Forth computer

With applications ranging from video games to research and process control this microcomputer combines the powers of Forth, a fast, threaded computer language/operating system, with an eight-bit processor having 16 bit internal architecture.

by Brian Woodroffe

Today a home or personal computer can more than match at lower cost the performance of a typical mid-1970s minicomputer. Then a minicomputer costing tens of thousands of pounds would have a memory of less than 32K words with a cycle time of about a microsecond and a Teletype terminal capable of ten characters per second. Disc-drive memory was rare and expensive and double-precision/floating-point instructions would be executed by software. This microcomputer design costing a few hundred pounds has a 48K read/write memory operating at 666ns, further 8Kbyte rom containing the operating system, a 100-character-per-second terminal and a 200Kbyte disc memory.

	Forth computer	1970s minicomputer
Memory size	56K (48K ram, 8K rom)	64K ram
Memory speed	666ns	960ns
C.p.u. 16-bit add time	4.6μs	1.96μs
Output peripherals	composite video RS232 8 ports	64 ports
Input peripherals	parallel keyboard RS232 8 ports	standard peripherals
Disc storage	200Kbyte/drive	5Mbyte/drive
Access time	333ms	35ms
Cost	£100-500	£10,000-100,000

The cost of developing control software and language application packages is the main reason why low-cost microprocessors have not destroyed the minicomputer industry. It will be a long time before any microprocessor has the software support of the PDP11! Further, when designing a home computer from the i.cs upwards one does not have the support of other computers to develop the software on and one cannot afford to develop the software alone. For these reasons the control program was chosen from those already available. This also applied to the choice of language; I was not willing to start from the bottom with machine code, for one sees too little reward for the effort of keying in programs on a hexadecimal keypad, nor was I prepared to design a 'bootstrap' rom that loaded the operating system in from disc, for I felt it an unnecessary commitment while the rest of the system was unproven.

Language/operating-system choice

The most popular operating system and language in the microcomputer field are CP/M and Basic respectively. Although Basic is readily available, in for example the INS8298 rom for the 8080, I was not prepared to use the language for reasons too many to mention but summed up by Dijkstra[1] who said "It is practically impossible to teach good programming to students that have had prior exposure to Basic." He seems equally impressed by most other languages, including Fortran, PL/1, Cobol, APL and Ada.

My first choice would have been Pascal but for this application Forth appeared to be the best choice. Besides being a language, Forth forms the basis of an operating system and the Forth Interest Group[2] have made FIG Forth a public-domain product. The language is efficient, which is important when using a processor with a limited address range of 64K and it is interactive, avoiding the traps of edit-compile/load-run phases which are a left over of batch-processing systems. FIG Forth is a single-operator, single-task operating system but it has 'hooks' which allow it to be expanded into a multi-operator, multi-task system. It promotes good programming habits in that its programs are structured in blocks and work from top to bottom.

The language has drawbacks – unfamiliar notation, no file structure and poor data structures – but it is readily available and has more advantages than disadvantages. The power and flexibility in Forth allows the operator to expand the language and add any desired feature, and a new version of Forth may be placed on disc by editing and compiled using the resident language to give a completely user-defined version.

Forth

Details of Forth and how it operates are available (ref. 3) and the following is a brief summary. Forth uses 16bit arithmetic and reverse Polish notation, which implies the use of a data stack. Control between executable statements, referred to as a word, is accomplished by the use of indirect-threaded code and a control stack which is separate from the data stack. Features of Forth not found in Basic are virtual memory, compiling, extensibility and vocabularies. These features make better use of the processor resources and a program written in Forth will use less memory and run faster than its Basic equivalent, often by a factor of ten or more in both cases. As Forth compiles the 'English' program into a form readable by the processor (threaded code) the operating speed will always be faster than when using Basic which stores the program as text. Memory space taken up by compiled code is much smaller than would be taken up by its equivalent in English text so larger programs are possible in a limited memory space.

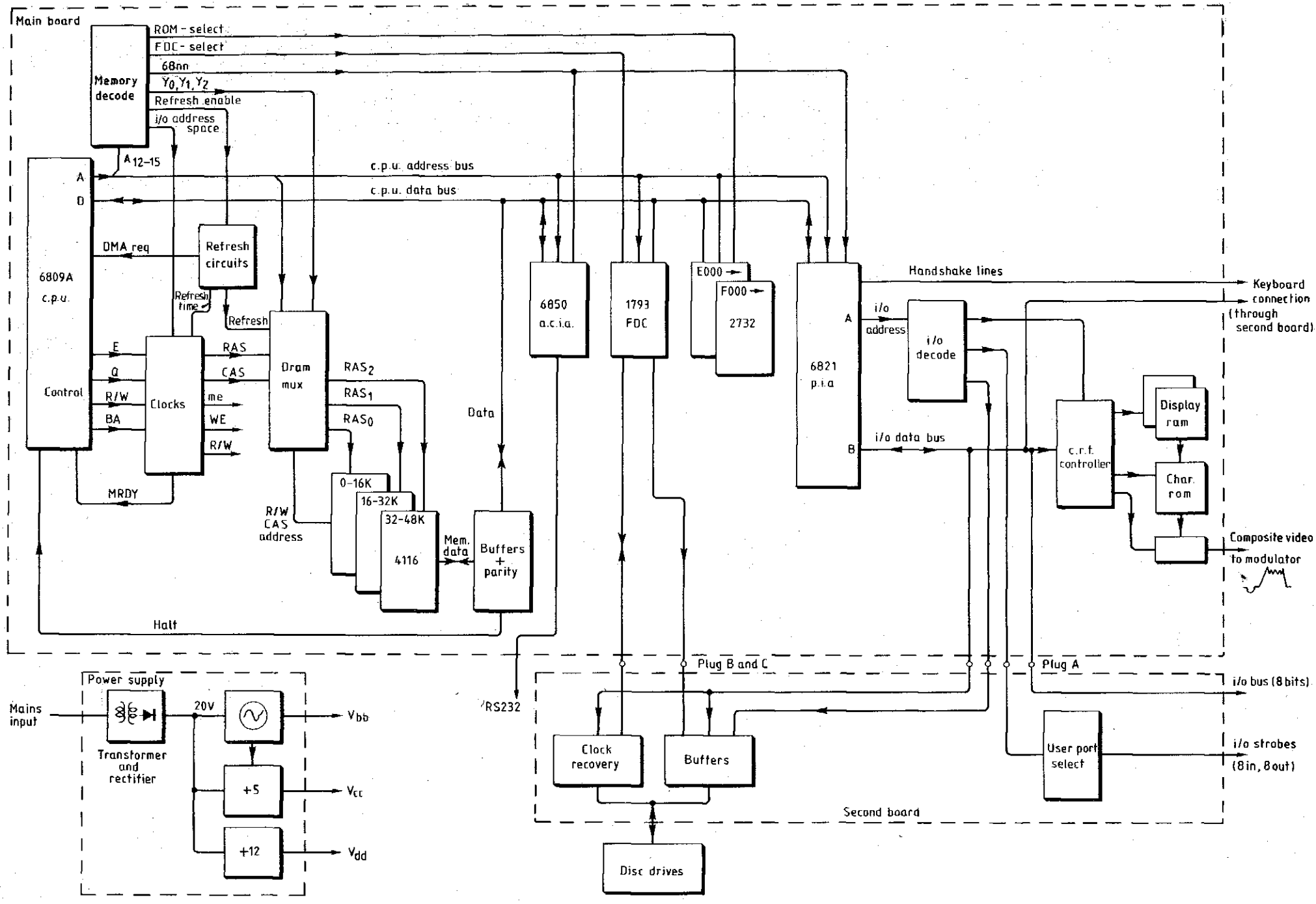
The virtual-memory feature allows the programmer to treat disc storage as processor memory so memory space is not limited by the processor but by the disc. Data is moved to and from the disc by the operating system so the programmer need not be concerned with the problem of mapping the disc memory. Vocabularies allow the programmer to keep different application programs in memory which are physically concurrent but logically separate. Further, there are features found in Forth that are not generally available in Basic such as recursion, extensibility and self-compiling. Recursion allows a portion of the code to use itself more than once at the same time and extensibility is the ability of Forth to define new control words.

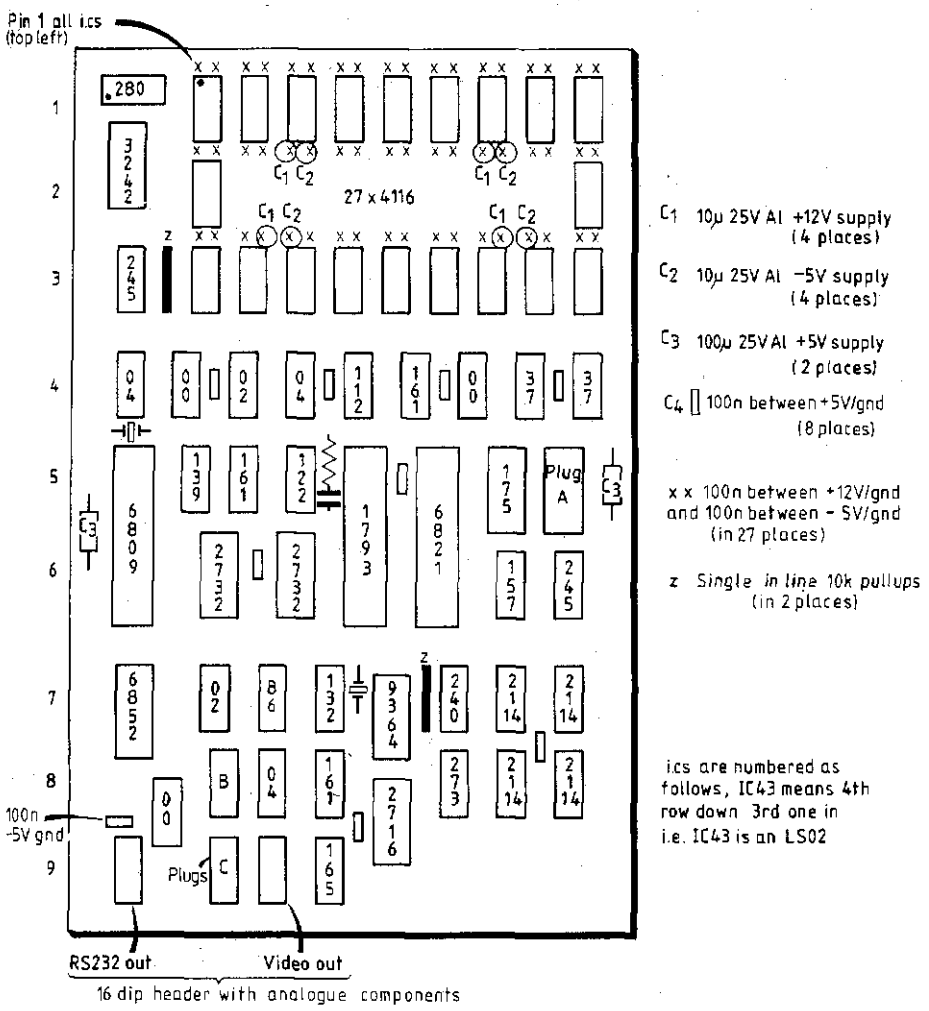
Memory choice

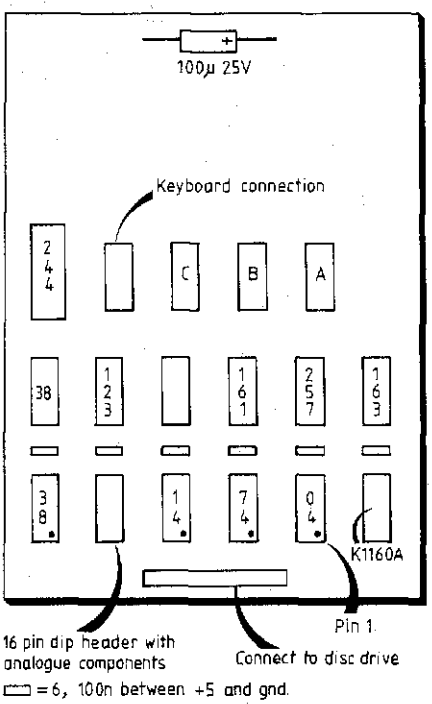
Before selecting a processor for the computer, another design decision has to be made. This concerns memory, in particular what type and how much to use. In time, memory will always become too small and too slow because of the programmer's rising expectations of what the computer should do[4], so 4116 dynamic rams were chosen because they offer the best performance in terms of cost, size and power consumption when compared with static rams such as the 2114. This decision does present some problems in that refresh circuits and three-rail supplies are required. Because dynamic rams are prone to 'soft' errors, parity-checking circuits are included in the design.

Processor choice

The Z80 microprocessor contains dynamic ram refresh circuits and CP/M is written in


Main board
Memory decode
ROM - select
FDC - select
68nn
Y0, Y1, Y2
Refresh enable
i/o address space
A12-15
c.p.u. address bus
c.p.u. data bus
A
D
6809A c.p.u.
DMA req
Refresh circuits
Refresh time
Refresh
Control
E
Q
R/W
BA
Clocks
RAS
CAS
me
WE
R/W
Dram mux
RAS2
RAS1
RAS0
MRDY
0-16K
16-32K
32-48K
4116
R/W CAS address
Mem. data
Buffers + parity
Data
Halt
6850 a.c.i.a.
1793 FDC
E000
F000
2732
6821 p.i.a
A
B
Handshake lines
Keyboard connection (through second board)
i/o address
i/o decode
i/o data bus
c.r.t. controller
Display ram
Char. rom
Composite video to modulator
Power supply
Mains input
Transformer and rectifier
20V
Vbb
+5
Vcc
+12
Vdd
RS232
Plug B and C
Plug A
i/o bus (8 bits)
Clock recovery
Buffers
User port select
i/o strobes (8in, 8out)
Second board
Disc drives





Complete Forth computer system has a 48K memory, floppy disc storage and memory-data parity checking but the system may be used with 16K ram and without disc storage and parity checking to reduce costs. Wire wrapping allows the computer to be built on one relatively small board with a minimum of bus buffering. A further small board holds the disc controller and i/o-port hardware. The system can be set to read most disc formats.

Z80 machine code which surely explains why it is the most widely sold processor, but to use Forth, the most suitable 8bit microprocessor is the 6809. Although most of Forth is written in Forth, the computer must execute some machine code to interpret the most primitive Forth instructions. The 6809 has indexed addressing modes (see "6809 evaluation system" by R. Coates, *Wireless World* July 1980) which suit stack operations and as said earlier, Forth uses two stacks. These examples of stack addition illustrate the merits of the 6809; they represent code of the Forth word '+' for various processors.

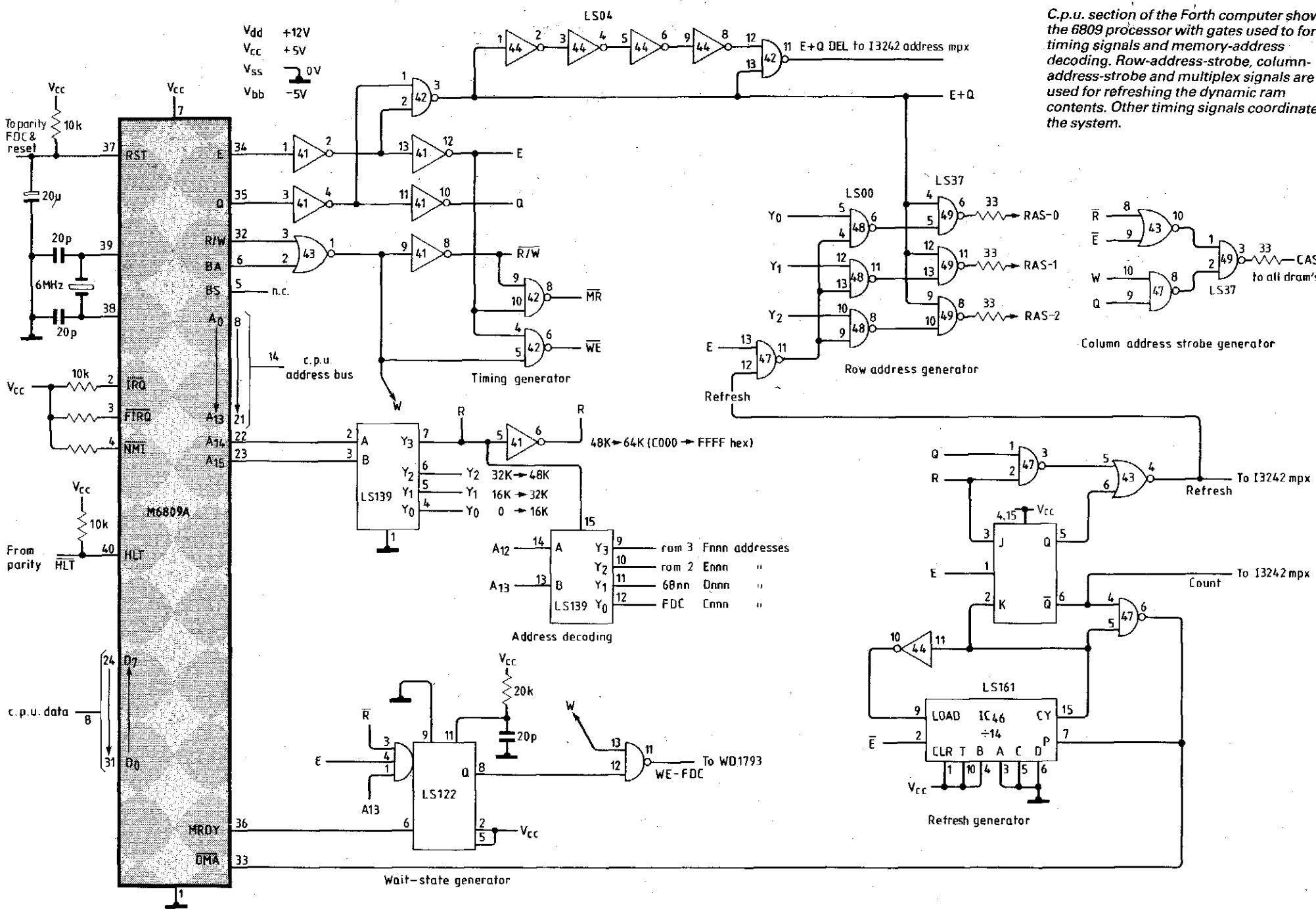
6809		**Z80/8085**		**6800**	
PULU	D	POP	D	PULB	
ADDD	0,U	POP	H	PULA	
STD	0,U	DAD	D	TSX	
		PUSH	H	ADDB	1,X
6502				ADCA	0,X
CLC		**8088**		STB	1,X
LDA	0,X	POP	AX	STA	0,X
ADC	2,X	POP	BX		
STA	2,X	ADD AX,BX			
LDA	1,X	PUSH	AX		
ADC	3,X				
STA	3,X				
INX					
INX					

Secondly, the 6809 instruction set is particularly suited to code the crucial Forth word 'next'. The speed at which 'next' is executed determines the performance of the Forth system since this word controls the indirect-threaded code. 'Next' is called the inner (or address) interpeter to distinguish it from Forth's text interpreter which performs the function of a compiler.

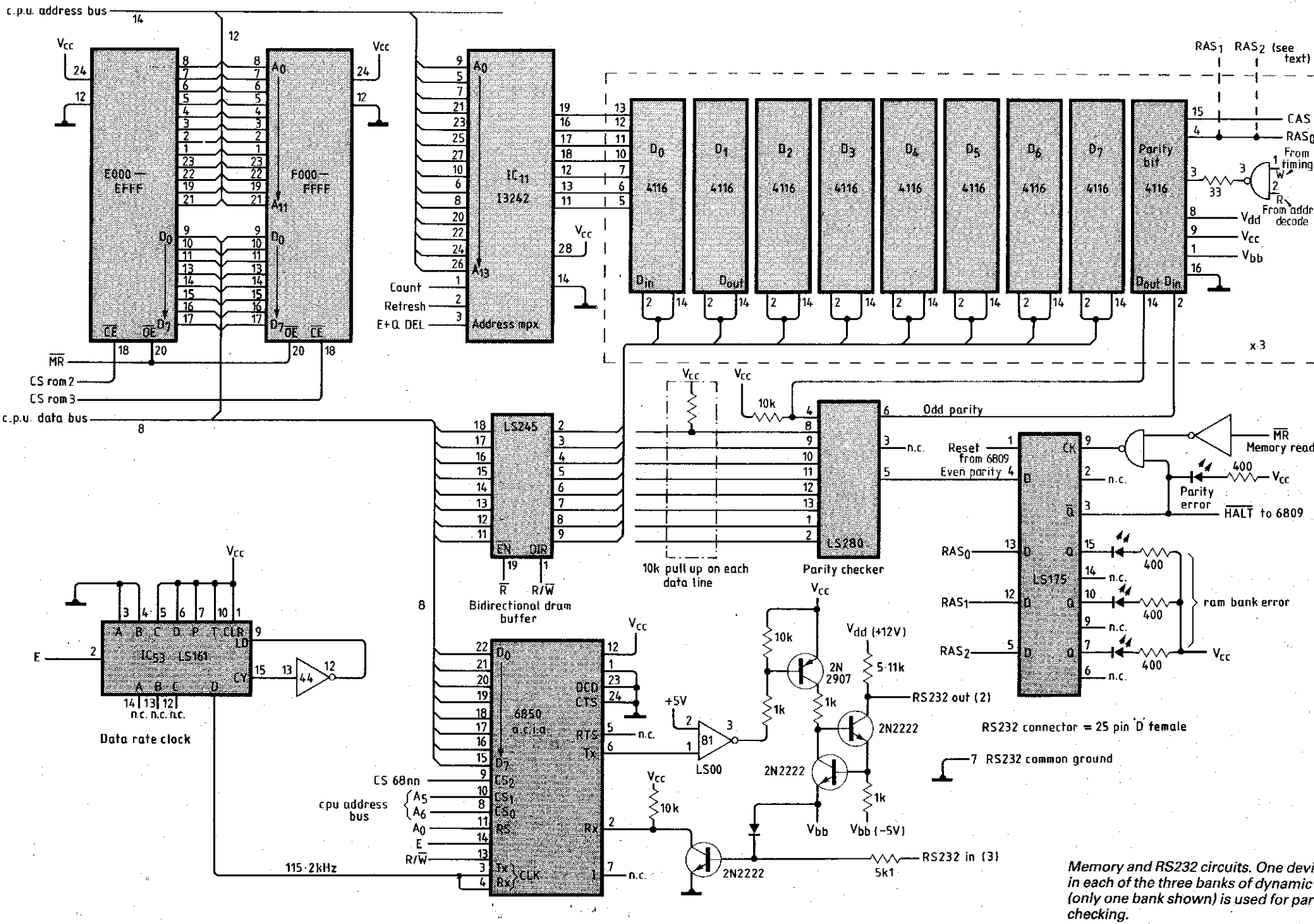
Having worked in Hewlett Packard's production and systems-engineering departments, Brian Woodroffe currently works with the company's South Queensferry research and development group and has recently been involved with designing the microprocessor control section of the HP3724/25/26A baseband analyser. Brian obtained a BA degree in engineering and economics at Downing College, Cambridge in 1970 and an MA in 1975. His computing interests include real-time control, languages and microprocessor graphics but outside electronics, his main interest – rifle shooting, in which he has represented Scotland in full bore – has been curtailed through part-time studies for an M.Sc degree in computer systems engineering at Edinburgh University.

Machine code in the computer emulates Forth operation, the Y register taking on the role of the Forth program counter, and the Forth instruction-fetch cycle is a 'next' machine-code routine. So you can see that the processor choice is dominated by the speed and memory cost of the 'next' operation. Equivalent Forth 'next' operations for some microprocessors are listed below. Because the 6809 'next' operation is so short, it may be copied in line as required resulting in improved performance through avoiding the JMP NEXT instruction required for most processors.

6809	**8088**	**6502**
LDX 0,Y++	JMP NEXT	JMP NEXT
JMP [0,X]	LODS AX	LDY #1
(4.11)	MOV BX,AX	LDY [IP],Y
	MOV DX,BX	STA W+1
Z80/8085	INC DX	DEY
JMP NEXT	JMP WORD	LDY (IP),Y
LDAX B	PTR (BX)	STA W
INX B	**(3.19)**	CLC
MOV L,A	**6800**	LDA IP
LDAX B	JMP NEXT	ADC #2
INX B	LDX IP	STA IP
MOV H,A	INX	BCC $+4
MOV E,M	INX	INC IP+1
INX H	STX IP	JMP W−1
MOV D,M	LDX 0,X	**(1.25)**
XCHG	STX W	
PCHL	LDX 0,X	
(1.37)	JMP 0,X	
	(1)	



C.p.u. section of the Forth computer showing the 6809 processor with gates used to form timing signals and memory-address decoding. Row-address-strobe, column-address-strobe and multiplex signals are used for refreshing the dynamic ram contents. Other timing signals coordinate the system.



Memory and RS232 circuits. One device in each of the three banks of dynamic ram (only one bank shown) is used for parity checking.

Values in parentheses are merit figures obtained by multiplying the number of processor cycles by the processor cycle time then dividing by the memory-access time in the processor cycle. It is interesting to note that the 6809 fares better than the more recently introduced 8088. This is especially so when one realises that the 8088 has a 16bit arithmetic unit whereas the 6809 in common with the other processors noted has an 8bit arithmetic and logic unit (a.l.u.).

Finally, the register set of the 6809 exactly matches that which is required to operate Forth.

6809 register		Forth operation	
S	system stack pointer	RP	return stack pointer
U	user stack pointer	SP	data stack pointer
Y	index register	IP	instruction pointer
X	index register	W	code field pointer
D	accumulator		accumulator

Peripheral devices

Having chosen Forth and the processor to run it on, other design requirements are easily determined. These were selected to maximize the number of peripheral devices that can be easily driven. First a floppy disc was included to provide a modest amount of non-volatile memory with much faster operation than tape recorders. Mini floppy discs were chosen for two reasons, firstly because they are cheap and secondly because the data rate of eight-inch double-density drives is too high for most microprocessors to handle without direct-memory access. Further, eight-inch drives normally require phase-locked loop clock-recovery circuits and also a mains supply.

Three-inch disc drives from Sony were investigated but the data transfer rate is high so that only single-density recording could be used, which would mean wasting half of the data-storage capacity. Both these drives and eight-inch types can be used with the system, provided they run in single density. Processor memory in this system is greater than 40Kbyte so a disc capacity of greater than 400Kbyte is reasonable; one double-sided floppy-disc drive meets this requirement. It is interesting to note that the BBC Micro and Atom computer can only use single-density 5¼in disc drives because of data-rate problems.

Different types of terminal are accommodated. Operating-system words for terminals, KEY, TERMINAL and EMIT, are vectored so that they may be changed on-line between terminal types. At switch-on the system automatically sets vectors for the available terminals. These terminals are either serial RS232 or 8bit parallel for a keyboard such as the RCA VP601/611 and integral video compatible with 625-line tv, displaying 1,024 characters in 16 lines (the EF96364B controller may be used for 525 lines). The video section has its own memory, leaving 48K of memory free for other programs. Bit-mapped graphics video is best handled through a secondary processor connected to the user ports. A number of definable i/o ports are spare to allow for expansion of the system. Certain design features were included to reduce cost. By keeping the computer system down to one board, bus drivers necessary to overcome capacitance encountered in larger systems are avoided. Another reason for avoiding these buffers is that they cause delays which eat into the access time available from communicating devices. The switch-mode power supply used means that a readily obtainable transformer with a single secondary winding may be used to provide all three rails (+12, +5 and −5V).

Next article describes computer circuitry.

References

1. E. W. Dijkstra, How do we tell truths that might hurt? ACM Sigplan notices, vol. 17 no 5, May 1982, p.14.
2. Forth Interest Group (FIG), PO Box 1105, San Carlos, CA94070, USA.
3. L. Brodie, Starting Forth. Prentice Hall, 1981.

B. Allan, Forth: a threaded interpretive language, *Wireless World* Nov. 1982 p.74.

G. Feierbach, Forth – the language of machine independance, Computer Design, June 1981 pp.117-121.

Byte, August 1980 for various Forth articles.

4. W. Wulf, Compilers and Computer Architecture, *Computer*, July 1981 p.41.

continued from page 50

Z8 Basic listing for eight-channel a to d converter (@ ≡ byte, % ≡ hexadecimal)

```
  1 PRINT "SILICONIX LD120/121A TO Z8
    INTERFACE"
  2 PRINT "HIT ANY KEY TO RUN";: GO
    @%61,%07:R=USR(%54)
  5 A =  )      : B =  )
 10 C =  )      : D =  )
 15 E =  )      : F =  )
 20 G =  ) see* : H =  ) see*
 25 I =  )      : J =  )
 30 K =  )      : L =  )
 35 M =  )      : N =  )
 40 O =  )      : P =  )
 45 N =1 : Z=%80
 50 @2=Z
 55 @3=%A0:03=00
 60 @2=%00
 70 W =W+1:IFW<15 THEN 70
 80 W =0:GO@%1400 (m.c. routine for
    collection of data)
 85 IF N=1 THEN X=A: Y=B: GO TO 130
 90 IF N=2 THEN X=C: Y=D: GO TO 130
 95 IF N=3 THEN X=E: Y=F: GO TO 130
100 IF N=4 THEN X=G: Y=H: GO TO 130
105 IF N=5 THEN X=I:  Y=J: GO TO 130
110 IF N=6 THEN X=K: Y=L: GO TO 130
115 IF N=7 THEN X=M: Y=N: GO TO 130
120 IF N=8 THEN X=O: Y=P: GO TO 130
125 GO TO 45
130 Z=Z+%10
135 V=@%26+@%25×10+@%24×100
    +@%23×1000+@%22×10000
140 IF V>X THEN PRINT "CHANNEL";
    N;"OVERRANGE";:GO TO 1000
145 IF V<Y THEN PRINT "CHANNEL";
    N;"UNDERRANGE";:GO TO 1000
150 N=N+1: GO TO 50
1000 GO@%61,%07:PRINT @%22;".";
    @%24;@%25;@%26;
1010 PRINT"MAX";X;"MIN";Y
1020 N=N+1 : GOTO50
```

*Limits entered here for process monitoring.

Machine code routine

Line	Assembler	Hex
200	LD % F7, # % 41	E6 F7 41
210	LD % F6, # % 0F	E6 F7 0F
220	AND 3, # % 04	56 03 04
230	JRZ, * 220	6B FB
240	LD % 22, 2	E4 02 22
250	AND 3 # % 04	56 03 04
260	JR N2, * 250	EB FB
270	CLR % 21	B0 21
280	AND 3 # % 08	56 03 08
290	JRZ, * 270	6B FB
300	PUSH 2	70 02
310	INC % 21	20 21
320	CP % 21, # 4	A6 21 04
330	JRZ 2, * 350	6B 07
340	AND 3, # 08	56 03 08
350	JR N2 * 320	EB FB
360	JR * 270	8B EB
370	POP % 26	50 26
380	POP % 25	50 25
390	POP % 24	50 24
400	POP % 23	50 23
410	RET	AF

Using the Z8 microprocessor

The potential of the Zilog Z8 microcomputer for low-volume or one-off applications has not been well publicised. Introduced in 1981, the chip includes all the blocks that you would find in a standard computer system including c.p.u., ram, i/o and a rom containing Tiny Basic. In addition there are two timers, one for serial i/o, the other available to the user. External memory of 60K can be added for program storage together with 60K of data storage. The evaluation kit used in this was supplied by Ambit International and posesses a comprehensive monitor, enabling machine-code routines to be written in Assembler and stored in eprom. The Tiny Basic name does not reveal the full potential of itself, and with an 8MHz clock driving the Z8, it executes most of the instructions in 1.5 to 2.5 μs.