A scientific computer — 1

Powerful design uses two microprocessors and BURP, a new high level language

by J. H. Adams, M.Sc.

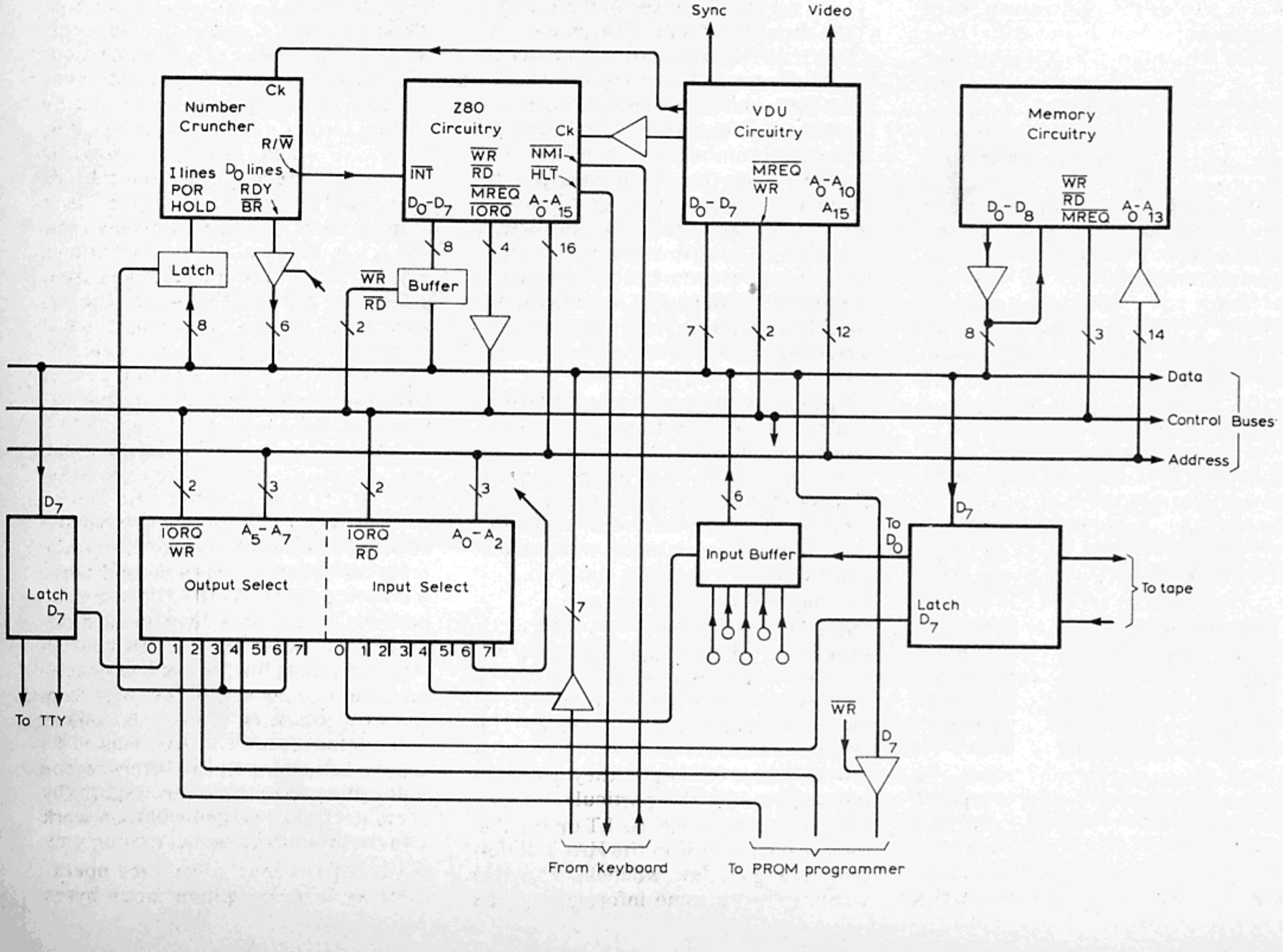
This series of articles describe a complete scientific computer which is based on two microprocessors. Unlike conventional mini and microcomputers which rely on large and expensive memories for complex subroutines, the present design uses a Z80 to handle the general processing and leaves the "number crunching" to a MM57109 microprocessor. This second device contains all of the algorithms necessary to execute standard mathematical functions. The basic design uses 8K of memory although this can be easily expanded to 32K.

The series will also describe tests and diagnoses, together with the computer's operation, both in machine code and the high level language BURP. Games, mathematical and financial programmes will also be given, and the series will conclude with several options including graphics and symbolic displays, graph plotting and an e.p.r.o.m. programmer.

A COMPUTER, to quote the dictionary, is an "apparatus for making calculations or controlling operations that are expressible in numerical or logical terms". With the advent of the microprocessor it has become possible to build low cost computers but, as industry has demanded, the majority of these devices are designed with the controlling-operations aspect of computing as their particular forte. Therefore, as numerical calculators, they are rather limited, and this has led to many disappointed owners of development kits who find their idea of microcomputers and that of the kit manufacturers are poles apart.

To meet the needs of such buyers, i.e. a system that can do sums and communicate in a language based upon English, various versions of the language BASIC (Beginner's All-purpose Symbolic Instruction Code, a programming language devised in 1964 by the Americans Kemeny and Kurtz) have been produced, ranging from Integer Basic, the mathematical applications of which are limited to computer games and the like, to quite complex versions which are able to handle floating point numbers and to perform several mathematical and trigonometrical operations on those numbers. These operating systems need computers with large memories, not for running large pro-

Fig. 1. *Block diagram of computer. Control and data buses are buffered as they leave the Z80, while buffering for the address bus is in the memory circuitry. Five buffered serial inputs and one buffered output are available for external add-on circuitry, together with three output and five input select signals which may be used to latch data off or buffer it on to the data bus.*



Specification

Commands available

LOAD	Loads program lines into memory.
ADD	Adds program lines to those already in memory.
DEL n	Deletes program line n.
RUN n	Runs from program line n.
MOD*	Converts Print statements to Write and vice-versa. (For use with second output device.)
LIST n	Lists from program line n.
DUMP n	Lists on second output device.

Statements available

Input, Print, Write, For, Next, Goto or Go, If, Then, Gosub, Return, Top Erase, Halt, Let.

Write is the print command for a second output device, such as a teleprinter.

Top clears the top line of the v.d.u. and sets this as the next printing position.

Erase is similar to Top, except that the whole screen is cleared.

Halt stops execution until any key, except FS or RS is depressed.

Mathematical capability

Calculates to 8 figures plus 2 exponent digits.

Functions

+, −, ÷, square, square root, log, 10^x. ln (nat log), e^x, sin, cos, tan, $\sin^{-1}$, $\cos^{-1}$ $\tan^{-1}$, y^x, 1/x, π, degree to radian conversion and vice-versa.

Variables	26, denoted by A to Z.
Range	10^{-99} to 9×10^{99}.
Program lines	Up to 254, or the limit of the read/write memory.

Print capability

Automatic switching to scientific mode on results greater than 10^9 or less than 10^{-4}.

Printed figures may be tabulated or close-packed, to any number of decimal places from 0 to 7.

Automatic rounding occurs on results abbreviated in this way.

Alphanumeric data may be interposed with printed variables.

Input/output

Input via ASCII encoded keyboard.

Output via v.d.u. of 32 lines, 64 char/line. Separate video and sync signals to 625 standard.

Optional output to teleprinter.

300 baud f.s.k. input/output, using tones of about 1200Hz and 2400Hz, for the storage and retrieval of data via a tape recorder.

---

grams, but for storing the mass of information required to instruct the logic oriented microprocessor on how to behave as a number oriented device.

The aim of this project was to produce a computer with extended mathematical capabilities and avoid the need for such heavy investments in memory i.cs. This has been achieved by using two processors, the Z80 standard microprocessor which takes the dominant role as the processor of data moving around the system, and the MM57109, a number-oriented processor which handles the calculations. The Z80 has been well covered in this and other journals, but the MM57109 may be less familiar. This device appears to be a not entirely successful transplant of a scientific calculator chip into the world of data buses and memories. It can perform most of the standard scientific calculator functions and, in common with many such devices, it uses a sequence for instructions known as Reverse Polish Notation. This system differs considerably from the standard algebraic notation, and is based upon the logical idea that the instructions to be performed or, in the case of a calculator, keys to be pressed, should be listed in the order which they are to be performed. For example, consider the calculation $c = \sqrt{(3^2+4^2)}$. The first operator following the equals sign is the last operation to be carried out and yet the last, brackets, is not the first. The actual order of execution is in fact quite complex, and the more complex the expression to be solved, the worse things become. The algebraic sequence would actually be 3, sq, move to memory, 4, sq, +, memory recall, =, root. In algebraic BASIC, the computer line would be, LET C = (3↑2 + 4↑2) 0.5 where ↑ means raised to the power of.

However, a simple RP calculator would execute the operations in the sequence which the operator would follow, ie, 3, sq, store, 4, sq, recall, +, root. In practice this is even simpler because calculators and the MM57109 have a stack of registers, each capable of holding a number. In the MM57109, this stack consists of four registers called X, Y, Z and T for top. Data enters and leaves via the X register, but may be pushed up into the stack either for temporary storage, or to take part in a two number operation involving the contents of the X and Y registers (e.g. YX which calculates the Y number to the Xth power). There are specific instructions which move or exchange the contents of the stack registers to facilitate calculations; however, the system ensures that in normal use of the language, numbers are pushed or entered into the stack as and when necessary. As RP is a step-by-step system, no brackets are required, and in the expression originally considered the RP BASIC version of the computer line would be

LET C = 3 SQ 4 SQ = ROOT

Some other examples of calculations and the stack operations are given in table 1. In my view, having used both types of notation, the Reverse Polish wins every time. For this reason a new language was formed for the computer Basic Using Reverse Polish or BURP.

Hardware

The block diagram of the computer in Fig. 1 follows standard microcomputer techniques, with an eight wire data bus, a sixteen wire address bus and a four wire control bus which interconnect the various elements of the computer. The majority of lines in this design are active low, ie, for an i.c. output, they go to the low state when the particular output label is occurring, e.g. HALT on the Z80 goes low when it is in the HALT condition, WR goes low whenever the Z80 wants to write some information into the memory. With an input, that input must go low for the input label to occur, e.g. $\overline{\text{INT}}$ needs to go low for the Z80 to be interrupted. An active low label is identified by a bar over it.

To describe the operation of the c.p.u. shown in Fig. 2, it will be helpful if some aspects of microprocessor operation are discussed. The Z80 can execute a repertoire of 158 groups of instructions, of which there are about 600 in total, and these instructions are read in through the data bus of the system as 8-bit words or bytes. The reading in, or writing out of bytes from or to memory locations or input/output devices is controlled by the four processor output lines RD, $\overline{\text{WR}}$, $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$. For example, a low RD and a low MREQ output from the Z80 indicates that it wants to read in a byte from a memory location, whereas a low WR and IORQ means that it is writing a byte out along the data bus to an output device, such as a teleprinter. The address of the memory location, which is stored in a 16-bit register within the Z80, known as the program counter, or code number of the input/output device, is simultaneously sent out onto a second 16-line bus by the Z80. External circuitry selects which memory location or device is coupled to the data bus for that particular Z80 operation. When pin 26 RESET, of the Z80 is taken to 0V, the program counter is cleared, and when pin 26 returns to 5V, the Z80 begins by reading in the data byte in memory location 0, executing it as an instruction, increasing the program counter by one, and reading in the next byte from memory and so on. Thus, the memory will contain lists of instructions to be executed sequentially, interspersed with bytes of data which are required by some of them. The Z80 will then work its way through these lists or programs.

The instructions cover such operations as LOADS, which move bytes between registers within the Z80 or the memory. Logical and arithmetic functions, usually on the contents of the A register of the Z80. JUMPS, which, by feeding two new 8-bit bytes into the program counter register, cause the sequence of instruction execution to jump to a different point in the program. And CALLS, which are similar to jumps except that the old program counter contents are kept in a last-in first-out store in the read/write memory of the system, to be restored to the program counter on execution of the Z80 instruction RETURN. Calls are particularly useful whenever a certain block of instructions need to be used at several points in a program. If the instructions are written once in a program with a return instruction at the end, the block may be called at any point during the rest of the program. Such blocks are known as subroutines.

One feature of most microprocessors, including the Z80, is that CALL instructions may also be forced into the instruction sequence by activating either of the pins $\overline{\text{NMI}}$ or INT. The subroutines called by these interrupts are executed immediately after the instruction in progress, and, as with most CALLS, once the subroutine has been completed, instruction execution recommences at the point where the sequence was originally interrupted. These interrupts are generally used by other devices that want to communicate with the Z80 and, in this case, the $\overline{\text{NMI}}$ interrupt is initiated by the strobe pulse from the keyboard and hence by the depression of any key. The keyboard subroutine reads in and acts upon the keyboard data before returning control to the main program as shown in Fig. 3. This is just one method of using the keyboard and another common approach is the polling system where, as part of the main program, the Z80 reads in the strobe pulse as part of a byte, then tests to see if the strobe is active and jumps back to the read operation if it is not. When it is active, the Z80 reads in the keyboard data byte. This method requires six bytes of instructions and memory locations, or at least a three byte CALL instruction to a read-the-keyboard subroutine whenever a byte of keyboard data is required.

In contrast, an interrupt driven system only requires the one-byte HALT instruction to be executed whenever a byte of keyboard data is needed. The only method for the processor to get out of the HALT state, once the instruction has been executed, is by operation of the reset button or by an interrupt, the Z80 waits for the interrupt which directs it into the subroutine for the keyboard. The interrupt system which was chosen, saves on memory space and the subroutine contains an extra section which will reset the entire system if the processor is not in the HALT state. If it is necessary to interrupt, for example, a program under development which has a fault, this can be achieved by pressing any key. The HALT command is also available in the high level language where it will also stop program execution until a key is pressed.

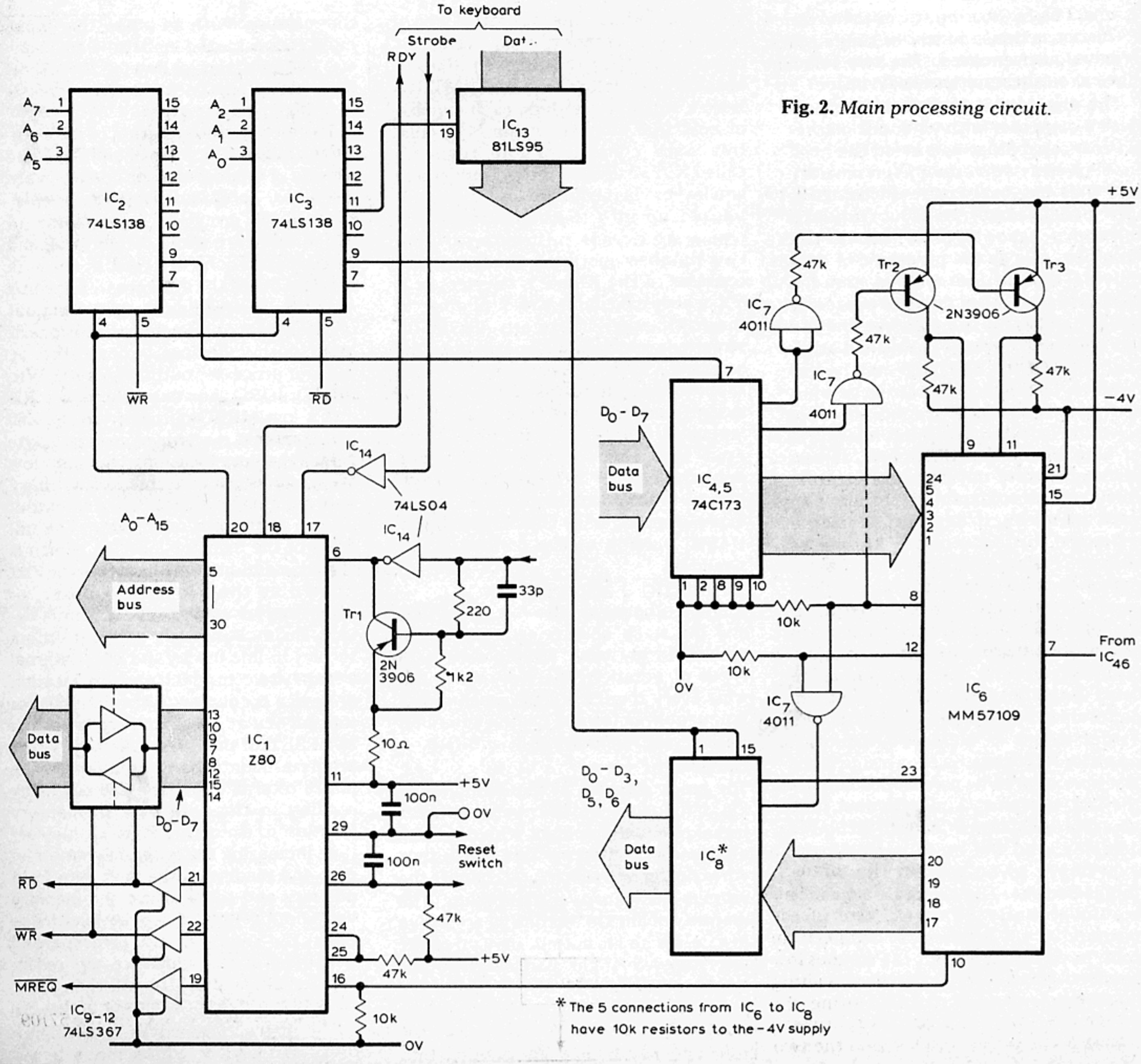


Fig. 2. *Main processing circuit.*

* The 5 connections from IC_6 to IC_8 have 10k resistors to the −4V supply

The NMI input (non-maskable interrupt) operates under all conditions and just calls to a fixed address in the memory. The other interrupt input, INT, can be enabled or disabled by instructions within the program being executed. This particular interrupt line comes from the MM57109 R/W output and indicates that the 57109 wants to send b.c.d. data to the Z80. This output can do some peculiar things during the initial reset of the processor; therefore, during this period, it is essential that the INT input is disabled. This is automatically done by the Z80 whenever it receives the reset signal. The interrupt can be programmed to respond in one of three ways, but in this system its response is similar to that of the NMI input, i.e., a CALL to a particular memory address.

Operation of the MM57109 is most easily understood by reference to the simplified diagram in Fig. 4. The device has open drain outputs which require external pull-down resistors, and some non-t.t.l. compatible inputs, but these have been left out of the diagram. The 57109 has a 6-bit input word into which 70 instructions, mostly of single bytes, have been encoded. The two relevant control lines are the RDY output and the HOLD input. RDY indicates that the device is ready to receive an instruction. If the HOLD input is low, RDY will go low after 16 μs, and the current instruction on the input lines will be executed. If the HOLD is high, RDY remains high and the operation of the 57109 is suspended until HOLD goes low. In this system, RDY is sensed, and HOLD is controlled by the Z80. The HOLD input is normally high, and the typical sequence used by the Z80 when it wants the 57109 to execute an instruction is shown in Fig. 5.

Although this sequence is adequate for the execution of most instructions, the 57109 has been designed to operate as a separate microprocessor and will therefore sometimes produce RDY pulses during the execution of certain instructions. These pulses are intended to cue memory counters etc. As the HOLD is on during the execution of instructions, these RDY pulses must be supressed otherwise the Z80 may think that it's time for another instruction to be sent to the 57109 latch. Fortunately, in such cases the output ISEL at pin 12 goes low, and this is used to gate the RDY signal to, and the HOLD signals from, the Z80 via IC$_7$.

The data read in by the Z80 via the tri-state buffer, IC$_8$, consists of the modified RDY signal together with the four DO (digit output) lines which carry the b.c.d. data from the 57109 X register, and the $\overline{BR}$ line on pin 23. This line pulses low whenever one of the seven tests that the 57109 can perform proves to be true. Pin 10, R/$\overline{W}$, goes low whenever b.c.d. data bytes are waiting to be read in by the Z80. During the execution of an OUT instruction, twelve such pulses occur which signal the two

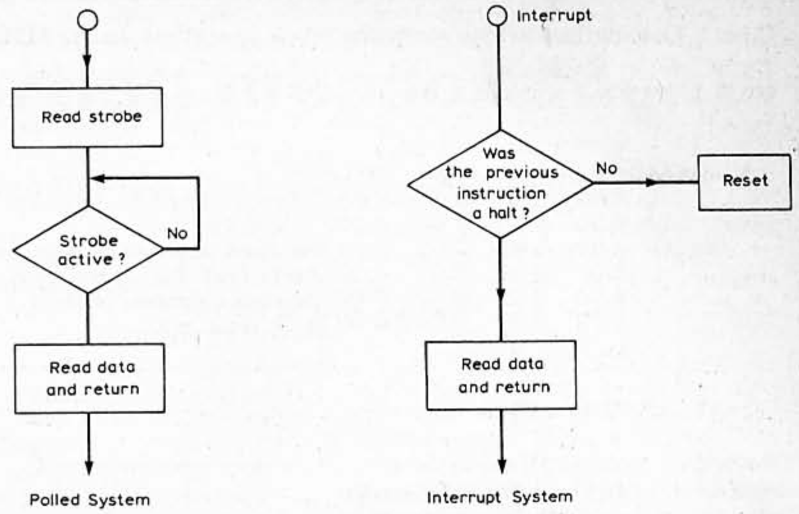


Fig. 3. *Polled and interrupt keyboard systems.*

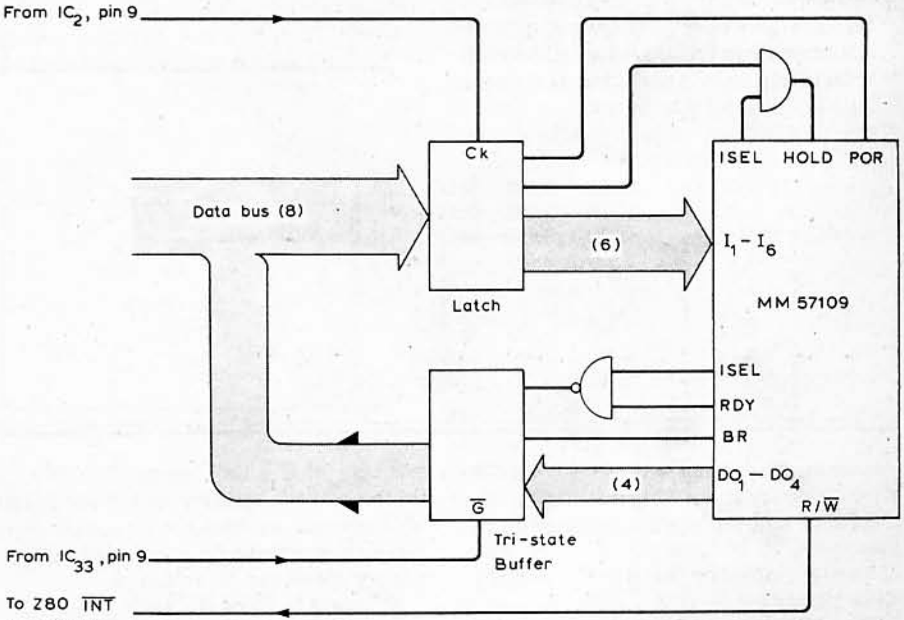


Fig. 4. *Simplified diagram of the MM57109 logic. The microprocessor also has open drain outputs which require external pull-down resistors.*

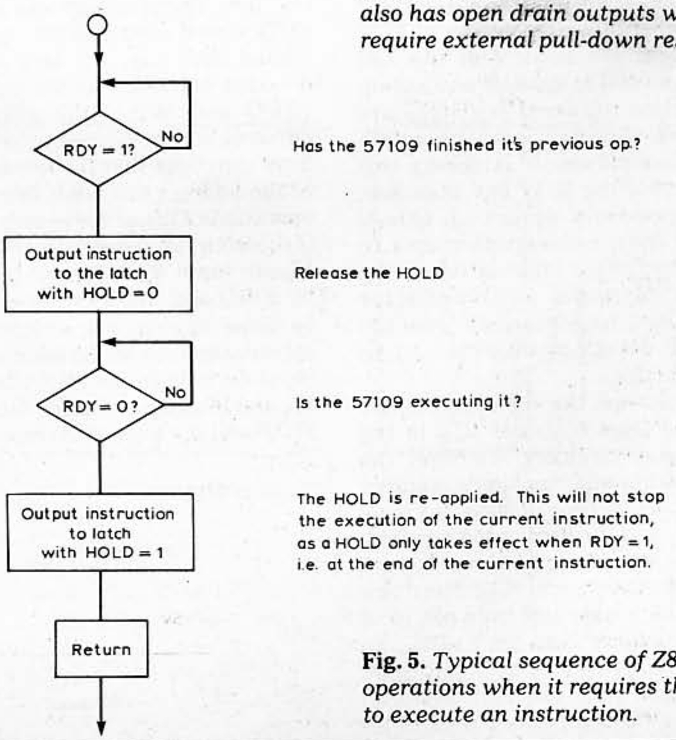


Fig. 5. *Typical sequence of Z80 operations when it requires the 57109 to execute an instruction.*

Table 1. Calculations which show the stack operations in the MM57109.

COMMAND	X	Y	Z	T	
$\sqrt{3^2+4^2}$					
3	3.00	0.00	0.00	0.00	
SQ	9.00	0.00	0.00	0.00	
4	4.00	9.00	0.00	0.00	the stack is automatically pushed, i.e.
SQ	16.00	9.00	0.00	0.00	X→Y, Y→Z, Z→T, T lost
+	25.00	0.00	0.00	0.00	the stack collapses, i.e. 0→T, T→Z
ROOT	5.00	0.00	0.00	0.00	Z→Y, result in X.
$\frac{(6+9)}{(4-1)}$					
6	6.00	0.00	0.00	0.00	
9	9.00	6.00	0.00	0.00	
+	15.00	0.00	0.00	0.00	
4	4.00	15.00	0.00	0.00	
1	1.00	4.00	15.00	0.00	
—	3.00	15.00	0.00	0.00	
/	5.00	0.00	0.00	0.00	
$\sin(1+\sqrt{(5^3-4)})$					
5	5.00	0.00	0.00	0.00	
3	3.00	5.00	0.00	0.00	
YX	125.00	0.00	0.00	0.00	Y to the Xth power
4	4.00	125.0	0.00	0.00	
—	121.00	0.00	0.00	0.00	
ROOT	11.00	0.00	0.00	0.00	
1	1.00	11.00	0.00	0.00	
+	12.00	0.00	0.00	0.00	
SIN	0.2079116	0.00	0.00	0.00	

exponent digits, a byte representing the signs, one byte for the decimal point position, and the eight mantissa digits' readiness to be sent to the Z80. The interrupt caused by this line has already been described.

The HOLD and POR (power on reset) lines on pins 9 and 11 respectively, are not t.t.l. compatible and Tr_2, Tr_3 in Fig. 2 act as level shifters. Operation of POR occurs under the control of the Z80 whenever a reset is applied, and during the operation internal registers are cleared and various conditions within the 57109 initialised. It is during this operation that the R/W line goes low, but, as previously described, this is prevented from causing interrupts to the Z80. For further information on the MM57109, National Semiconductor produce a data booklet which gives full operational details of each instruction and pin function.

The clocks for the microprocessors are derived from IC_{30} and IC_{46} in the visual display circuitry. To meet the specified swing and rise times required by the Z80, a rise time of 30ns to a level of 4.4V, Tr_1 and the associated circuitry form an active pull-up on the output of the Schottky inverter in IC_{14}. The clock for the 57109 is obtained from pin 12 of IC_{46}. With the link from pin 1 of IC_{46} to pin 12 of IC_{29}, the frequency of this clock is 400kHz, which is the maximum specified in the data sheet. The other position of the link, to pin 14 of IC_{29}, doubles this frequency and, if the 57109 will operate at 800kHz as tested ones have, a worthwhile increase in computing speed can be achieved.

Tri-state buffers IC_9 to IC_{12} are connected to form an eight-line bus transceiver, and buffer the system control lines. These buffers also provide the extra drive required for the heavily loaded data bus. IC_2 is a 3 to 8-line decoder, activated by the control lines IORQ and WR. This allows eight different output devices to be addressed from the codes that the lower eight bits of the address bus holds during output operations. One of these output devices is the latch $IC_{4,5}$. A similar job is done by IC_3 for input devices, and it is enabled by IORQ and RD. The three input lines to these devices are different, which spreads the load on the address bus. The input devices shown are tristate buffers IC_8 and IC_{13}, which buffer data from the 57109 and the keyboard respectively. ☐

To be continued

Literature Received

Digital optical **tachometer** is described by Compact Instruments Ltd in colour leaflet, available from Binary House, Park Road, Barnet, Herts EN5 5SA.

WW 401

Camera tubes. Plumbicon and Newvicon from Mullard are on colourful, but not very informative, wall-chart which can be obtained from Department C1H, Mullard Ltd, Mullard House, Torrington Place, London WC1E 8HD.

WW 402

A range of alphanumeric **printers** is subject of brochure recently sent to us by Syntest, 169 Millham Street, Marlborough, Mass. 01752, U.S.A.

WW 403

General and specific information on **stepping motors** is given by Moore Reed in their 50-page catalogue, which can be had from Moore Reed and Company Ltd, Walworth, Andover, Hants SP10 5AB.

WW 404

Computer **card recorders** for analogue or other information are made by Houston Instrument and described in a brochure from Louis Arnold, Houston Instrument, One Houston Square, Austin, Texas 72753, U.S.A.

WW 405

Brief descriptions of test and measuring **instruments** marketed by Lyons Instruments in product guide, not sent to us but available from L.I. at Ware Road, Hoddesdon, Herts EN11 9DX

WW 406

Audio test instruments produced by Bang and Olufsen shortly described in recent brochure, obtainable from Eastbrook Road, Gloucester GL4 7DE.

WW 407

Health and safety recommendations for Bakelite materials in two publications: TIS B212 for impregnated materials; TIS B211 for industrial laminates. Copies from Bakelite UK Ltd, Sales Office, Tuition House, St. George's Road, Wimbledon, London SW19 4DS.

WW 408

Sound reinforcement systems made by Millbank, described by Royal Institute of British Architects in a product data sheet, is available from Millbank Electronics Group, Uckfield, Sussex TN22 1PS.

WW 409

Company magazine of Rohde and Schwarz for end of 1978 deals with range of r.f. **test gear** and television techniques. Includes description of standard stereo decoder. In English. Rohde und Schwarz, Postfach 80 1469, D-8000 Munich 80, Fed. Rep. Germany.

WW 410

Catalogue of voltage and current **stabilizers** is produced by Techmation Ltd. Units are made by Kepco and use switching techniques, ferroresonance and ordinary feedback methods. Techmation Ltd, 58 Edgware Way, Edgware, Middx.

WW 411