Related to the engine: combustion and ignition control, temperature control, oil pressure control, etc.
- Related to the structure: adaptive damping control, airbags, brakes (ABS) and traction control, etc.

- Related to the comfort, navigation or others: air-conditioned, navigation computer, audio devices, centralized lights control, anti-burglar alarms, etc.

A quick analysis shows the following:

- The tasks related to the engine are strongly related among them, and the relation between these tasks and those of the general type is almost nonexistent. The tasks related to the engine require intensive numerical computation, that could be solved by DSP´s.
- The tasks related to the structure conforms also a compact block sharing common sensors and actuators. In this case common solutions are based on fuzzy logic.
- The general type tasks include a high amount of I/O bit-level operations, resources for multiple timers, and communication channels to peripheral devices. A general-purpose processor could be used in this case,

3. ADAPTING AN IP MC6805 CORE FOR MULTITASKING

The MC6805 is widely used in low cost applications. Their characteristics can be found in the technical manual [17], nevertheless we present its main aspects.

It is a fixed-point processor, with an 8-bit data bus, and Von Neumann architecture. The CPU has a few internal registers: a variable –up to 16 bits- program counter (PC), an 8-bit accumulator (A), an 8-bit index register (X), a 5-bit stack pointer (SP), and a 5-bit status register (CCR). Variables, instructions, and I/O share the 64 Kbytes address space, and can be referenced using ten different addressing modes.

The design of a single task MC6805 processor using Altera's FLEX10K devices has been presented in [3]. This design uses a very reduced amount of resources (about 500 logic elements), and has be taken as the starting point for this work.

For multi-task support it is necessary to perform a fast context switching, saving all the variable values belonging to the leaving task, which will be used during the next instance of this task. That implies the saving of two resources:

- The private data (variables stored in RAM).
- The value of the processor registers.

The protection of the private data can take advantage of the fact that the size of code and data used by each task is known 'a priori', before the synthesis of the processor core within the programmable device. Due to that characteristic, it is possible to use one common memory for all the tasks, assigning slices of this memory to each task, pointed by constant offsets.

Figure 2 shows the necessary changes to perform over the MC6805 address computation unit presented in [3]. The resources added are an adder and a constant offsets table.

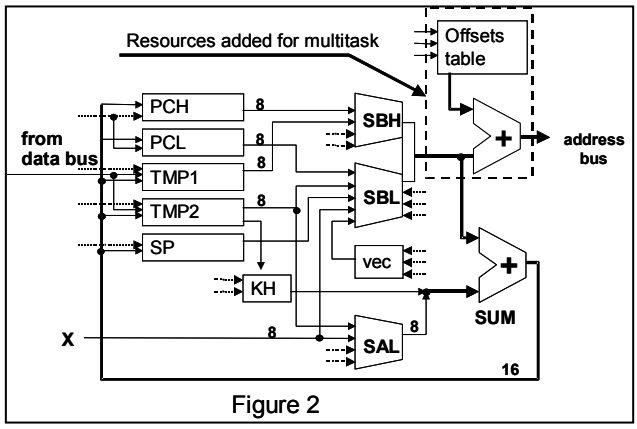

Figure 2

A later elaboration could be to differentiate the access to either RAM or ROM, generating offsets over different memory areas to optimize memory usage. That distinction should be essential when using external RAM/FLASH memories. Besides, this multiple offset scheme can also be used for the definition of shared areas of memory.

For up to 16 tasks of variable code length, the generation of the offset table will use as much logic elements as the wide of the address bus plus those necessary for the adder. As an example, given 8 tasks, with less than 12-bit address buses each, the generation of the final 15 bit address bus would require only 30 additional logic elements.

The saving of the register values can be performed in parallel or sequential form. In the parallel case, each register of the original single-task processor is replaced by circular buffer of registers, one for each task.

Figure 3 shows the hypothetical case of a processor attending 7 tasks, where it can be seen that the active register behavior is independent of which is the active register (selected by the multitask control stage).

The circular nature of the registers buffer enables the switching from one task to the next one in a single clock cycle, with minimum time overhead. In

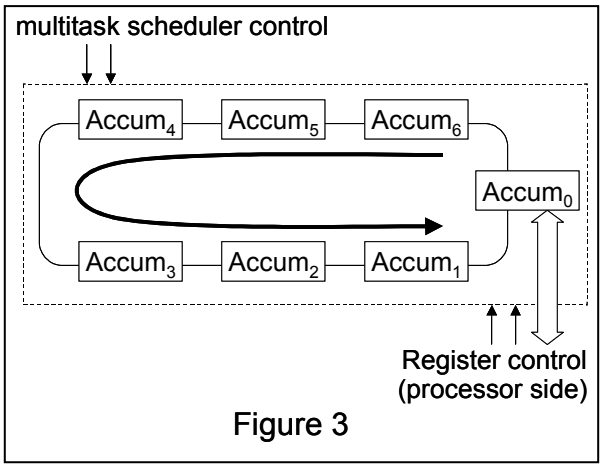

Figure 3

the worst case, if task 0 must be switched to task 6, the context switching latency could be 6 clock cycles.

To save the registers A, X, CCR, SP, and a 12 bit PC, it is required to add 38 new logic elements for each additional task.

In the sequential case, the register saving process can take advantage of the fact that the MC6805 automatically stores the registers on the stack before serving an interrupt request, and it reads they back when returning from the interrupt routine. The only register not preserved in that automatic saving is the SP, which can be stored using a circular buffer.

Figure 4 shows the modifications made on the original MC6805 control state machine, where only one new state (s30-SCHED) was added to the 30

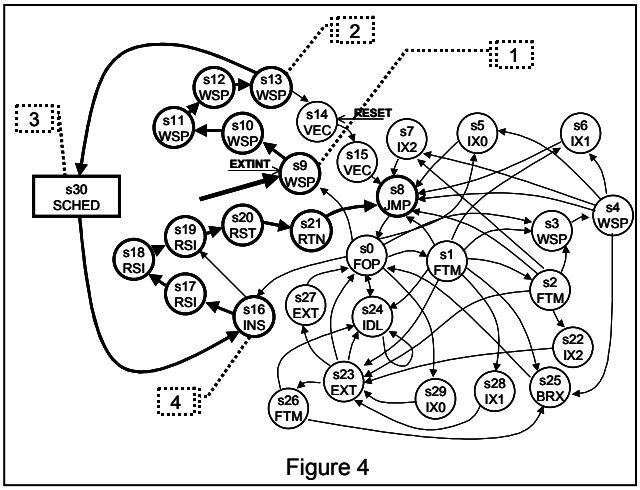

Figure 4

previous states (s0 to s29) to make possible the context switch.

The scheduler begins the switch cycle requesting an interrupt (marked as 1), which forces the registers stacking (state sequence s9, s10, s11, s12, s13). When the state machine reaches state s13 (marked as 2) during a context switching, it moves to state s30 (marked as 3) instead of s14, where an interrupt vector is fetch. The first action in s30 is to save the old SP in the circular buffer, loading it with the SP value of the incoming task. Concurrently, offsets are changed to point to the memory areas of the activated task. In the next cycle the state machine returns from interrupt (marked as 4) already in the new context (state sequence s16, s17, s18, s19, s20, s21, s8). The modification requires adding minimum hardware: 5 logic elements for each new task (for saving the stack pointer), and 5 logic elements for adding state s30. In this case, minimum switching latency is 13 clock cycles.

4. COMMUNICATION CHANNELS

From the MPOC point of view, a communication channel is a hardware object describing a link among processors. From the point of view of a processor, a channel is seen as a peripheral, that can be a serial transceptor, a parallel port, or any more complex element, such as a shared memory area, or a queuing buffer.

A transaction message is sent by one task and received by another, and can be used for synchronization. The transaction can be either originated by the transmitter (writing new output data) and closed when the receptor reads it, or started by the receptor (requesting new input data) and closed when the transmitter send it. In both cases, the agent which triggers the transaction remains halted until the transaction is closed, therefore it is reasonable to include resources to take advantage of that time to process other tasks.

Models for channel ports are presented in [5] including synchronization signals (rdy). As an example, a parallel port is the most simple hardware scheme for implementing task communications (Figure 5). The transmitter uses a register for storing the data and a simple state machine for synchronization. The receptor is implemented using also another small state machine.

- When the transmitter writes new data (***ld*** active) the ***rdy_tx*** line becomes inactive, meaning that it is waiting, and the availability of data in the channel is indicated by ***new*** active. When the receptor read the data (***rd*** active), the signal ***rdy_rx*** becomes

inactive, remaining in that state until the transmitter writes new data. At the same time, ***rd*** activates ***ack***, which reinitializes the transmitter signals ***rdy_tx*** and ***new***.

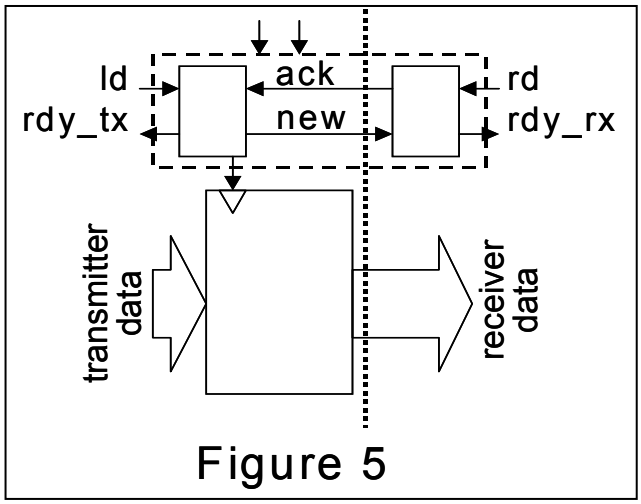

Figure 5

- The other situation is when the receptor is willing to read new data when it is not present (***new*** inactive). In that situation the signal ***rdy_rx*** remains inactive until the transmitter writes new data.

5. MULTITASKING SCHEDULER

There is not any predefined scheduler, because it architecture depends on several items: the tasks priority, the interrupts management, the existence or not of a front-end interrupt processor, and other conditions.

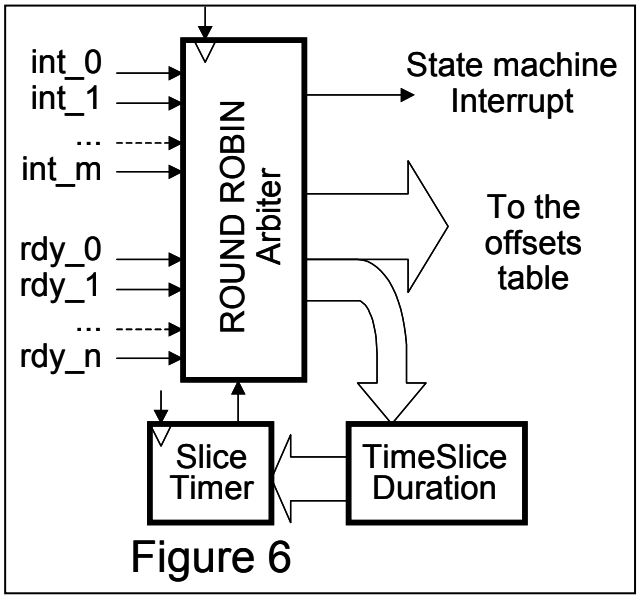

Figure 6

The simplest case is a multitasking system with equal priority tasks, using a "round-robin" arbitration scheme (Figure 6). In this case, the schedule of a new task may have several causes:

- The active task has triggered a communication transaction, then passing to idle until that transaction is closed.
- The slice time available for the active task is already exhausted and another task is awaiting.
- An external interrupt is demanding attention, and the scheduler is assigning the CPU to the corresponding task.

In any case, the scheduler interrupts the processor, and when in the SCHED state, it switches the offsets and decides the time assigned to the new task.

6. CONCLUSIONS

When an application in entirely known 'a priori' before the beginning of the design cycle, then hardware and software can be optimized according to the requirements. It has be shown that an IP core for a conventional processor could be easily extended to operate in a multiprocessing and multitasking environment., just adding a few hardware resources. That solution and the short design cycle for programmable logic devices, allow a minimal development time, an easy debugging, and short time to market.

Supposing an 8-tasks multiprocessor where the addressing bus width for each task is lower than 12-bit, then 30 logic elements are needed for the management of private memory areas; 45 logic elements are needed for the modification of the state machine and for the stack pointer buffer; and 50 logic elements for the round-robin scheduler. That represents a 25% increase in hardware when compared to the single task processor.

That increase can be reduced:

- For this microcontroller core, if the private memory areas have the same length and equal to 2^N, there is not need for the 30 logic elements required for the management of the low order address bits of that memory areas. In such a case, the 3 upper lines of the final address bus come directly from the arbiter, and the overhead is reduced to a 20%. Also, if the time slice is identical for each task, the hardware needed by the arbiter is also decreased.
- For more complex and powerful processors, with

larger address and data buses, and requiring more hardware resources, the logic complexity for a multitask operation is almost the same than that for the core described here. As a consequence the percentage of hardware assigned for that functionality is smaller.

7. REFERENCES

[1]. Dömer, R. et al, "Specification and Design of Embedded Systems", it+ti Magazine N° 3, Oldenbourg Verlag, Munich, Germany, June 1998.

[2]. Janka R.S., Wills L.M., "A Novel Codesign Methodology for Real-Time Embedded COTS Multiprocessor-Based Signal Processing Systems", Proc. of the 8th. Intl. Workshop on Hardware/ Software Codesign. San Diego, USA, May 2000, pp.157-161.

[3]. Jaquenod G., "Diseño de un microcontrolador MC6805 usando lógica programable FLEX de ALTERA". VI Workshop IBERCHIP, Sao Paulo, Brasil, Mar 2000, pp. 130-139.

[4]. Jaquenod G., De Giusti M., "Diseño de microcontroladores empotrados mediante procesamiento serial: análisis usando FLEX10K para sintetizar un microcontrolador tipo COP8Sax". VII Workshop IBERCHIP, Montevideo, Uruguay, Mar 2001. Proc. on CDROM.

[5]. Jaquenod G., Villagarcía H., De Giusti M., "Towards a Field Configurable non-homogeneous Multiprocessors Architecture". SCI 2001, Orlando, Florida, USA, Jul 2001. Proc. Vol XIV pp. 248-253.

[6]. Keating M., Bricaud P., *Reuse Methodology Manual For System-On-A-Chip Designs, Second Edition*, Kluwer Academic Publishers 1999, USA, ISBN 0-7923-8558-6.

[7]. Meerwein M. et al, "Linking Codesign and Reuse in Embedded Systems Design", Proc. of the 8th. Intl. Workshop on Hardware/Software Codesign. San Diego, USA, May 2000, pp. 93-97.

[8]. Pollard L.H., *Computer Design and Architecture*, Prentice Hall 1990, USA, ISBN 0-13-167255-X.

[9]. Seepold R, Martinez Madrid N. (Editores), *Virtual Components Design and Reuse*, Kluwer Academic Publishers 2000, USA, ISBN 0-7923-7261-1.

[10]. Smith M., *Application Specific Integrated Circuits*, Addison Wesley 1997, USA, ISBN 0-201-50022-1.

[11]. Villagarcía H., Bria O., "Diseño de bloques IP: Programabilidad y Reutilización". WICC2001, San Luis, Argentina, May 2001, pp.2-5.

[12]. Wolf W., *Computer as Components: Principles of Embedded Computer Systems Design*, Morgan Kaufmann 2000, USA, ISBN 1-55860-541-X.

[13]. ALTERA Corp., "NIOS Soft core Embedded Processor Data Sheet. Version 1". San José, CA, USA, 2000.

[14]. ALTERA Corp., "ARM-based Embedded Processor Device Overrview. Version 1.1", "MIPS-based Embedded Processor Device Overrview. Version 1.1". San José, CA, USA, 2000.

[15]. ALTERA Corp., "Intellectual Property Catalog", M-CAT-AIPS-01, Altera Corp., 1999, USA.

[16]. INMOS Ltd., *Transputer Reference Manual*, Prentice Hall 1988, UK, ISBN 0-13-929001-X

[17]. MOTOROLA, *MC68HC705C8A/D, Rev.1*, Motorola Inc., 1996, USA.