Hardware Support for Testability, Debugging, and Better Performance

Wolfgang Matthes, 1989

Abstract

Design recommendations are given and low-cost hardware enhancements are suggested to facilitate testability, debugging etc. in multimicroprocessor systems. A typical example is a program tracing capability in microcomputer modules. Performance augmentation allows for retaining the inherent advantages of microcomputers (self-test capability etc.) in a performance range which would otherwise require genuine dedicated hardware.

1. Introduction

The paper is concerned with design recommendations and low-cost additional circuitry for multimicroprocessor systems and microcomputer modules of such systems. The particular topics are:

- Error handling within the whole system
- Testability provisions in a microcomputer module
- Program tracing in a microcomputer module
- Performance augmentations of microcomputer modules, with the primary goal to avoid genuine special hardware whenever possible.

The recommended principles and hardware structures are well-proven in a few hundred systems which are in everyday use around the clock, namely in service processors for data processing systems (one example has been described in [1]).

2. Error handling within the whole system

To facilitate in-operation error handling in a multimicroprocessor system, one microcomputer module should be devoted to this task. This module should be able to monopolize the system bus, to prevent all other modules from trying to get bus mastership, to log and display errors, and to execute the test and maintenance software. In an example system (Fig. 1), this microcomputer is directly connected with the centralized circuitry of the system bus (timing, arbitration control, watchdog). This module is called BUS CONTROL. The system bus contains a NES (NON EXECUTIVE STATE) line which can be energized by BUS CONTROL. (Refer to [2] resp. [6] for a system bus specification; however, the following recommendations may be

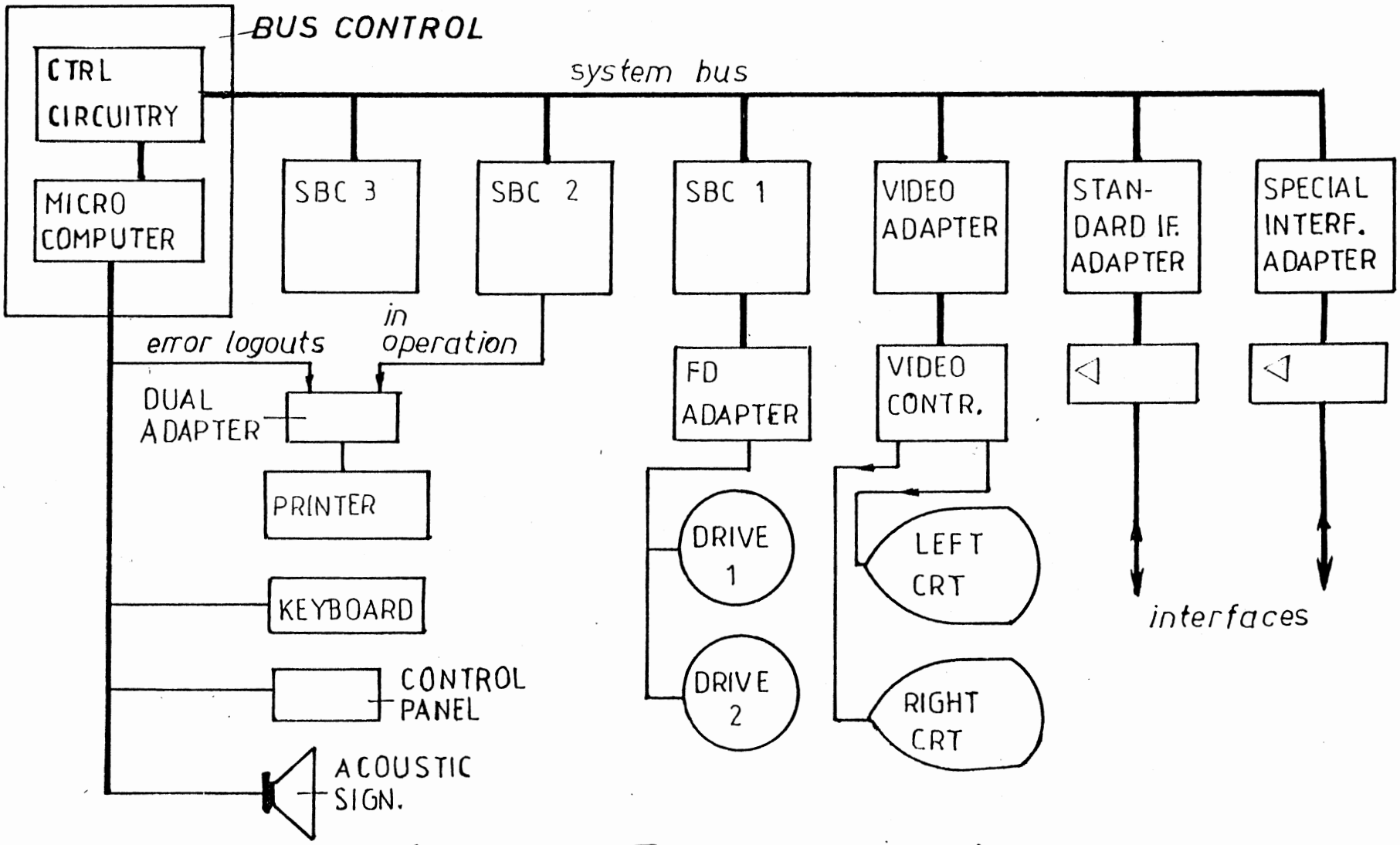
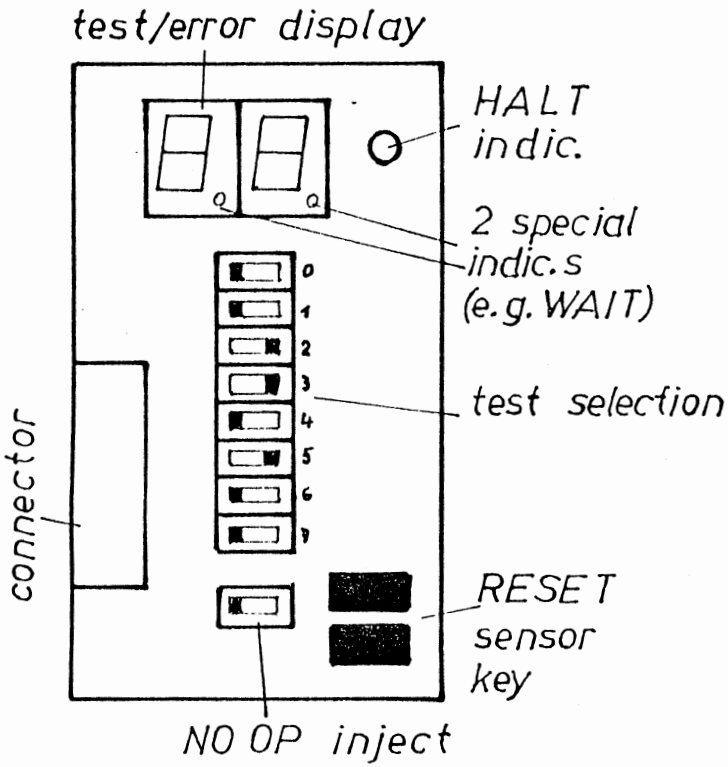


Figure 1. Example system

applied to nearly all multi master bus systems, with introduction of a few additional lines, if necessary.) The effect of NES is to cease system bus operations in all other modules. With NES line activated, BUS CONTROL can inspect all other modules. The system bus is also equipped with special lines for error signalization (ERROR, SLAVE ERROR), but a local error handling within each module and cyclic inspection of error latches via the system bus has shown to be the more cost-effective solution. An other important feature is the capability to reset each module selectively. For this purpose, the system bus protocol provides a selective reset sequence: BUS CONTROL issues a slave access to the module to be reset and energizes a SELECTIVE RESET bus line. Immediately after this reset, parameter bytes (e. g. for test routine selection) will be transferred into dedicated RAM locations of the corresponding module. Thus BUS CONTROL is capable of initiating test routines even in presence of severe failures (in contrast, test initiation by a software protocol would require more hardware of the module under test to be free of failures).
During the error-free operation of the system, BUS CONTROL is loaded only with the cyclic inspection of the error latches in the other modules. Hence some additional workload could be assigned to it. A good example is the control of low-speed I/O devices, like printer, keyboard, some special keys, lamp indicators, and acoustic signalization. These devices are also useful for malfunction indication, test sequence control, and error logout.
An other recommendation is to make provisions for easy module interchangeability and module removal so that a service engineer can quickly locate suspected modules by a simple "interchange and try again" or "remove and try again" procedure. Some key points: application specific software should be kept in RAM, with initial program loading from diskette, from a centralized ROM module etc.; programs should be able to identify the own module and the system configuration; the system bus should be operable in spite of removed modules.

3. Testability provisions in a microcomputer module

Microcomputer modules should be equipped with sufficient ROM-resident diagnostic software and with some simple means for test selection and indication. One example is a diagnostic adapter with a few switches and L.E.D. indicators (Fig. 2). All microcomputer modules of the system have a special connector to accept this adapter. After hardware reset, ROM-resident test software checks the cause of the reset (power on, selective reset from BUS CONTROL, manual reset from diagnostic adapter). In case of a selective reset, sequence control information will be taken from dedicated RAM locations; in case of a diagnostic adapter reset, the program will read its switch positions and branch to the appropriate test routine.
In some systems error detection circuitry is necessary (e. g. RAMs with parity or ECC, watchdog hardware, memory protect circuitry etc.). It is strongly recommended to provide means



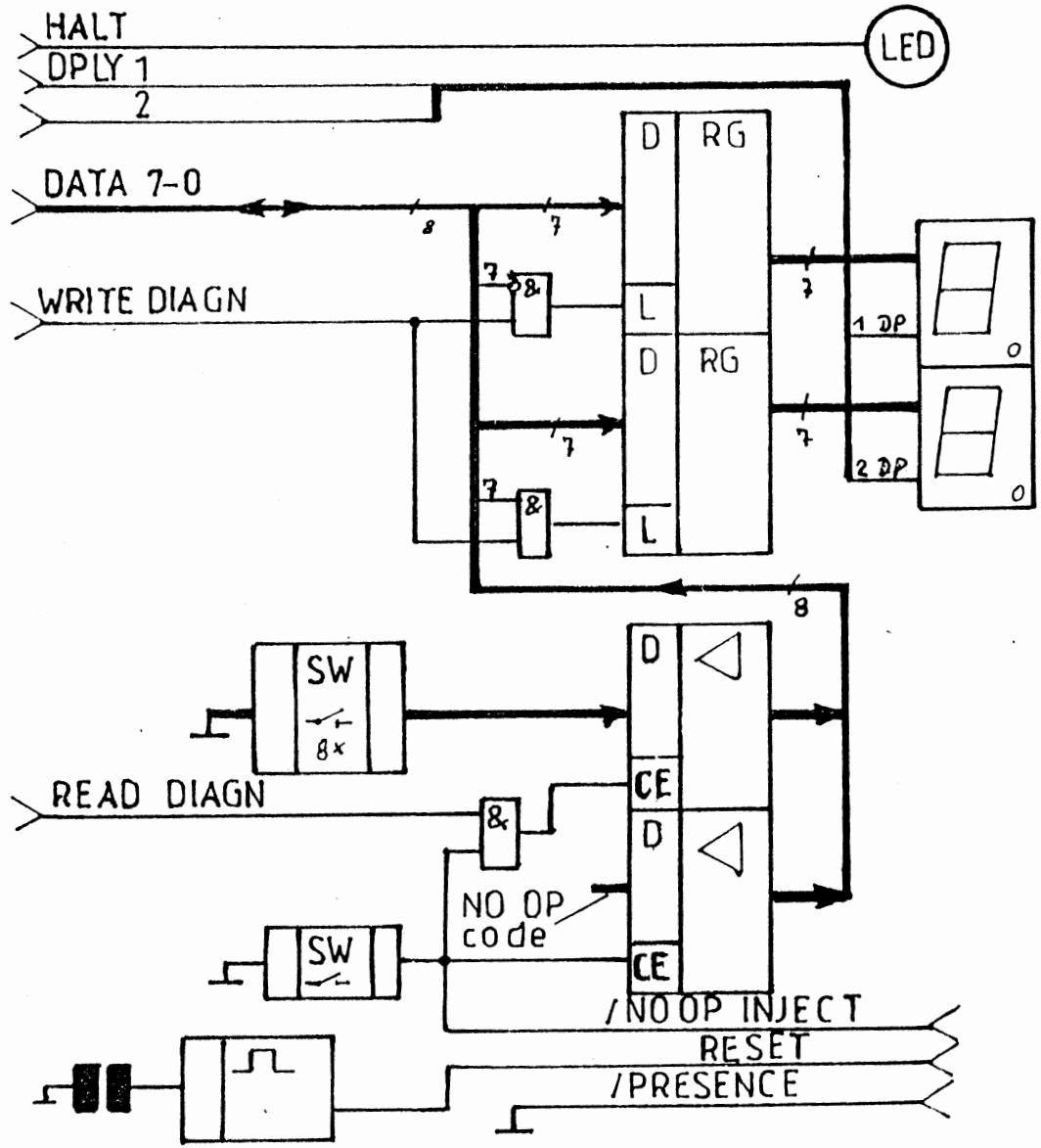


Figure 2.
Diagnostic adapter

for the self-test of this hardware (examples: provide for write-in of inverted parity, make ECC RAM bits directly accessible for write and read, make watchdog circuitry diagnosable etc.). Such diagnostic modes can be controlled by a loadable DIAGNOSTIC MODE register (see [2], [7]).
Signature analysis and oscilloscopic tracing have found to be useful in field troubleshooting. These principles require to follow some hardware design recommendations ([3]): provide test loops in ROM; provide at least one programmable source of SYNC- resp. start/stop- pulses; provide a source of cyclic reset pulses with known clock count between the resets; follow the known recommendations for S.A. loop testing (e. g. according to [10]).
Large DRAM memories are susceptible to soft errors. Known ECC provisions require many additional bit positions and may be too costly in some systems (e. g. in 8-bit- modules, where at least 4 additional bits must be provided). An alternate approach uses only a single parity bit and an inexpensive hardware sequencer to achieve a redundant (duplicated) storage without software intervention (Fig. 3; [4], [8]). A good application example is the modernization of older microcomputer modules, where the effective 2-fold increase in memory capacity instead of a 4-fold increase is quite acceptable. A disadvantage is, however, that the CPU must be kept in wait states at least during the read cycles.

4. Program tracing in a microcomputer module

An inexpensive approach to hardware address compare stop or instruction step requirements may be based on an additional RAM bit position in the memory. During the normal operation of the system, this position may be used for parity checking. In case of a memory with ECC provisions, at least one of the ECC bit positions may be used for that special purpose. Also needed are means to activate NMI (or a similar highest-priority interrupt) and a program-loadable control register (Fig. 4; [2], [4], [5]). This register should provide code positions for the following modes:

a) Conventional mode. Parity resp. ECC is enabled. According to Fig. 4, ENABLE PARITY CHECK and ENABLE ERROR SIGNALIZATION are active. In all write cycles, the parity bit will be written. In all read cycles, the memory parity will be evaluated.

b) Compare stop mode. Write access to the additional bit(s) is inhibited. A "1" read out of the corresponding bit will activate NMI. ENABLE CMP STOP is active. Additional control register bits may be set to specify the stop condition (STOP ON READ, STOP ON WRITE).

c) Compare stop setup mode. Neither compare stop nor parity resp. ECC are enabled. ENABLE INJECT is active. In write cycles, the value of the INJECT HI position of the control register will be written into the compare stop bit RAM position. A simple block move (read and write again) of all RAM

locations (with ENABLE INJECT set and INJECT HI cleared) is sufficient to leave the conventional mode. A particular stop condition can be set resp. removed by a read - write access with ENABLE INJECT set and INJECT HI set resp. cleared.

d) Conventional setup mode. Parity resp. ECC bits are written, but in read cycles error signalization is inhibited. Only ENABLE PARITY is set in the control register. A simple block move (read and write again) of all RAM locations will establish the conventional mode.

e) No-effect mode. The control register is cleared completely. The additional memory bits are neither affected nor evaluated. This mode is used by the diagnostic software.

The hardware requires only a few MSI parts, but a very flexible and elegant program tracing is possible. Stop conditions can be set not only for a single address, but for arbitrary areas in programs or data segments.

5. Performance augmentations of microcomputer modules

In many real-time applications the performance of microcomputer modules is quite sufficient, with exception of a few time-critical operations, especially with respect to the control of real-world interfaces. Dedicated hardware modules are hardly to design and to debug. Hence it is useful to have some design options to improve the performance of microprocessors for such operations.

5.1. Control principles

The user of a microprocessor CPU cannot modify its instruction set. Hence all additional effects require some circuitry outside of the CPU. For some applications, coprocessors are available which extend the performance for a class of operations. The following suggestions are not concerned with coprocessor-like delegation of software loops to dedicated autonomous hardware, but with the extension of the effects of some CPU instructions. The typical speedup is in the range 1:2...1:20.
In some configurations, special effects can be controlled by high-order address bits which are not used for memory addressing. Appropriate suggestions can be found in [4].
Fig. 5 illustrates an other principle. Additional memory (called control memory) is addressed parallel to the instruction memory during CPU instruction fetch. Its content is loaded into a control register. In this way the width and hence the effect of some CPU instructions can be extended.
An instruction fetch within a special address range (e. g. with some high-order address bits set) will cause the control register to be loaded and the special mode of extended effects (called CONTROL MODE) to be set. Each instruction fetch within an other address range will cause CONTROL MODE to be reset and the control register to be cleared. It is not necessary but

possible to provide conventional access to the control memory by the CPU. A cleared control register should cause no special effects, therefore instructions fetched within the said special address range will have no special effects if the corresponding control memory locations contain zeroes. This allows instructions with and without special effects to be intermixed freely, without any performance degradation which would otherways be inevitable due to the necessary mode switching.
In the following sections, a few application examples will be described. Fig. 6 shows some details. For more details, circuit diagrams, and discussion of intricacies, refer to [4].

5.2. Extended output

a) output of register contents

In CONTROL MODE, the CPU address and data lines can be used together for output purposes (e. g. an 8-bit- CPU can drive outbound interfaces with up to 24 data lines). The execution of a register-to-memory move instruction (e. g. Z 80 LD (HL),A) is modified. All memory write pulse and chip enable lines are deactivated. Instead of being written into memory, the information on the CPU address and data lines will be moved to the interface.

b) output of immediates

Parts of the control word (e. g. an EMIT field) can be directly delivered to interface lines (other parts of the control word can contain a destination code, if required). The immediate part of the control word can be augmented by immediates of appropriate "Load Immediate"- instructions (e. g. Z 80 LD HL,nm). The width of the interface can be extended to a great number of bits (e. g. 16 out of the immediates m, n of a LD HL,nm instruction and 24 out of the EMIT fields of the 3 control words accompanying the 3 bytes of the instruction, thus the good old Z 80 can deliver 40 bits in 11 machine cycles).

5.3. Extended input

Bit strings exceeding the CPU word size can be read by data substitution on the CPU data bus. Most CPUs have load instructions which load an immediate twice as long as word size into a CPU register (e. g. Z 80 LD HL,nm). In CONTROL MODE, the fetch of the 1st part of the immediate is accompanied by a 1st control word. This causes the immediate to be removed from CPU data bus (e. g. by disabling the memory bus driver) and a bus driver for the first part of the inbound interface to be enabled. The fetch of the 2nd part of the immediate has a similar effect. Thus the CPU will receive the information from the inbound interface instead of the immediate.

5.4. Conditioned branches

The principle of substitution on the CPU data bus can be used to provide conditioned branches or subroutine calls resp. conditioned skipping of instructions according to external

conditions. An unconditioned branch, call, or other instruction to be skipped conditionally is accompanied by control words which select the external condition. The control hardware decides whether to execute or to skip the instruction. In the latter case, the instruction is removed from CPU data bus and substituted by the NO OP code according to the CPU instruction list. The hardware must make the decision in the first cycle of the instruction and keep the result until the last cycle has been executed. If necessary, wait states must be inserted during the decision interval. The hardware must suppress all interrupts during the substitution (the substituted NO OP sequence may never be interrupted). With such provisions it is possible to inspect many external conditions directly (e. g. 256 or even more) wich would otherwise require a longer instruction sequence (output to select the condition -> input of the selected information -> bit test -> branch).

5.5. Loop simplification

In many critical software loops, much time is spent for the analysis of abnormal and ending conditions. This code could be substituted by hardware. The detection is done with external circuitry under control of control words. Examples are: parity checks at external interface lines, occurrence of special bit patterns on interface lines (e. g. escape characters), access to the last position in the corresponding memory area etc. The condition could be signalized by an interrupt with sufficiently high priority (e. g. a NMI).

6. References

[1] Marschner, R.: Bedien- und Serviceprozessor EC7069.M. Rechentechnik und Datenverarbeitung, 8/1984, p. 8 ff.

[2] Matthes, W.: Multimikrorechnersysteme. Series in: radio fernsehen elektronik, 4/1984...12/1985.

[3] -: Diagnostik und Wartung an Multimikrorechnersystemen. radio fernsehen elektronik, 8/1986, p. 518-521, 9/1986, p. 566, 567.

[4] -: Ergänzungsschaltungen für Mikroprozessoren. IIR Informatik Informationen Reporte. Institut für Informatik und Rechentechnik, Berlin (in preparation).

[5] WP 154 244. Speicheranordnung mit Fehlererkennungs- und Diagnoseeigenschaften, vorzugsweise für Mikrorechner.

[6] WP 156 743. Bussystem zur Verbindung von logischen Funktionsmoduln.

[7] WP 159 916 resp. EP 00 67 982. Mikrorechneranordnung, vorzugsweise für den Einsatz in Multimikrorechnersystemen.

[8] WP 246 857. Speicheranordnung mit Fehlerkorrektur, vorzugsweise für Mikrorechner.

[9] WP 246 858. Mikrorechneranordnung mit erweiterten Steuerwirkungen.

[10] A Designer's Guide to Signature Analysis. Hewlett-Packard Application Note 222.

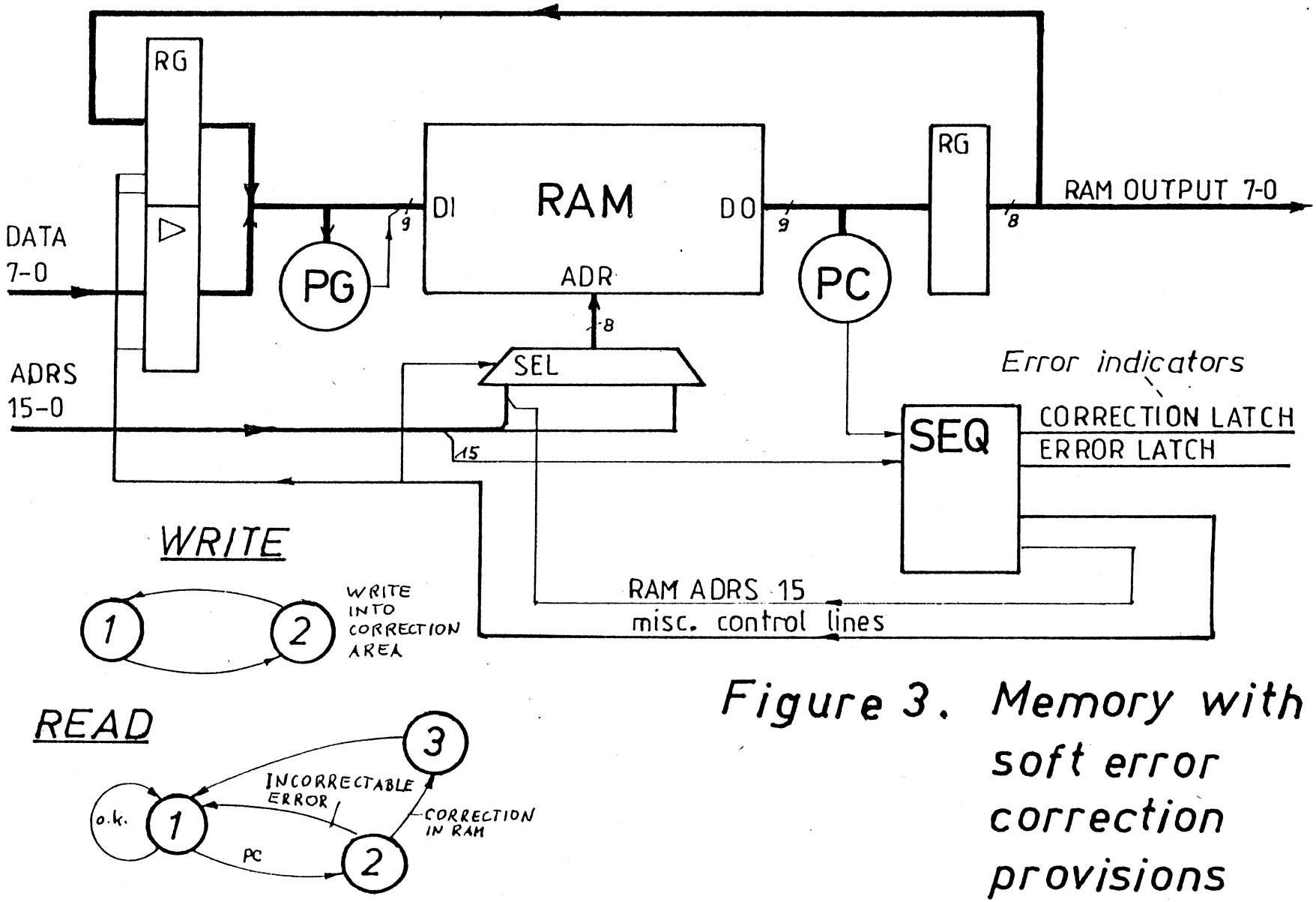


Figure 3. Memory with soft error correction provisions

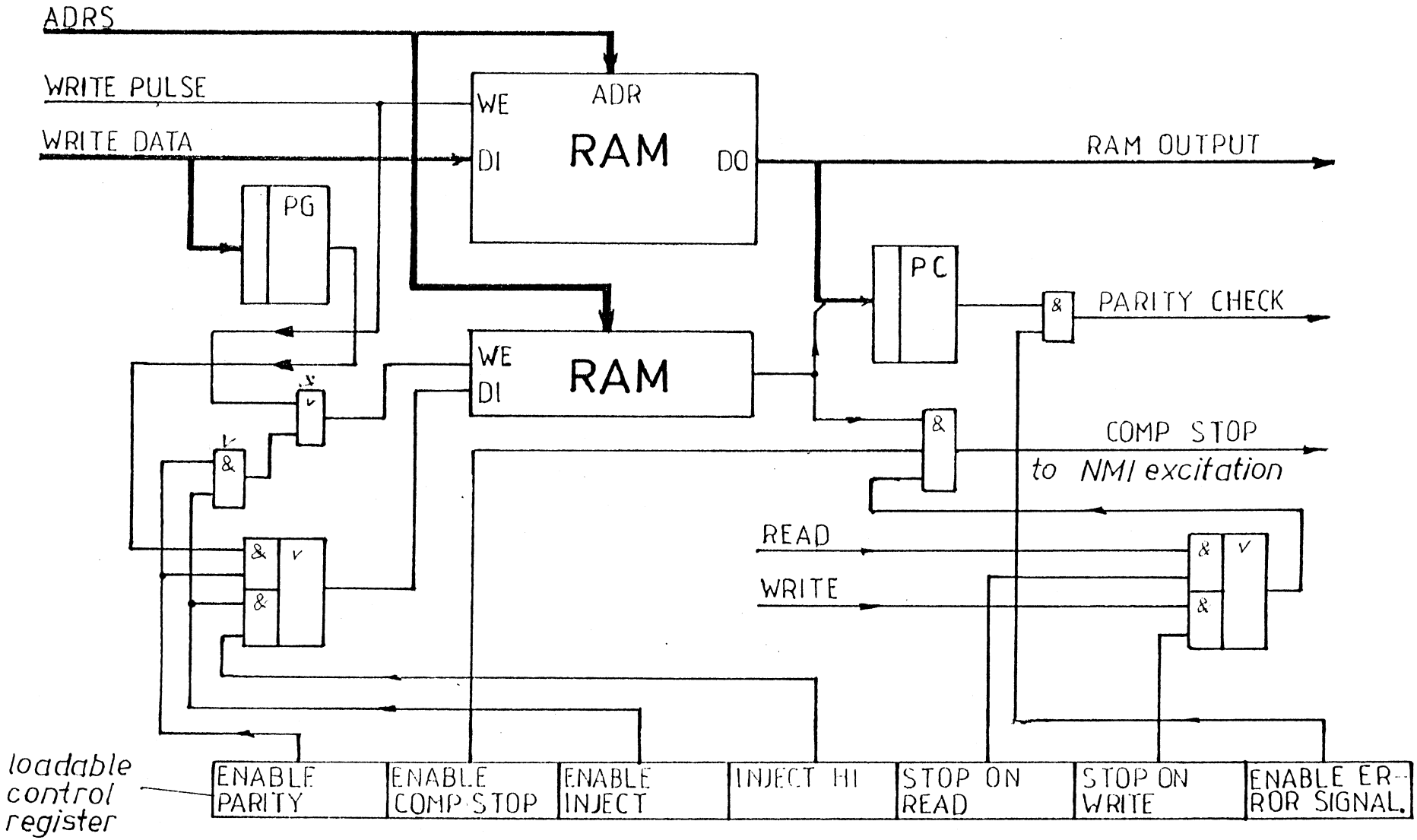


Figure 4. Memory with compare stop provisions

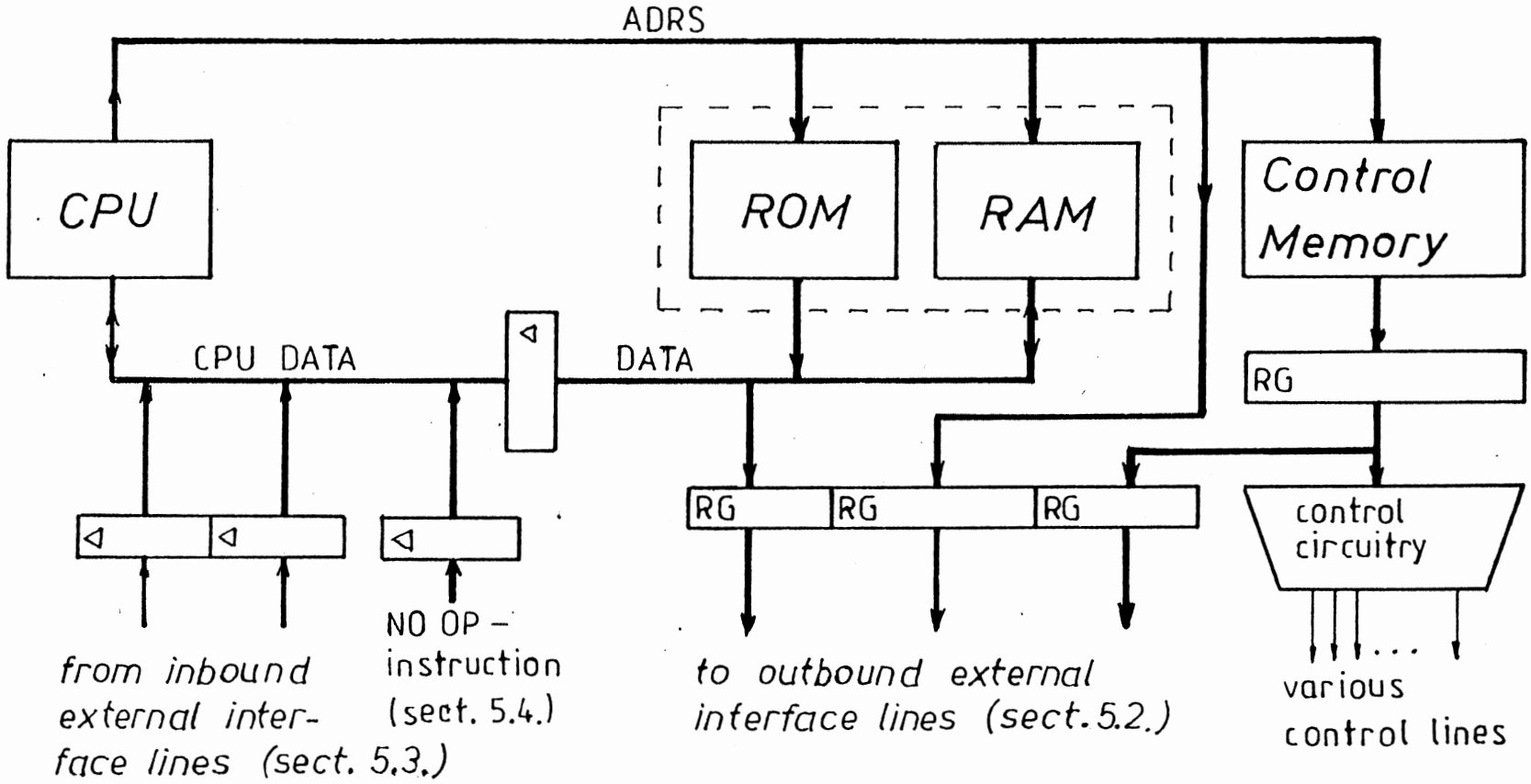
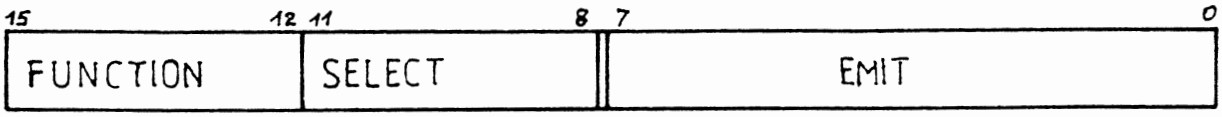


Figure 5. Microcomputer with extended control effects

Example control word format



Z80 instruction LD HL,nm accompanied by control words

Instruction memory

21H
m
n

Control memory

1st control word
2nd control word
3rd control word

Control register details

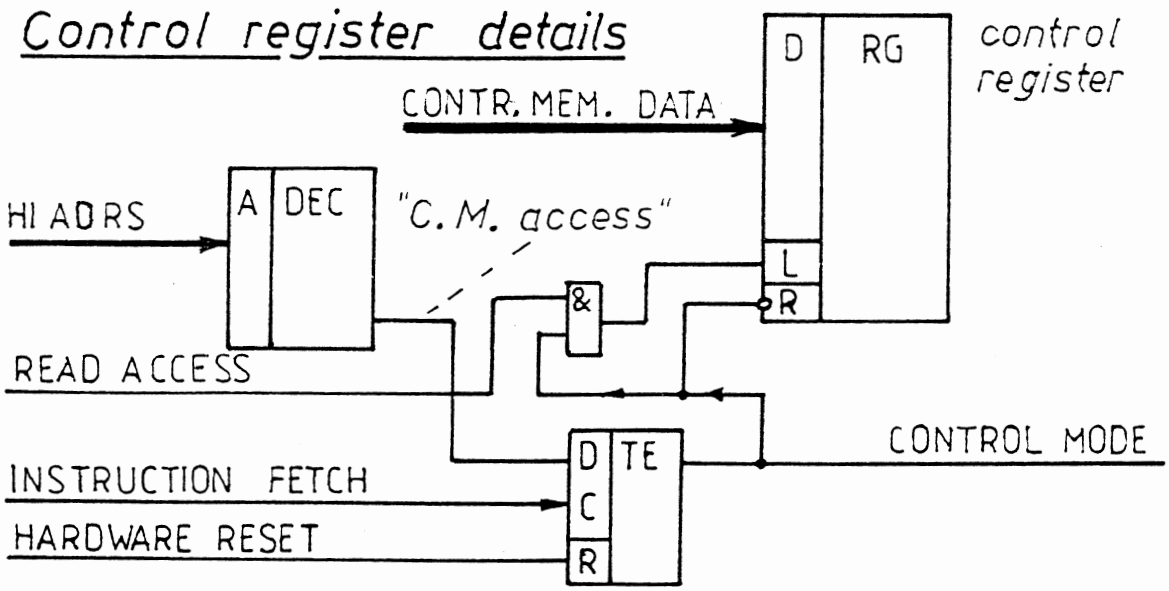


Figure 6. Control mode details