KFKI-1979-08

I. ERÉNYI

DEVELOPMENT TOOLS FOR DESIGNING AND DEBUGGING MICROPROCESSOR BASED SYSTEMS

Hungarian Academy of Sciences

CENTRAL
RESEARCH
INSTITUTE FOR
PHYSICS

BUDAPEST

2017

KFKI-1979-08

DEVELOPMENT TOOLS FOR DESIGNING AND DEBUGGING MICROPROCESSOR BASED SYSTEMS*

I. Erényi

Central Research Institute for Physics
H-1525. Budapest, P.O.B.49. Hungary

*The paper was delivered in the Conference "Mimi '78" in Zurich June, 1978

HU ISSN 0368 5330
ISBN 963 371 506 7

ABSTRACT

Descriptons are given of two different development systems used in designing, developing and debugging microprocessor based devices and systems. One of these is a minicomputer based system, it is advantageous when the main development efforts are concentrated on software design and debug.
The second tool is the Universal Microprocessor Development System which can be effectively applied when the user develops his hardware and sofware simultaneously. The UMDS has changeable emulator modules enabling it to be fitted to several microprocessors.

АННОТАЦИЯ

В статье описываются две системы проектирования и наладки цифровых устройств, выполненные на базе микропроцессоров. Одна из них построена на мини-ЭВМ; ее применение при разработке устройств особенно эффективно в случае, когда в первую очередь, осуществляется разработка и исправление программ. Второе средство - универсальная система проектирования и наладки микропроцессорных устройств; с ее помощью можно эффективно и одновременно разрабатывать и проверять аппаратуру и программу. Система содержит сменные модули эмуляторов, таким образом обеспечивается ее применимость ко многим типам микропроцессоров.

KIVONAT

A cikk két különböző fejlesztő rendszert ismertet, amelyek mikroprocesszoros készülékek fejlesztése és bemérése során használhatók. Az első rendszer miniszámitógépet foglal magában és használata különösen akkor célszerü, ha elsősorban programok fejlesztését és hibakeresését végezzük a segitségével.
A második eszköz az Univerzális mikroprocesszoros fejlesztő rendszer(UMDS)használata effektiv és párhuzamos hardware és software fejlesztést tesz lehetővé. Cserélhető emulátor egységeinek köszönhetően több mikroprocesszor tipushoz is alkalmazható.

INTRODUCTION

LSI devices and microprocessors have enabled designers to build systems and to perform tasks which previously necessitated the usage of large logical circuitry and/or minicomputers. In this way, the modern electronics industry has given us greater cost effectiveness with a much wider choice of products. Not only are these new products lower in price and smaller in size, but they incorporate totally new features, aims and application areas as well. The decreasing cost of LSI chips and the less expensive production of systems based on microprocessors act as stimulants to designers to apply them ever more widely and to utilize them in new developments. In the futute, hardware design will tend to reconfigurate LSI chips /CPUs, memories, I/O devices/ for solving different tasks. The basic problem will remain in the desing and generation of software parts, and mainly in the integration of software with the new hardware. It is not a simple procedure to test such a system, nor is it simple to check and debug errors.

Designers of microprocessor based systems often met failure due to their inability to reduce development costs and time. Microprocessors and LSI chips are new types of electronic elements that demand a new procedure for their incorporation and utilization in new products. Indeed, so far as their inner structure is concerned they represent very complex elements, much more complicated than any earlier ones.

Thus, the designers of microprocessor-based systems are faced with three requirements: to be familiar with logic

design, to have experience in writing programs, to have a system-oriented view of solving problmes.

In the past, efforts to build systems with /micro/computers were concentrated along two lines, namely: the designing of hardware and the development of software. The division of the problem along these two lines resulted in great difficulties in coordination which in turn often led to significant delays in production and the concomitant extensive increase in cost. For this reason it is important to employ tools for the development of LSI based systems. The most important step in the introduction of microprocessors was the application of the development aids making it possible, at one and the same time, to design, debug and verify the hardware and the software of the new system.

The purpose of this paper is to describe two such development systems worked out and in current use at the Research Division of Measurement and Computing Techniques of the Central Research Institute for Physics, Budapest.

APPROACH TO APPLICATION OF MICROPROCESSORS

From the pont of view of the designer /user of microprocessors/, microprocessors and their LSI chip families can be considered as microcomputer building blocks or as semiconductor logic elements [1]. These two different approaches can lead to the application of different types of design aids.

First let us examine the characteristics of the tasks and the aims which must be solved by the new systems based upon microprocessor families considered as microcomputer elements.

These are:

- The product tends to be used in low volume and its building blocks are purchased by the user in almost ready-to-use condition.

- The system memory is mainly RAM-based, the operating software must be loaded before the operation.
- The development of the product, usually the software design, is performed on the system itself.
- Testing of the system is performed by means of minicomputers, i.e. tests do not depend on the application.
- The system uses standard mechanical peripherals.

It can be said that this sphere of applications is close to the minicomputer-like usage of the microprocessors and LSI devices.

In comparison, the other end of application tasks can be described in the following way:

- The product tends to be used in much higher volume.
- The system memory is ROM-based, the program is fixed and can operate immediately following switch on. Only a little RAM is used for variable parameters.
- The software development usually demands achieving the minimal memory space since the cost of the ROM memory directly affects the final cost of the product. It is for this reason that the programs must be tightly coded.
- As the product tends to be used in high volume the development system cannot be included in the product economically: it is totally separated.
- Testing is application specific, it leads to the demonstration of the normal operation of the product.
- The peripherals of such systems are usually special LSI circuits. These chips are rather complex ones and the main part of the hardware design is concentrated in this area, i.e. the I/O circuitry.

This sphere of applications is close to the "logic element-like" usage of the microprocessors.

MINICOMPUTER BASED DEVELOPMENT SYSTEM

The basic reason behind the development system was the need to have some sort of design aid for systems with modules

of the MMPS bus configuration. MMPS is a Multi MicroProcessor bus System elaborated in our institute. Equipment and systems built around the MMPS tend to be microcomputer oriented, though some of their features suggest their classification among those built up by the "semiconductor element" approach to LSI devices; e.g. they are ROM-based, the development system is not involved in the product. On the other hand, different items of equipment are composed of different hardware building blocks /modules/, which are more-or-less in working state before they are put together.

Thus, the main purpose of this minicomputer based development system is the increase in effectiveness of the software design, assembling, editing and debugging. Naturally, this does not mean that it cannot be used for certain kinds of final tests of the hardware and/or both the hardware and software parts at the same time.

Figure 1 represents the block-diagram of this tool. The system operates on the principle of storing microprocessor programs which are developed in the minicomputer's core. By placing a single PC board interface between the computer and the MMPS bus it is possible to solve this task in a very easy manner. Thus, using all the features of the minicomputer system the programmer and/or engineer-designer has good opportunity to examine and intervene in the microprocessor's operation. Such a debug procedure enables the microprocessor's program run to be traced in real time /if the minicomputer's memory is fast enough/, or a program run close to the real time condition.

The microprocessor's source program can be written with the help of the minicomputer's editor. In many cases assemblers can easily be adapted to the instruction set of the microprocessor; there is also the possibility to have cross-assemblers for the selected type. These cross-assemblers can often be arranged by exchanging the symbol table for a new one.

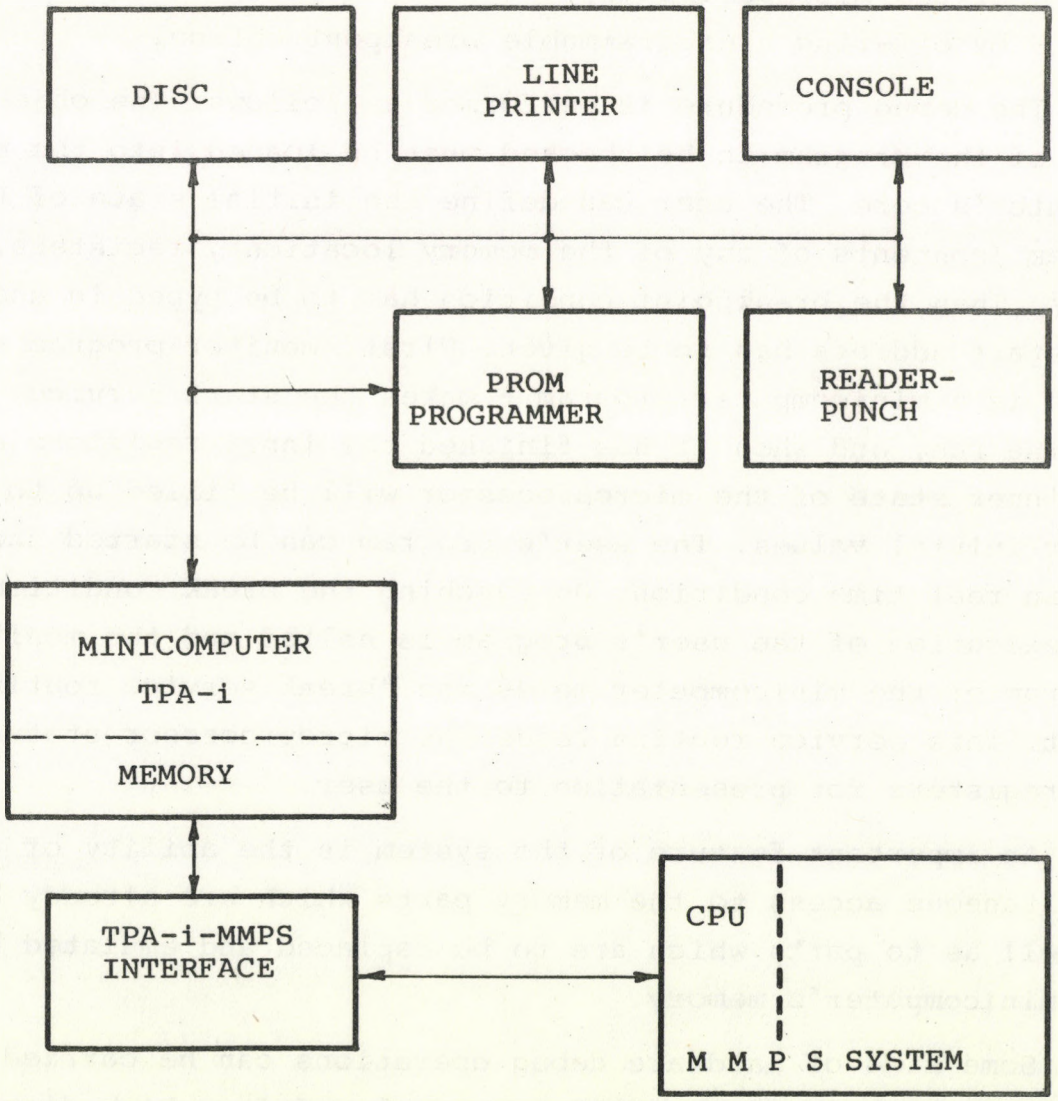


Fig. 1: Minicomputer based development system

In the debug phase the following main functions of the development system help the user:

- read and write data into the main microprocessor memory /in this case it is the microcomputer's memory as well/,
- execute the program by single instructions,
- read and write data into any of the inner registers of the microporcessor,
- by offering a programmable breakpont option.

The debug procedure is performed as follows: The object code of the program to be checked must be loaded into the minicomputer's core. The user can define the initial state of the system /contents of any of the memory locations, registers, etc./. Then the breakpoint condition has to be typed in and the start address has to be given. First, monitor program - which is a minicomputer program - makes the start service routine run, and when it has finished the inner registers and the inner state of the microprocessor will be filled up to their initial values. The user's program can be started and run in real time condition. On reaching the break condition the execution of the user's program is halted and the monitor program of the minicomputer makes the "break service routine" start. This service routine reads the microprocessor status and registers for presentation to the user.

An important feature of the system is the ability of simultaneous access to the memory parts which are already built as well as to parts which are to be replaced and emulated by the minicomputer's memory.

Some kind of hardware debug operations can be carried out with the help of the MMPS bus panel module, which displays all the bus signals, the data in connection with the bus transfers, and gives the possibility to address any of the bus modules for execution of a user initialized bus transfer.

After final checks and debug operations the operational program of the microprocessor can be burned into REPROM-s by a standard peripheral - /RE/PROM programmer - of the system.

UNIVERSAL MICROPROCESSOR DEVELOPMENT SYSTEM

A totally new application philosophy was taken into account when the UMDS architecture was worked out. The idea that microprocessors themselves can be used as building elements of the development systems is not new. They can perform minicomputer-like tasks such as editing, assembling, data handling and control of the peripherals. These functions are necessary for the software design.

Some years ago Intel was the first to announce the method of "in circuit emulation". This method offers the possibility of developing equipment and checking out errors in such a way as if the development system itself was the part of the equipment under development, i.e. it serves as a /micro/computer facility with monitor program, front panel with interrogating functions, interactive peripherals and mass store. At the same time it allows the microprocessor system to run in a free way, in realtime, until reaching the previously programmed conditions /breakpoints/. As this method is considered one of the more effective design and debug procedures it was chosen to be a fundamental principle for the UMDS.

Bearing in mind our own possibilities together with the requirements of the probable users of such a development system, the main features of the UMDS were defined as follows:

- It must fit parallel and simultaneous hardware/software design and development purpose.
- It must be able to promote the development process from the very begining to the final tests of the operation. In other words, it must be an "integrated" development system.

- It has to represent a general purpose /universal/ development system, which can be applied to many of the commonly used types of microprocessors.
- UMDS contains modules performing utility operations such as: /RE/PROM programming, FPLA programming, etc.
- The technique for handling the UMDS must be simple, clear and quickly acquirable.

The first requirement is in accordance with the principle of the "in circuit emulation". The second requirement - integrity - demands not only the "in circuit emulation" of the microprocessor itself, but emulation of the user's whole memory, and parts of this memory as well as emulation of the user's input/output ports. The third feature can be reached by applying changeable microprocessor emulator modules. This means that the UMDS must be constructed as a two-processor system. The first microprocessor is the so called master microprocessor, it serves as the CPU of the "microcomputer". The second one is applied to the emulation process.

The remaining two other requirements are self-explanatory; the only noteworthy thing is that the real simplicity and easy use of the UMDS can be achieved by the application of backing stores /e.g. floppy disc units or magnetic casette store units/.

STRUCTURE OF THE UMDS

The block diagram of the UMDS is shown in Fig.2. The UMDS can be divided into two quite separate parts. In Fig.2 these parts are separated: on part being shown at one side of the system bus, the other part at the other side. The first part organizes a well known structure of /micro/computers. It contains a Central Processor Module containing the Z-80 microprocessor, ROM and RAM modules and control modules of peripherals. The microcomputer part serves for editing, assembling the programs and for utility operations such as: programming

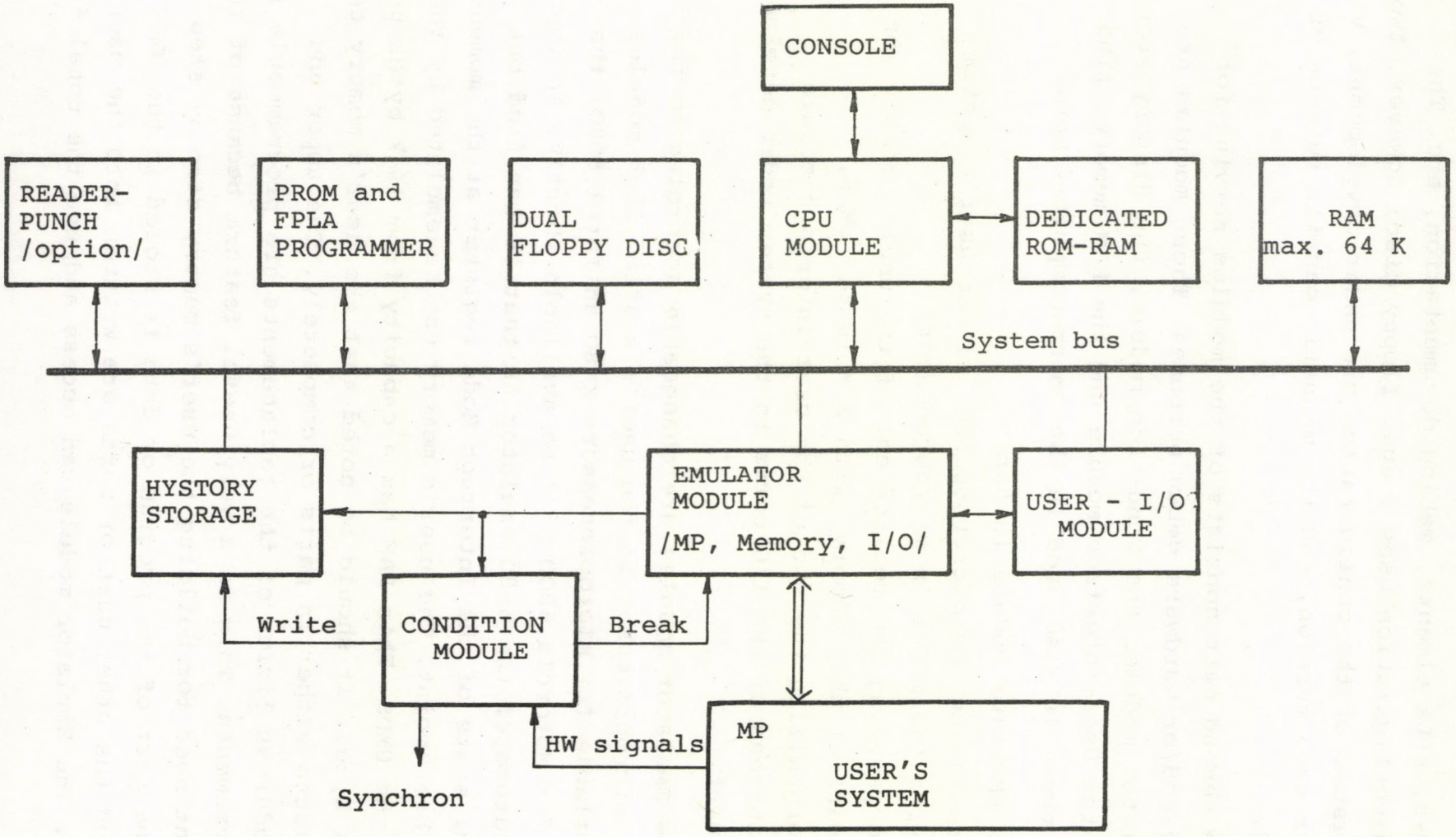


Figure 2: The hardware modules of the UMDS

/RE/PROM-s, FPLA elments, making documentation, etc. The standard configuration uses a dual floppy disc; however, two other versions of the configuration are under development, viz. the paper tape version, and the magnetic casette version of the UMDS.

The second part consists of the modules serving for software and/or hardware debug purposes. These modules are: the Emulator module, the Condition module, the History storage module, the User-interface module and the RAM memory. /The last of these is also used in the "microcomputer" part/.

The EMULATOR module is for

- replacing the microprocessor of the user's system with the module's microprocessor,
- organizing the replacement of the user's memory /or part of this memory/ with RAM in the UMDS,
- organizing the use of the user-interface module instead of the I/O ports in the system under development.

The Emulator modules are changeable according to the type of microprocessor in the user's system. Such modules are available for microprocessors Z-80 an Intel 8080; the module for Motorola 6800 will be available shortly. An important feature of the Z-80 emulator is that it can find out the real state of the Interrupt Mode register at the moment of the breakpoint. The user's memory can be emulated by the RAM of the UMDS. This RAM has a capacity from 16 k bytes up to 64 k bytes. It should be noted that the user's memory can be replaced either in parts or completely. The upper and lower address limits of the replacements are programmable in 4 k increments. This is a very useful feature because of the frequent need for building the user's memory step by step, when one part of the program or data is stored in the UMDS RAM, and the other part or parts are written into the user's memory. The Emulator module can access and use the total 64 k

bytes of the UMDS RAM, as the master processor's operating system and debug program is written into the dedicated memory which does not use the RAM address lines.

The USER-INTERFACE module serves for emulating the I/O interface circuits. The user can define the port addresses which must belong to the User-interface and not to the system under development.

The HISTORY STORAGE module can collect and store up to 256 events. An event is characterized by 40 bits, these are: the control signals, the content of the data and address buses of the CPU, and - this is a new feature - any 8 hardware bits chosen by the user. This module contains a counter to record the machine cycles or clock pulses of the CPU during the running time of the user's program.

The CONDITION module has a totally new construction - not to be met with in other development systems. This new construction provides a "multicondition-multifunction" feature enabling up to four conditions to be prgrammed before the user's program starts. These conditions are logical combinations of the CPU's control signals, data and address bus'content, or the 8 hardware bits mentioned in connection with the History storage module. The content of the data bus and the hardware bits can be masked out should any of these bits be don't care bits. If any one of the previosuly programmed conditions is met the module generates a signal named "event signal". Any of the following functions can be connected to the appearance of the "event signal" in a programmable way: breakpoint and/or history write and/or scope synchron function. This module contains an 8-bit counter as well; thus, not every event results in a function; the events can be counted and only the n-th event initializes the prescribed function /n is the initial content of the counter/. This is useful when program loops are examined. Another new feature of this module is that the conditions can be preprogrammed so that reaching an address

interval means an event. This is a useful feature if a program needs to be traced in one or more intervals of the address.

SOFTWARE SYSTEM OF THE UMDS

The software system consists of five environments. These are: the Editor, the Assembler, the File maintenance, the Debug and the Utility programs. These programs with the exception of the debug program are stored in the backing stores. The assembler or cross-assembler, a little part of the debug environment and the utility programs must be changed according to the type of emulator module.

The main part of the debug environment is stored in the ROM of the dedicated memory. During the design and generation of the debug program we followed a very careful process of defining the debug commands' form. These debug commands must be typed into the UMDS by the user. The definition of the commands has followed the principle of the minimum typing work; and on the other hand, the commands must be easily memorizable and not easily confused. The command set has functions which are similar to the others used in almost every other microprocessor development system.

The utility programs contain such program parts and subroutines which can easily be built into the user's program. A very important group of these programs is the set of the backing store routines. These routines give the possibility for the user's programs to easily access the backing store via the User-interface module and the system bus. Among the utility programs are the handlers of the /RE/PROM and FPLA programmers, too.

CONCLUSIONS

The TPA-i minicomputer based development system is highly useful during the developing and debugging phase of microprocessor based systems when the main design effort is concentrated on writing and checking out programs. The hardware

construction follows the design philosophy of using a standard system bus and standard hardware building blocks or modules connected to this bus.

The Universal Microprocessor Development System is an effective and general purpose tool for various development and error checking processes while building equipment and systems of varying complexity based upon microprocessors, LSI memory and interface circuits. It is suitable for a wide variety of microprocessor types. Because of its programmable facility to emulate the user's microporcessor, memory and LSI interface circuitry, and especially because of its unique multi-breakpoint, multi-trace and pulse features, it can be successfully utilized for any kind of microprocessor-based hardware and software development work. The UMDS has the great advantage of decreasing the time and the cost involved in designing and debugging. Fig. 3 shows the development path when the UMDS is used for designing and debugging microprocessor based systems.

BIBLIOGRAPHY

[1] L. Krummel, G. Schultz : Advances in Microcomputer Development System, Computer, February, 1977.

[2] Csákány A., Vajda F.: Mikroszámitógépek - Microcomputer /In Hungarian/ Müszaki Könyvkiadó, Budapest, 1977.

[3] W. Davidow: The Coming Merger of Hardware and Software Design. Electronics, May 29th, 1975.

[4] R.D. Catterton, G.S. Casilli: Universal Development System is Aim of Master-slave processors. Electronics, September 16th, 1976.

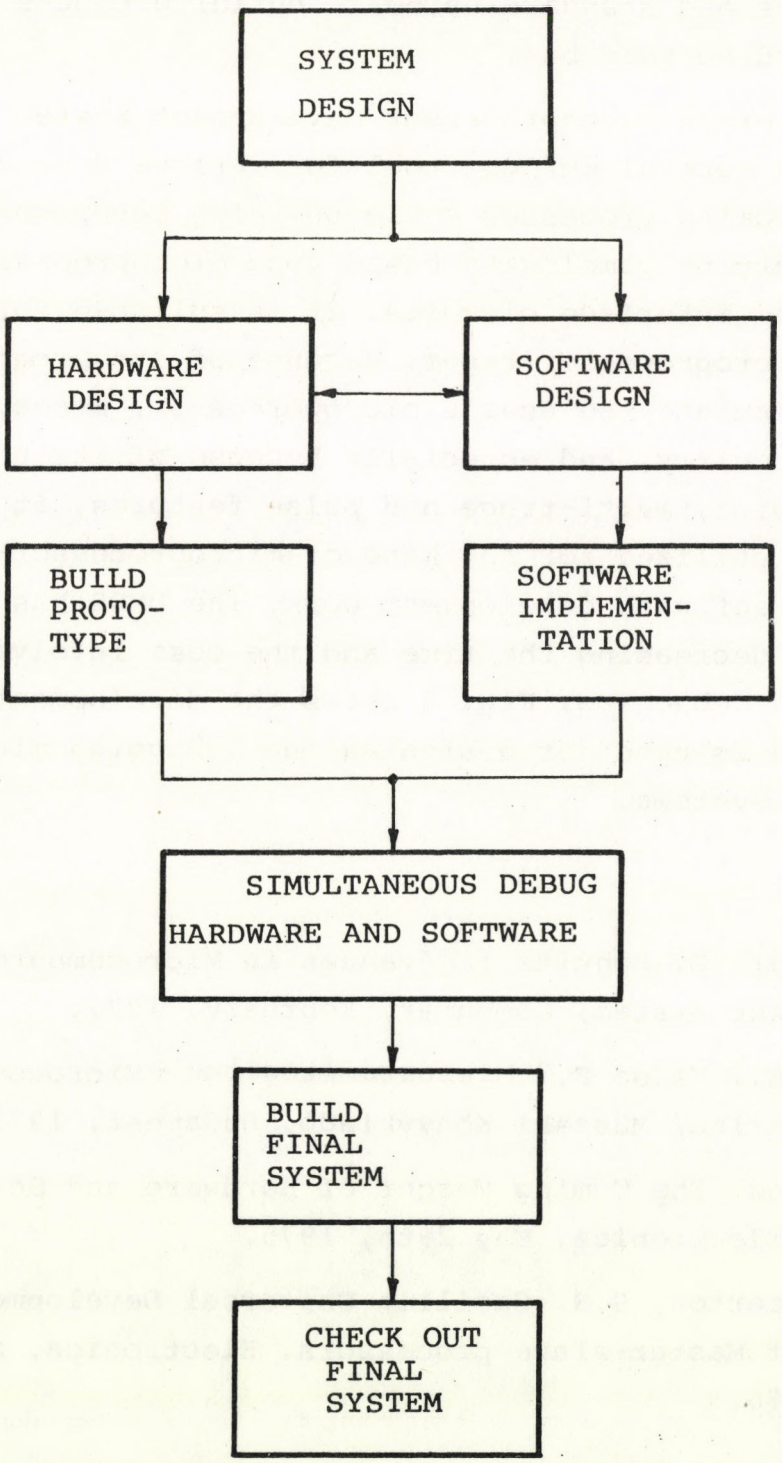
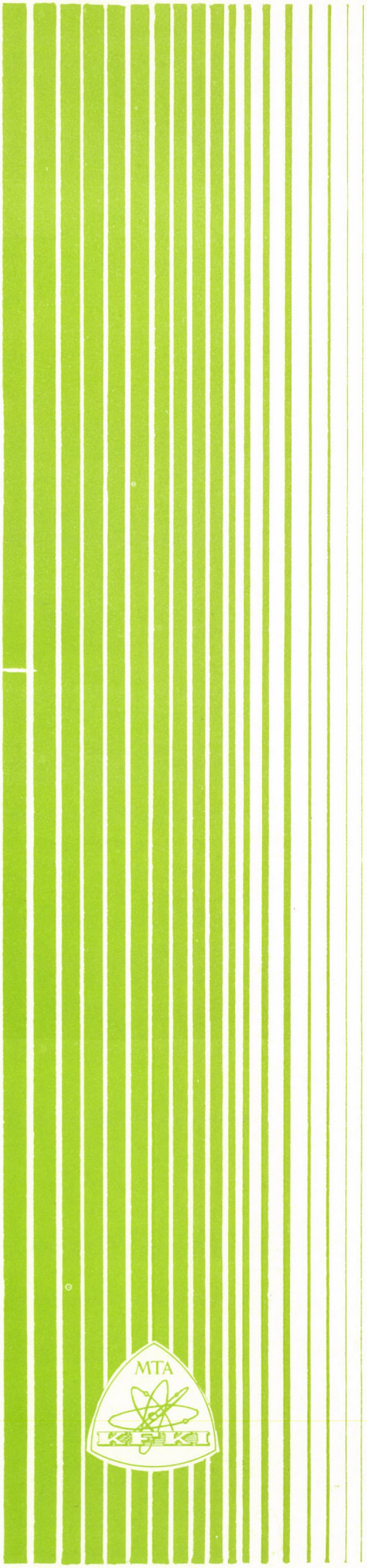


Fig. 3: Development path when the UMDS is used [3]



Kiadja a Központi Fizikai Kutató Intézet
Felelős kiadó: Sándory Mihály
Szakmai lektor: Vajda Ferenc
Nyelvi lektor: Harvey Shenker
Példányszám: 320 Törzsszám: 79-96
Készült a KFKI sokszorositó üzemében
Budapest, 1979. február hó