© EUROMICRO
EUROMICRO Journal 6 (1980) 41-43

# MODEST – a Novel Development System for an Industrial Microcomputer System

A. H. J. Schatorjé
Philips Eindhoven, Electronic Components, and Materials Div., The Netherlands

B. I. J. Bruinshorst
Philips Research Lab., Eindhoven, The Netherlands

*Microcomputers in the industrial environment require high performance but physically small low cost development systems providing on-site (re)programming facilities. Philips have introduced "MODEST", a novel system for the program development and debugging of their Industrial Microcomputer System (IMS) programs.*

## 1. INTRODUCTION

During the last few years there has been an increasing demand for microcomputer card systems. However, the starting point with microcomputers is nearly always the program development system. Until now, the potential user had the choice between large complex and expensive systems and small straightforward systems which are very limited in performance.

To make it easier to start with microcomputer systems, Philips have designed for their Industrial Microcomputer System (IMS) – the IMS-MODEST programming and debugging system.

## 2. THE IMS CONCEPT

IMS offers a wide selection of compatible system functions on individual modules. At present, ten different types of module allow design of unique microcomputer systems for most applications. Other modules are in preparation and the range is being continually extended.

Modules plug into a printed circuit back panel on which the system bus is implemented. This arrangement allows almost any combination of modules. Because modules are built on standard Eurocards, the whole system can be housed in a 19-inch Eurocard rack (DIN 41612).

Flexibility of hardware is complemented by equally flexible software. As well as fully proven standard subroutines available off-the-shelf, there is MODEST which offers indispensable aid when developing customized software.

Modules available include:

- **Central Processor** based on the Signetics 2650 series of 8-bit microprocessors, having 75 instructions and 7 addressing modes. The module has 1 Kbytes RAM, up to 4 Kbytes PROM or (E)PROM. A clock oscillator as well as DMA facility and fully buffered control lines are on the board.
- **PROM and RAM memory modules** extend CPU memory by up to 32 Kbytes, or more if memory management facilities are used. Each PROM module holds up to 16 Kbytes of PROM or (E)PROM; each RAM module holds up to 8 Kbytes of static RAM. The (E)PROM memory ICs used can be selected from several types offered.
- **Input and output modules** each having two 8-bit TTL compatible ports. Filtering of inputs is possible and outputs can sink up to 300 mA.
- **Teletype module** provides RS232 teletype interface, a current loop with opto-coupler and an audio cassette interface.
- **Modest modules** containing all (E)PROMS, peripheral circuits and drivers to be used in the MODEST system.

## MODEST

The MicrOcomputer DEvelopment SysTem, MODEST, is not more than three cards connected to the system bus (back panel) of an IMS user system (fig. 1).

With the help of these cards, the user program can be developed and debugged directly in the user system. MODEST uses the CPU of the CPU card, the RAM and/or PROM of the CPU card and/or the memory boards. In the testing phase of the system, MODEST has access to the user I/O via the CPU.

### 3.1 Principle of operation

Instruction mnemonics are typed in character by character on the VDU keyboard. The MODEST mnemonics assembler loads object code in the user RAM (for a small program, e.g. the RAM on the CPU card), where the program under development is stored. An on-line disassembler in MODEST returns mnemonics to the CRT for display.
In this way a very direct programming – eliminating the pitfalls of hex or machine level coding – is obtained. Furthermore, the addresses, address stack, Program Status Word, registers and RAM contents can be brought to the screen clearly formatted to allow program evaluation and direct corrections.

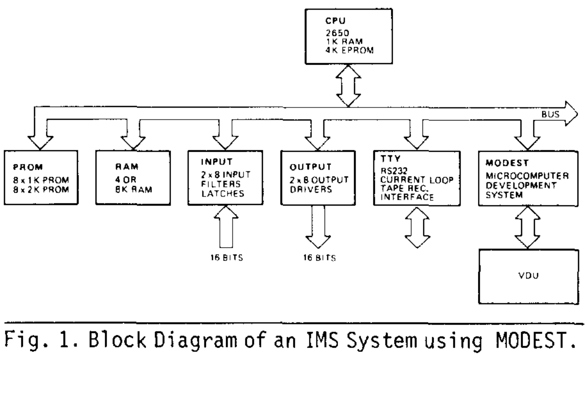

Fig. 1. Block Diagram of an IMS System using MODEST.

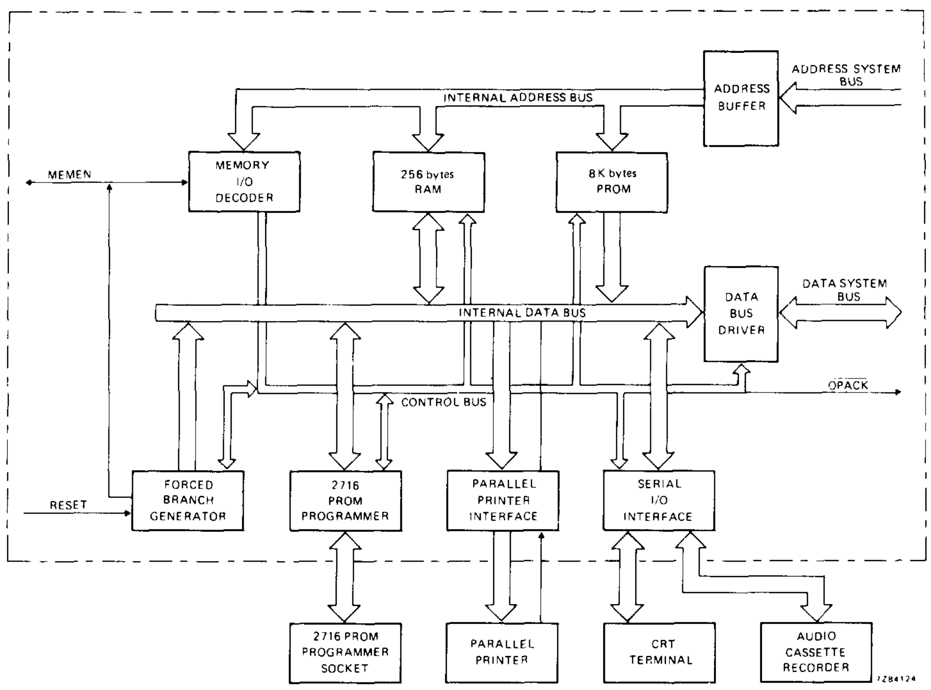

Fig. 2. Block diagram of MODEST.

**3.2 MODEST features**

- Program development in user system

  By having a resident development system using the MODEST principle a minimum hardware cost is obtained. Actual costs are so low that in many instances the user system will have the MODEST integrated permanently to support ease of reprogramming and machine maintenance.
- Programming socket for 2716 (E)PROM

  Makes MODEST into a self-contained total development system. Strongly supports applications where the need for program changes is a prime consideration.
- Audio cassette interface

  This feature permits program loading and dumping in a low cost non-volatile medium. Thanks to object code storage there is no need for the expense of large storage media. Reprogramming by cassette supports system flexibility.
- Parallel printer interface

  Permits fast program print-out with low cost printers. Simple handshake procedure is provided.
- RS232-C interface for VDU

  Allows the large amounts of MODEST info to be displayed quietly.
- Programming on assembler level

  Avoids the tedious error-prone job of hex or machine level code writing. Cuts programming time to a fraction.
- Storage in object code

  Gives minimum memory requirements. Readable acronyms and clear text present valuble test and maintenance info.
- Disassembler for mnemonics display

  Allows program evaluation from a formatted readable display in each stage of the project.
- Forward labelling

  Pre-labelling for programming flexibility provided.
- Step-by-step operation, MODEST controlled

  Permits a display of all CPU variables, e.g. Internal Registers, Program Status Word, Address, Return Address Stack, calculated next instruction address. Manipulation of variables via keyboard. Either in RAM or PROM.
- Run operation, MODEST controlled

  System runs without display up to one of the 8 breakpoints. As soon as a breakpoint is reached the actual address and the status of the CPU variables are displayed.
- Run operation, User controlled

  System runs in real time without MODEST control up to one of the 8 software breakpoints in RAM.
- Delete, Insert and Move

  A delete, insert and move facility is provided to correct program parts with automatic address updating. This eliminates the retyping of lengthy programs.
- Transfer

  This eases the copying, loading and dumping of program words (PROM Programmer). Useful in reallocation of program segments.

**3.3 Description of MODEST**

When a user-system with MODEST is switched on or is reset, the processor commences at the starting address of the MODEST program. This is caused by the Forced Branch

Generator which disables all the system program and generates a jump to the starting address of MODEST. MODEST always starts in the ENTRY mode. On the CRT screen "MODEST ENTRY:" is displayed.
MODEST has 10 modes. Each mode is selected by typing the relevant select character.
When an unknown character is typed, MODEST displays the 10 mode select characters: M-A-L-I-P-C-D-E-S-T. When a mode is selected, MODEST displays the name of the modes and asks the user to enter the start address, source or destination address, and all the variables to be used in that relevant mode. All information generated by MODEST, e.g. program listing and error messages, can be displayed on the screen of the CRT or can be printed on a parllel printer.

### 3.3.1 MODEST modes

**Move:** A block of instructions bounded by two addresses is moved and inserted from a destination (start) address. All addresses in this block and instructions referring to this block are updated. Erroneous relative instructions are listed on the screen or on the printer.
**Alter memory:** Permits the user to modify hexadecimal program values in RAM directly. This is a handy repair mode for small program errors or program updating.
**List program:** In this mode, a listing of the stored program from RAM and/or (E)PROM can be displayed or printed. The disassembler displays the program (bounded by two addresses) in clear text and is properly formatted. An example of a listing is given in fig. 3.

The listing starts with the current address followed by the object code in hex numbers, instructions in mnemonics, the associated address and, when present, a name of a subroutine. When the program is stored in object code, each character of text takes one byte of memory space.
To reduce the need for additional text, each subroutine can start with a name. Each call to that subroutine, results in the display of that name in the comment field.
To make some difference between instructions, text, subroutine names, data fields and address fields, MODEST makes use of pseudo-object codes. These are codes not known by the 2650 but only recognized by MODEST. This enables MODEST to give a conveniently arranged listing of the program using instructions, data blocks, address blocks and ASCII information.
**Insert:** A block of instructions bounded by two addresses is copied and inserted at a destination address block. All the addresses in this block are updated. Instructions referring to this block are not changed.
**Program entry:** In this mode, after the first RAM address has been allocated, the mnemonics can be typed on the keyboard. MODEST assembles these mnemonics and stores object code in the MODEST RAM buffer. The disassembler in MODEST supplies the video screen with the mnemonic together with the current address, the updated and associated addresses, object code and, if applicable, the title of the relevant associated address.

Finding the displayed program line correct, the programmer can enter the object code in the USER-RAM and go to the next line – the address will be incremented automatically. In case an error has been made, the line can be deleted (DEL key) and, by keying CR, rewritten.

It is possible to enter sub-routine titles, user text, data or address constants directly via the keyboard. Forward labelling is initiated by typing @ instead of the absolute address and declaration at the effective address. Entering incorrect mnemonics or symbols will cause MODEST to issue an acoustic warning (beep).

After the last instruction is finished, the "END" statement terminates this mode and MODEST returns to the Entry mode.
**Calls list:** This mode allows a presentation of a list of instructions of a program or part of a program referring to a block of instructions bounded by two addresses. This is a useful mode to get a listing of all the instructions in a program that call a subroutine or use some scratch RAM.
**Delete:** A block of instructions can be deleted by using this mode. When there are instructions referring to this block in the program, these instructions are listed on the screen or on the printer.
**Execute:** In the testing mode of MODEST there are three possible submodes:
– step by step operation, MODEST controlled
– run operation, MODEST controlled
– run operation, User controlled
a short description of thes modes is given in the feature list.
**Subroutine list:** As each subroutine with a name is preceded by a pseudo-opcode, MODEST is able to make a list of all these subroutines. This list can be displayed or printed. The name of the subroutine is shown, as well as the call address.
**Transfer:** A block of instructions bounded by two addresses is copied and moved to RAM position. The old RAM contents is overwritten.
This mode is also used to program a PROM or to read the contents of the PROM and store it in RAM. In all transfer functions, the addresses can be reallocated to any memory address.

```
MODEST ENTRY :L
LIST PROGRAM
FROM: UPTO:
Ø4ØØ Ø41D SP
PRINT ? <Y/N/CR>: N
Ø4ØØ Ø4 3F     LODI,RØ 3F
Ø4Ø2 Ø7 17     LODI,R3 17
Ø4Ø4 Ø5 5Ø     LODI,R1 5Ø
Ø4Ø6 3F 7E 93 BSTA,UN        7E93    TO SCREEN
Ø4Ø9 F9 7B     BDRR,R1        Ø4Ø6
Ø4ØB FB 77     BDRR,R3        Ø4Ø4
Ø4ØD 3F 7D 29 BSTA,UN        7D29    KEYBOARD
Ø41Ø C1        STRZ,R1
Ø411 Ø4 1A     LODI,RØ 1A
Ø413 3F 7E 93 BSTA,UN        7E93    TO SCREEN
Ø416 Ø1        LODZ,R1
Ø417 1B 69     BCTR,UN        Ø4Ø2
```

Fig. 3. Program Listing of MODEST.