SC84 Micro-computer

Designed for engineers and enthusiasts, the SC84 microcomputer uses a 6MHz Z80 processor and has 64K-bytes of ram — but its main feature is that it can be used with a disc operating system and much readily available applications software.

by J. H. Adams

I designed my first computer, the Scientific Computer published by *Wireless World* in 1979, to gain experience with microprocessors. This small system had novel features for its time including a hardware number cruncher and up to 5K-byte of ram! Looking back, the Scientific Computer appears embarrassingly primitive but, judging from correspondence, it served its purpose of giving readers the best possible introduction to microprocessors — hands-on experience.

This new design has a similar objective but it also permits the use of much readily-available software including word processors, language interpreters and compilers. Retained features are the Z80 microprocessor, the resident machine-code operating system extended to provide extra commands, and general accessibility needed for engineering applications. New features are the 64K-byte of user memory, a high-resolution c.r.t. controller and a flexible i/o section including interfacing for 3.5, 5.25 or 8in single or double-sided, single or double-density disc drives. Up to 32 lines of 96 characters or 192 by 192 picture elements may be displayed and graphics and characters may be mixed.

My disc operating system, SciDOS (see note at end of article), is compatible with most software written for the standard 8-bit operating system CP/M. I have also designed software to make use of special features of the computer, in particular the v.d.u. These programs include utilities, disc editors and an extended Basic interpreter. Much of this software was developed in conjunction with the Scientific Computer whose disc interface came later, so users of the original computer will find that their software runs on the SC84 with little or no modification.

SC84 reflects the shift towards microcomputers with most of their programming on disc rather than in read-only memory. The only rom in this design is an 8K-byte eprom which on switch-on or reset copies the resident operating system into random-access memory (ram) and is then switched out, leaving the system entirely dependent on ram. There are two advantages in this approach. Firstly, having everything in ram means that every aspect of the computer is open to experimentation. Secondly, while a system with, say, Basic in rom will be ready to program in Basic as soon as it is switched on, that rom is an encumbrance when you want to use anything else but Basic. The classic argument against disc-based systems is that a rom-based system is ready for use as soon as you switch it on, whereas initiating a disc-based system can take as much as 45 seconds. SC84 initiates in just under one second and leaves virtually all of the system ram available for whatever you want — Basic, Pascal, machine-code assembly, word-processing etc. A major feature of SC84 is that a disc operating system, SciDOS, has been written especially for it. As well as implementing those commands and functions necessary for CP/M compatibility, this software provides some extra commands and functions which make the system of use to those who see a computer as more than a black box. SciDOS has been kept small by efficient programming; when it is loaded and running, up to 58K-bytes of memory are free for user programs.

SC84 is built on 100 by 160mm Eurocard p.c.bs interconnected through a 64-way bus system. The basic configuration consists of a processor card, a character v.d.u. card and an i/o card. Frames for housing Eurocards are available in various sizes and materials, the interconnecting bus or 'backplane' being either a p.c.b. — again readily available — or a series of card sockets linked using wire-wrapping techniques. Prototypes have been constructed using both methods and while a p.c.b. saves time, wire-wrapping a series of sockets together is recommended as being cheaper and giving a little more flexibility should you not want all of the connectors wired strictly in parallel, as would be the case in a 'daisy-chained' interrupt system. Bus signals are shown in Table 1. Pin designations refer to a standard DIN 41612 64/96 connector i.e. the type with spacing for three rows of pins but with the middle row missing. Power is provided through the outer two pairs of pins at each end of the connector which suits p.c.b. backplanes available from Vero and other manufacturers. All signals are buffered in and out of the processor board using low-power Schottky t.t.l. i.cs.

Processor/memory board

On this p.c.b. is the Z80, 64K-bytes of ram, system rom and a buffered interface to the rest of the computer. The decision to integrate memory with the microprocessor was taken as the size

SC84

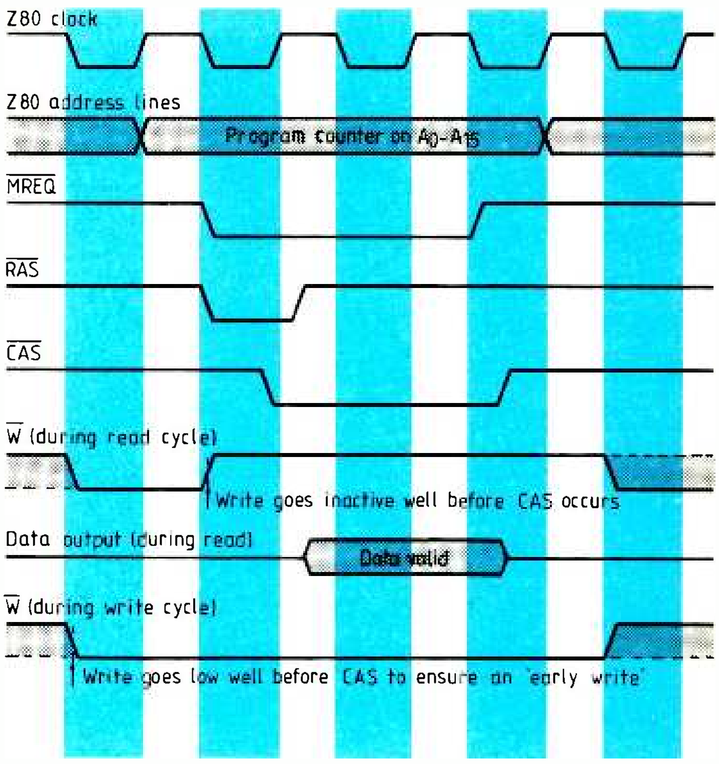
Processor
4/6MHz Z80 processor.
Maximum 64K-byte ram.
58K-byte ram available using SciDOS.

Display
Up to 32 lines of 96 characters fully programmable. Scrolling window determined by software.
Graphics mode 0 gives 192 by 96 pixels, mode 1 gives 192 by 192 resolution. Characters and graphics may be displayed simultaneously.

Input/output
Up to four single or double-sided 8, 5.25, 3.5 or 3in disc drives may be used, either single or double density.
RS232 serial i/o data rates range from 1 to 38400baud with separate transmit/receive clocks. Synchronous serial i/o format is 5 to 8-bit auto-search and sync. or asynchronous 5 to 8-bit with 1, 1.5 or 2 stop bits. RTS and CTR signals control serial data flow.
Eight-bit parallel data input is buffered by schmitt i.cs. Eight-bit parallel output drives five t.t.l. loads.
Three mos i/o lines operate event counters, pulse timers and Z80 interrupts. Four mos timer lines are available for timing and sound generation.

John Adams is currently working on a high-resolution colour-graphics processor using the 7220, and an eprom programmer interfacing to SC84 but with its own processor.

Timing for Z80 memory read and write cycles. 'Early write' cycles are common in larger systems but are not found in most eight-bit processors.



Z80 clock
Z80 address lines
Program counter on A0-A15
MREQ
RAS
CAS
W (during read cycle)
Write goes inactive well before CAS occurs
Data output (during read)
Data valid
W (during write cycle)
Write goes low well before CAS to ensure an 'early write'

First of SC84's three main sections — the processor — with 64K-byte ram, operating system eprom and logic for dynamic-ram, bus-driver and reset control(far right). On resetting, part of the rom content is loaded into high ram and the rom is then switched out, leaving up to 64K-bytes for user programs.

of the system memory is largely determined by the processor. Also, without the memory the processor board would be rather bare and an extra Eurocard would be needed. Integrating the two on one board doesn't preclude the use of extra memory on other boards — as indeed happens with the v.d.u. memory. Timing diagrams shown will be referred to throughout this explanation of the processor board.

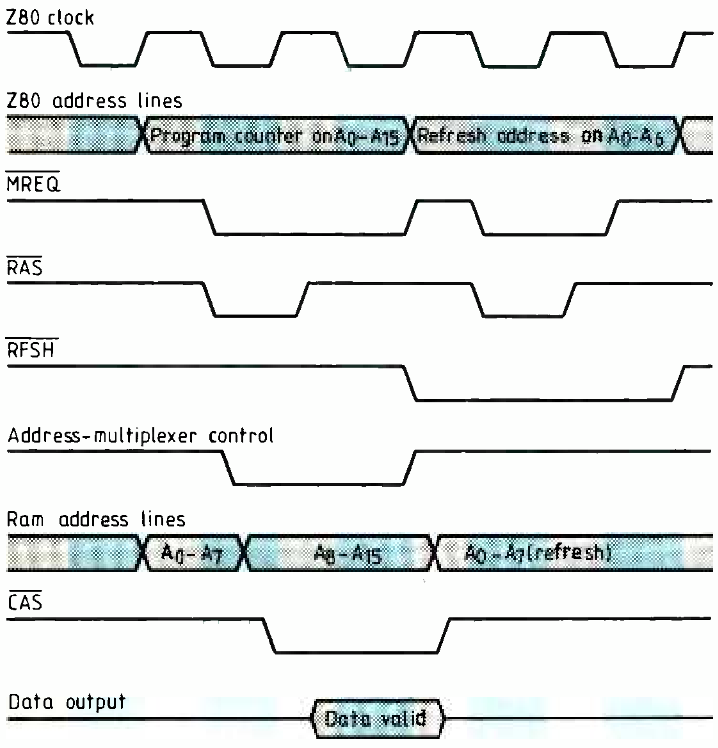
There are three types of memory cycle that the Z80 can execute. Fetching of an instruction or 'op-code' from memory is illustrated in the first diagram. The second is a composite diagram illustrating the writing or reading of data to or from memory. The difference between fetching an op-code and fetching data from memory is that the op-code fetch is shortened and followed by a special memory cycle intended, and used in this case, to refresh dynamic memory. Three relevant Z80 control signals in accessing memory are $\overline{\text{MREQ}}$ indicating that the current cycle is a memory cycle, $\overline{\text{RFSH}}$ indicating that memory refresh can now take place and $\overline{\text{RD}}$ which indicates that the current cycle will involve data passing into the Z80. There is also a signal called $\overline{\text{M1}}$ which becomes active during op-code fetches and interrupt acknowledge periods, and a $\overline{\text{WR}}$ signal which indicates that data is to pass from the Z80 to the system. Neither signal is used in memory access although $\overline{\text{M1}}$ takes part in controlling the buffering of the data bus.

Z80 control signals and virtually all others in this design are active low, i.e. they assert their function by going to the negative or '0' state. The normal description of a gate function is based upon positive logic so that, for instance, a 74LS02 is deemed to contain four two-input NOR gates. In this circuit diagram and following ones, gates are shown in their logical context. This can be seen in the 74LS02 which arbitrates the data-bus buffer direction (IC_{118}). Here, three of the four gates have been drawn in their inverse form, i.e. as AND gates with active low inputs rather than as OR gates with an active low output. Having differently shaped gates in the same i.c. takes some getting used to but it helps understanding of the logic. For example, in the case of IC_{118}, direction line $\overline{\text{IN}}$ becomes active when ($\overline{\text{M1}}$ *and* $\overline{\text{IORQ}}$) *or* $\overline{\text{RD}}$ *and not* ($\overline{\text{MREQ}}$ *and not* $\overline{\text{MEMDIS}}$) are active. Translated into English, this means that the data buffer faces towards the Z80 during interrupt acknowledges and during any read other than one on the processor board memory.

Control gating for the bus buffer is fairly complex as the buffer has to respond to various conditions, summarized in Table 2. One reason for such a tight definition of the bus operation arises from the use of mode 2 interrupts. This is the Z80's most complex mode of interrupt organization where, in response to the Z80 acknowledging an interrupt by taking control lines $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ low simultaneously, the interrupting device supplies eight bits of an address. This address combined with the contents of the Z80 'I' register forms a 16-bit pointer to a table of addresses of interrupt-service routines. Each device capable of interrupting is supplied with one or a range of such 'interrupt vectors' during the computer's initialization so upon an interruption the Z80 is able to pick out and make a call to specific routines for each interrupting vector. The strength of the system though is that by changing the interrupt vector or a particular entry in the table of addresses, different service routines can be provided for the same interrupt line. This is particularly important, for example, with the disc controller used which uses one interrupt line to signal both a request for data from the system during disc writing and the offer of a data byte during disc reading. This 'interrupt acknowledge' sequence is condition seven in Table 2. On receipt of an interrupt the Z80 disables its interrupt sequence but pauses before acknowledging to allow what may actually be several interrupting devices to decide which has the highest priority. This is done by the daisy-chain technique mentioned earlier. In this way lower priority devices that might need service are prevented from interrupting more important tasks. Note that condition four must be implemented as during an interrupt service, all devices capable of interrupting will want to watch what is being fetched from memory so that they can spot the return-from-interrupt op-code being fetched and automatically re-initialize themselves.

The dynamic memory control is quite novel. For this reason, and for the bad publicity that dynamic memory sometimes gets, it is worth detailing a few

Timing for an op-code fetch. The Z80 microprocessor has a special register for use with dynamic rams which provides a refresh address coinciding with a refresh control signal.



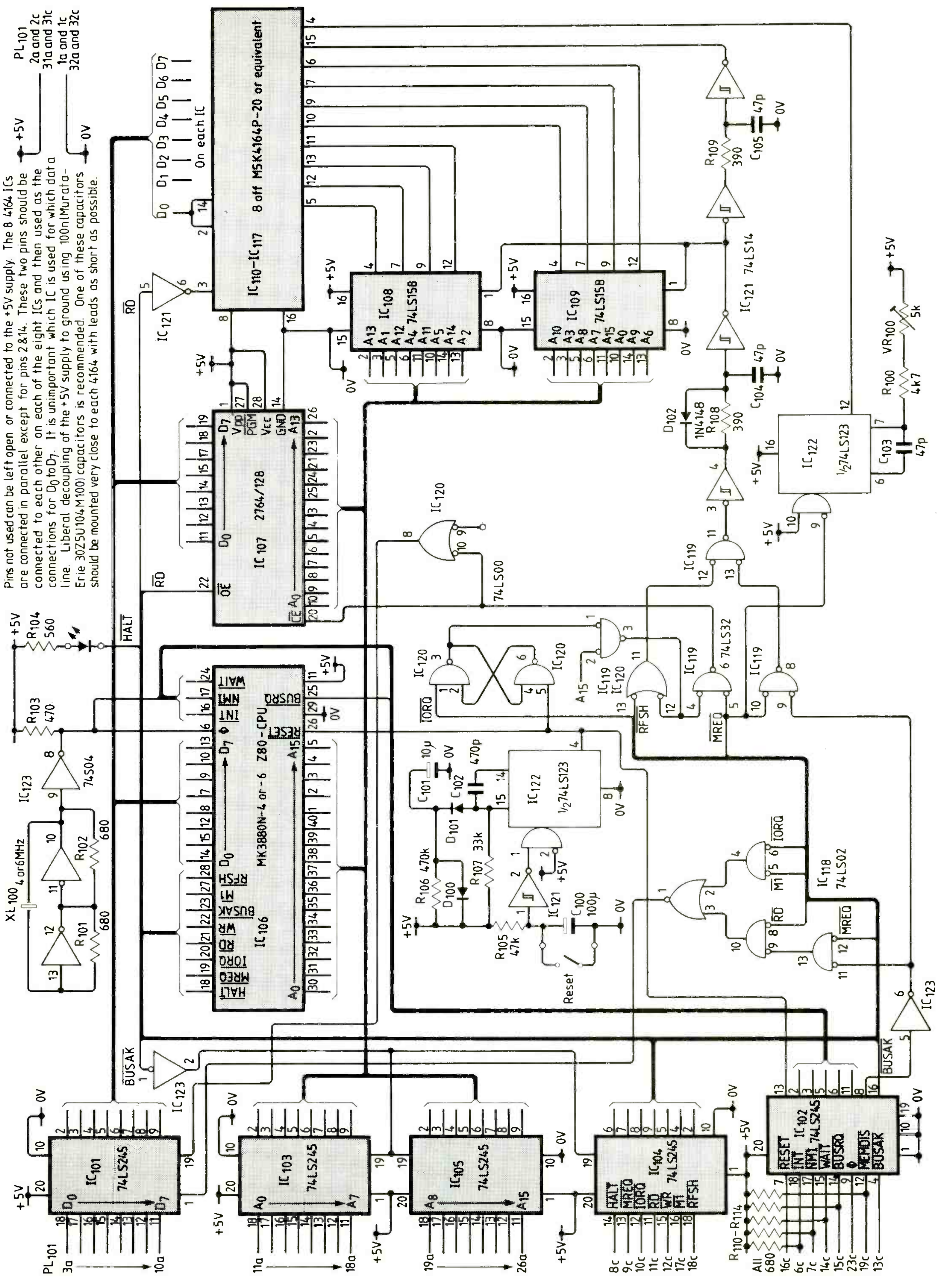

Pins not used can be left open or connected to the +5V supply. The 8 4164 ICs are connected in parallel except for pins 2&14. These two pins should be connected to each other on each of the eight ICs and then used as the connections for D_0 to D_7. It is unimportant which IC is used for which data line. Liberal decoupling of the +5V supply to ground using 100n (Murata-Erie 30Z5U104M100) capacitors is recommended. One of these capacitors should be mounted very close to each 4164 with leads as short as possible.

Table 1. Bus connections

Row A	Pin	Row C	Function
GND	1	GND	
+5V	2	+5	
D0	3		
D1	4		
D2	5		
D3	6	$\overline{\text{INT}}$	Z80 maskable interrupt line*
D4	7	$\overline{\text{NMI}}$	Z80 non-maskable interrupt line*
D5	8	$\overline{\text{HALT}}$	Z80 has executed a HALT instruction
D6	9	$\overline{\text{MREQ}}$	Z80 is performing a memory operation
D7	10	$\overline{\text{IORQ}}$	Z80 is performing an input/output operation†
A0	11	$\overline{\text{RD}}$	Z80 is requesting data from the bus
A1	12	$\overline{\text{WR}}$	Z80 is writing data to the bus
A2	13	$\overline{\text{BUSAK}}$	Z80 has relinquished control of the system
A3	14	$\overline{\text{WAIT}}$	Z80 is being asked to extend the current instruction*
A4	15	$\overline{\text{BUSRQ}}$	Z80 is being asked to relinquish system control*
A5	16	$\overline{\text{RESET}}$	20μs pulse generated when RESET is operated
A6	17	$\overline{\text{M1}}$	Z80 is fetching an op-code†
A7	18	$\overline{\text{RFSH}}$	Z80 is performing a memory refresh cycle
A8	19	MEMDIS	Non-system memory is to be used for this operation*
A9	20	VDUSEL	Character v.d.u. is mapped into memory
A10	21		
A11	22		
A12	23	CLK	System clock (from processor board)
A13	24		
A14	25		
A15	26		
	27		
	28		
	29		
+12	30	−12	
+5	31	+5	
GND	32	GND	

Note: address and data (A and D) lines are active high
* These lines have a pull-up resistor on the processor board and should be driven by open-collector drivers.
† When $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are both active the processor is inviting a peripheral device to supply part of an interrupt vector — interrupt acknowledge.

Table 2. Data-bus driver logic

Condition	Bus-buffer drives
1 I/O write	towards bus
2 I/O read	towards Z80
3 System-memory write	towards bus
4 System-memory read	towards bus
5 External-memory write	towards bus
6 External-memory read	towards Z80
7 Interrupt acknowledge	towards Z80

features of its operation before describing the external circuits. Modern dynamic memories use a multiplexed addressing technique where the address is split in two, in this case eight-bit parts. This reduces the number of address pins on the i.c. from 16 to 10, eight address pins and two 'strobes' to latch the address bytes into the memory. A prime consideration here is that this reduces the package size and hence the cost, but it also allows the memory access to be broken into two stages with consequent benefits.

Consider a 64K-bit dynamic memory to be a matrix of memory stores 512 by 128, each row of 128 being connected to a single bus line. Each store is a minute capacitance connected by gates to these lines. When the first part of the address known as the row address is latched into the memory, the highest bit is stored and the other seven are decoded to decide which of the 128 cells in each of the rows should be connected to the bus line for that row. Thus access starts well before the full address is in. The bus line is, naturally, physically much bigger than the individual cell which has now been connected to it by activation of the row-address strobe ($\overline{\text{RAS}}$) and so the potential stored in the cell is all but lost on the bus. At one point on the bus is a sense amplifier connected back onto the bus lines with positive feedback. While $\overline{\text{RAS}}$ is inactive the sense amplifier is held in balance so now, even though it has been all but lost, the potential delivered by the cell is enough to tilt the amplifier one way or the other. Having positive feedback, the amplifier pushes the potential on the line heavily in the direction of the input potential, putting the line — and the cell — back to the level in the cell prior to the access. There are two implications here. Firstly that there is a minimum length for the active $\overline{\text{RAS}}$ pulse in that if it goes off before the bus line is recharged by the sense amplifier it will disconnect the read cells from their bus lines before they have had a chance to recharge. Secondly there is a minimum inactive time for $\overline{\text{RAS}}$ as the sense amplifier is brought back into a state of balance. These conditions are paramount and to meet them $\overline{\text{RAS}}$ is driven by a monostable i.c. triggered by a signal directly from the Z80 rather than one which has been combined with others and might therefore be subject to glitches caused by timing problems between the various constituent signals. The monostable i.c. sets the minimum active $\overline{\text{RAS}}$ period and thus by definition the minimum inactive.

Once the leading edge of the $\overline{\text{RAS}}$ pulse has latched the first address byte into the memory i.cs., the address may be changed to that of the bus row to be fed to the output of the memory. The eight bits are gated in by the starting edge of the column-address strobe ($\overline{\text{CAS}}$) signal and combined with the stored bit from the previous addressing strobe to operate a 1-out-of-512 data multiplexer which selects and latches the signal from one of the 512 bus lines within the memory. Once this is done, $\overline{\text{RAS}}$ may go inactive; indeed it is a good thing if it does as then the sense amplifiers may return to a balanced state as soon as possible, ready for the next access. $\overline{\text{CAS}}$ also controls the state of the memory-output driver. While it is active the output is enabled and the selected data bit held there. $\overline{\text{CAS}}$ is a far less sensitive signal as far as integrity of the memory is concerned, the main consideration being that it becomes active as soon as possible in the access and stays on until the data is definitely available and the Z80 has it.

This has been a description of a memory read cycle. A write cycle is similar in that the $\overline{\text{RAS}}$ signal connects cells to buses and releases sense amplifiers and then the $\overline{\text{CAS}}$ signal operates the latching multiplexer and activates the output driver. What is different is that the signal on the data-input pin is routed through the multiplexer to the cell. During a conventional write cycle the data output pin will follow the output of the cell while $\overline{\text{CAS}}$ is active. This might seem to preclude the use of dynamic memory in circuits usually associated with static memory, where the same line is used for data input and output, but it is possible to prevent the output of a dynamic memory from coming on when $\overline{\text{CAS}}$ goes active by arranging for the write signal to go active before $\overline{\text{CAS}}$ does. These 'early-write' cycles are common in large systems but are not found in most eight-bit microprocessors such as the Z80 where the $\overline{\text{WR}}$ signal goes active well into the memory cycle and is too late to be of use. One answer is to use an eight-bit three-state buffer between the ram outputs and the data bus, but my solution is to use the inverse of $\overline{\text{RD}}$ as a write strobe to the memory. Whenever $\overline{\text{RD}}$ goes high, i.e. at the end of reading memory or i/o, the write strobe goes low and so the dynamic memory is primed for an 'early write'. As $\overline{\text{RAS}}$ will be over long before $\overline{\text{RD}}$ goes high, there is no chance of a memory-read cycle being transformed into a memory write cycle. Should it be a read cycle then the write strobe is removed from the memory by $\overline{\text{RD}}$ going active low at the beginning of the cycle, well before $\overline{\text{CAS}}$ is applied. Except for a slight increase in current consumption during the write-line strobe, the effect is unnoticable.

The sequence of pulses for the dynamic memory is generated by a series of Schmitt buffers (IC_{121}). $\overline{\text{RAS}}$ is generated by the leading edge of memory-request signal $\overline{\text{MREQ}}$ which triggers monostable IC_{122}. A potentiometer sets the $\overline{\text{RAS}}$ pulse length. For the devices specified in the diagram $\overline{\text{RAS}}$ should last for at least 200ns and should have a minimum off period of 120ns. In practice this adjustment is not too critical with a system running at 4MHz as a complete $\overline{\text{RAS}}$ cycle lasts at least two clock cycles which corresponds to 500ns. Set the potentiometer to give the off-period required for the dynamic memories used or, if measurement is not possible, to its mid-position for a 4MHz microprocessor or at or near its minimum for a 6MHz version.

Continued on page 51

SC84 microcomputer

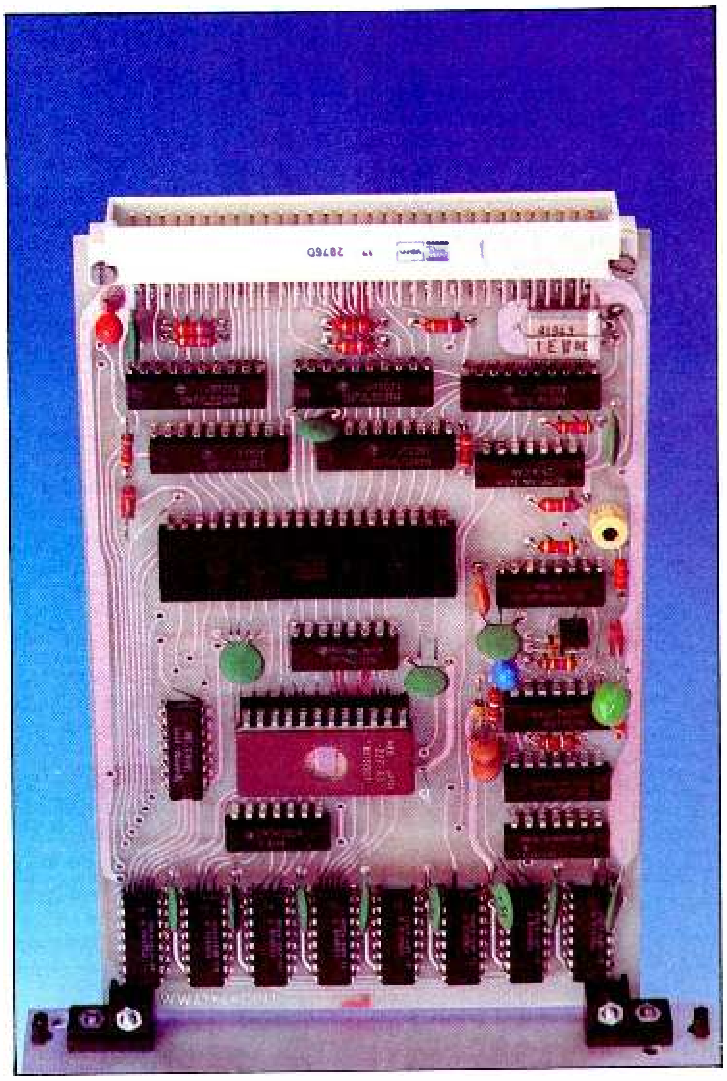
continued from page 40

The signal at the start of the Schmitt buffers is (MREQ *and not* MEMDIS) *and not* (RFSH *or* ROMEN), i.e. unless MEMDIS, RFSH or ROMEN is active it is MREQ delayed by a couple of gates. The starting (falling) edge of MREQ becomes a rising edge at the output of the first Schmitt gate. This rising edge is slowed down by the RC combination between the first two buffers so that the falling edge at the second output is somewhat later than that of MREQ. This signal is used to switch addresses being supplied to ram through multiplexers $IC_{108,9}$. Further delay is applied to produce CAS. To provide a quick end to the memory access so that the multiplexers are definitely reset to the row-address position ready for a refresh cycle, a diode shunts the first time delay which would follow MREQ's trailing (rising) edge. Refresh occurs when MREQ cycles while the RFSH signal is active. During this period, the Z80 puts out a seven-bit value from a special internal register which increments after every refresh. MREQ triggers the monostable i.c. to set off a RAS cycle but the RFSH signal inhibits production of address-multiplex or CAS signals. From the previous discussion of dynamic memory operation, one can see that the sole result of the RFSH cycle, or any memory cycle during which MEMDIS or ROMEN is active, is to 'refresh' 1/128th of the memory.

A simple inverter ring produces the system clock. The crystal frequency will depend on the microprocessor used, i.e., 4MHz for the Z80A or 6MHz for the Z80B. A Z80A should work with a popular 4.1943MHz crystal. These are relatively cheap and although the MK3880-4 or Z80A-CPU is only specified up to 4MHz, it is unlikely that they will not work at this slightly higher frequency. Each i/o section has its own crystal and oscillator, so the crystal used on the processor board will not affect i/o data rates, etc., with the proviso that the processor clock must not fall below 3.6MHz if 3.5 or 8in double-density disc drives are used. When the RESET pin on the Z80 is active everything, including memory refresh, stops. For this reason the reset monostable i.c. generates a short pulse and no matter how long the reset button is held in, refresh is only briefly interrupted so that no memory corruption occurs. As well as providing a reset signal for the Z80 and all peripheral circuits, the pulse is used to set a bistable i.c. formed from two gates of IC_{120}. The output of this bistable device gates with A_{15} to produce ROMEN, a signal which follows A_{15} while the bistable i.c. is set and enables rom whenever A_{15} is low, i.e. during access at any address from zero to $7FFF_{16}$. On receiving a reset signal, the Z80 starts to fetch and execute instructions from address location zero. This means that following reset, the Z80 executes instructions from eprom. At the base of the eprom it finds instructions to copy a part of the rom at the top 8K-bytes of system memory followed by a jump that area. This copied software is the machine-code operating system whose first instruction is an i/o read which, due to the IORQ line going active during its execution, resets the flip-flop and forces ROMEN to the inactive high state, disabling the eprom and freeing the entire 64K-byte ram. The timing circuit on the reset monostable i.c. arranges for a much longer time constant to be applied during power-up, providing a reset pulse long enough to allow start up of all of the system clocks. Eprom IC_{107} is shown as a 2764 or 27128 but the board may be modified to take 27256 or 27512 devices.

Prototypes of the computer have been made in wire-wrapped and soldered wiring forms, the most recent version using Vero 03-2989L boards and wiring pen type 79-1732G. Suggested wiring layouts and component placement diagrams for such boards can be obtained by sending a large s.a.e. to *Wireless World's* editorial offices. Construction using p.c.bs is a much easier matter and will be the assumed method. When using these boards the i.cs should be soldered directly onto the boards with the exception of eproms which should be fitted in good quality sockets. Sockets may be used if required, but only good quality ones. Dynamic rams are best soldered in. The natural fear is that of removing i.cs should faults occur. This is quite easy though, either using a desoldering tool or by snipping all of the pins off the i.c. body and removing them one by one. A system as complicated as this should not be repaired using the swap-and-see technique so for those who do not have the test facilities to trace a fault, a repair service will be available — for systems built on p.c.bs! Standard pitches have been used for discrete components, details of which appear with each circuit diagram.

John Adams' next article describes SC84's input/output board. SciDOS with utility software, extended Basic with graphics facilities and Basic with enhanced file manipulation, i/o control, numeric/constant string handling and 12-digit precision are available for £36, £22.50 and £31.50 respectively. These prices include vat and postage and become £24, £15 and £21 respectively for non-commercial users. Further discounts are available for those buying more than one software package at once. Write enclosing s.a.e. to J.H. Adams, 5 The Close, Radlett, Herts.



A set of three Eurocard-format plated-through hole boards for SC84 is available from Combe Martin Electronics, King Street, Combe Martin, Devon EX34 0AD. Price is £39 for the set including v.a.t. and inland or overseas postage. John Adams is considering producing a kit of parts for these boards and John Hodson - secretary of the Scientific Computer User Group - is organizing the SC84 user group. For further information send an s.a.e. to John Adams for kit details or John Hodson, 189 Trent Valley Road, Oakhill, Stoke-on-Trent ST4 5LE, for user group details.