Upgrading from 68000 to 68020/68881

Bob Coates has designed a Kaycomp-compatible platform board holding a 68020 processor and 68881 maths co-processor; it should plug into any single-master 68000 system that can keep up with the 68020's shorter bus cycle.

BOB COATES

My design objectives were to produce a platform board for the 68020 microprocessor and 68881 maths co-processor, together with interface logic, which could plug into the 68000 processor socket on the Kaycomp board, Fig.1. Interface logic is necessary because of differences between the 68000 and 68020, like the number of data-bus bits and the control-bus signals.

The result is a six-layer board measuring 80 by 100mm holding the two processors, two 20-pin Pals and one t.t.l. device – ten components in all. It plugs directly into the Kaycomp but it should also be suitable for other 68000 systems provided that there is no bus master other than the main processor and that the memory and peripherals can cope with the 68020's shorter bus cycle.

In order to significantly reduce component count and hence board size, I have used programmable-array logic for the interfacing. A problem for small organizations wishing to use such devices can be the cost of specialized equipment required for development and programming. In this article, I will show that programmable logic devices can be designed without any capital outlay.

Both of the processors have 32bit data paths and communication between the two is over the full 32bit bus. Communications with the rest of the Kaycomp board is over a 16bit bus, making use of the 68020's ability to dynamically adjust its bus size. For readers with small pockets, I have made the 68881 maths co-processor optional. Later upgrading is possible by simply plugging in the i.c. and breaking a link.

Kaycomp's firmware has been extended to cope with the new processors. The monitor has been improved to accommodate some of the new facilities on the 68020 and also to display all the extra registers. I have removed the 40-column option since there are too many registers for a 40-column display.

Both the assembler and disassembler have been extended to cope with all the new addressing modes and instructions, including all the maths instructions for the 68881.

Installation of the upgrade consists merely of altering a link on the Kaycomp board to configure the system for 27256 eproms, swapping the eproms, removing the 68000 and plugging in the platform board. Having only ten components, the board is easily and

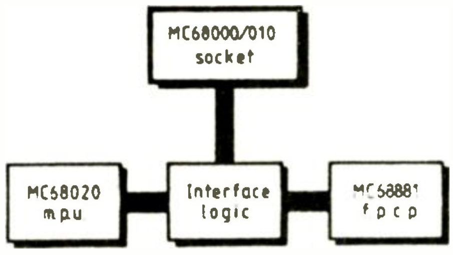


Fig.1. Platform board outline. Interface logic is needed because of differences between the 68000 and 68020.

quickly assembled. Details of how to obtain the board, etc., will be given later.

An application note published by Motorola describes a platform board that appears to the host board to be identical to a 68000 as far as functionality and bus timing is concerned, which guarantees that the platform board will work with any host board designed for the 68000. However, such a board would be excessive for a simple board such as the Kaycomp and there are certain features which can be simplified or left out providing that the design of the target board is known.

There are six basic areas that must be dealt with by the interface logic, these are:

- Dynamic bus-sizing considerations
- Bus-cycle timing modification
- M6800-family device communication
- Interrupt processing
- Bus arbitration
- Co-processor selection

Dynamic bus-sizing considerations. A new feature available on the 68020 is the ability to transfer byte, word and long word operands using its dynamic bus-sizing capabilities, allowing mixing of different-width memory and peripherals within one system. To allow this, the data-transfer acknowledge (DTACK) signal found in the 68000 is replaced by two data-transfer and size-acknowledge ($DSACK_{0,1}$) inputs. The two data strobe outputs (UDS, LDS) are replaced by a single data strobe (DS) and two size outputs ($SIZ_{0,1}$) to indicate the type of transfer.

Since any 68000 host board always uses 16bit data transfers, the inputs are configured to always initiate 16bit transfers, except for accesses to the 68881 coprocessor when full 32bit transfers can be used.

Bus-cycle timing modification. There are some minor differences between the 68000 bus-cycle timing and that of the 68020. Assuming no wait states, a normal 68000 bus cycle takes four clock periods whereas a normal 68020 bus cycle takes three. To correctly emulate the timing of a 68000 would therefore require a modification to the bus timing. However, if it is known that the host board is able to cope with the shorter bus cycle then this removes the need for some of the interface logic, simplifying the design, and speeding up the system.

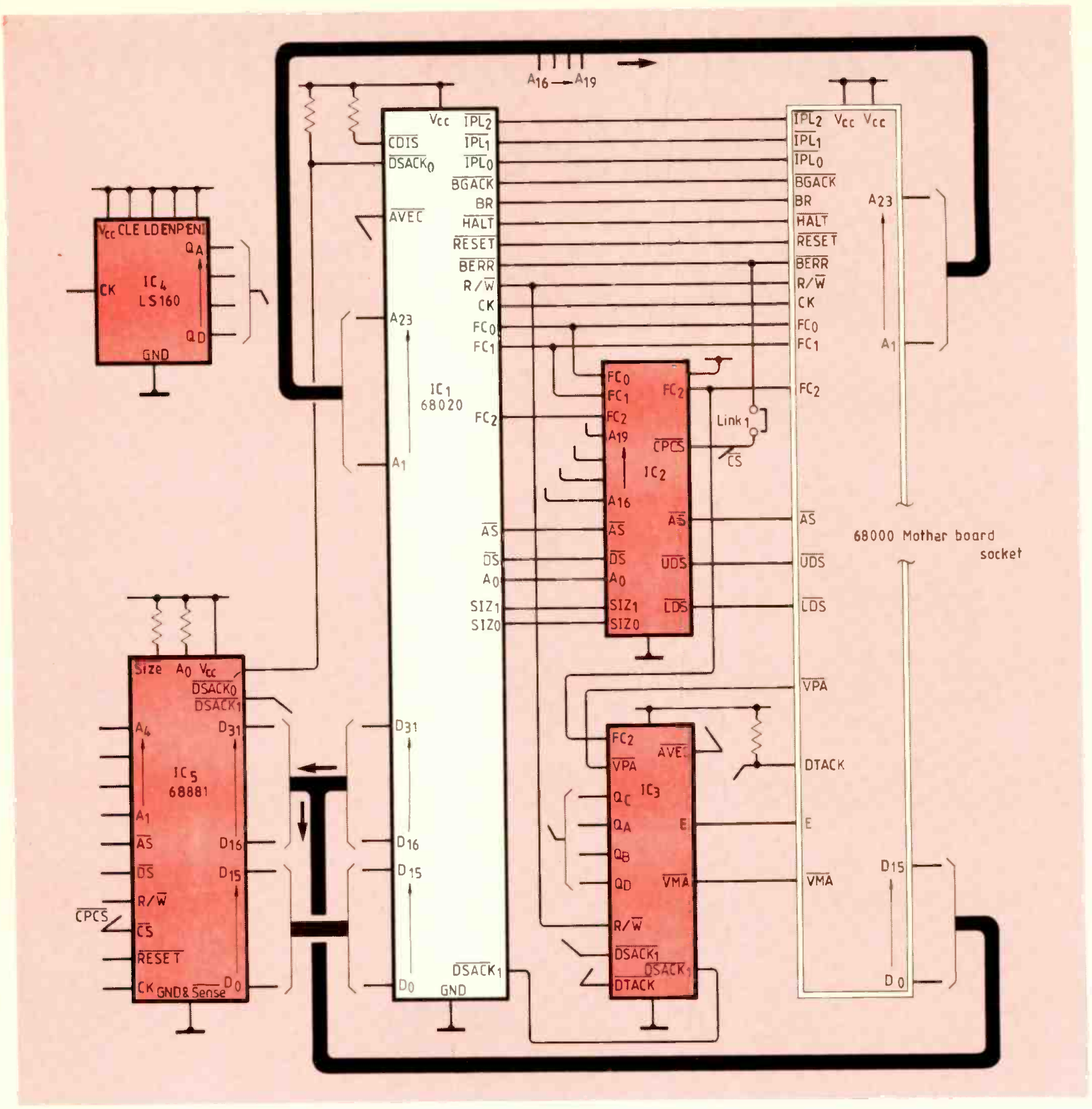
Kaycomp can cope with the shorter bus cycle provided that memories with an access time of 200ns or less (8MHz clock) are used, so I decided not to modify the 68020 bus cycle timing.

M6800 family device communication. On the 68000 the facility was available to interface to 6800-family peripheral devices, which require a synchronous bus transfer. This was initiated by the address decoding asserting the VPA input when accessing a 6800 device area. On the Kaycomp this type of transfer is used for the G64 bus interface.

This synchronous bus interface is no longer available with the 68020 and so an equivalent function must be implemented in hard logic in order to allow the G64 bus interface to continue to be used. This function accounts for most of the interface logic on the platform board.

Interrupt processing. On the 68000 an interrupt-acknowledge cycle is indicated by the three function-code outputs, FC_{0-2}, being all logical one and this is used to produce the IACK signal on the Kaycomp. With the 68020 however, all ones on the function-code lines indicates a c.p.u.-space cycle, which can indicate a breakpoint acknowledge, access-level control, co-processor communication or interrupt acknowledge cycles. Which it is is encoded in address lines A_{16} to A_{19} which have a different binary code allocated for each c.p.u.-space cycle type. The code for an interrupt-acknowledge cycle is all ones on the four address lines.

As the Kaycomp decodes only the function-code lines and not the address lines, FC_2 from the 68020 is gated with the four address lines before going to the Kaycomp and so this modified FC_2 signal will



only go high if all of FC_2 and A_{16-19} go high.

The only other c.p.u.-space cycle used in this design is the co-processor communication cycle.

On the 68000, the valid-peripheral address input, VPA, as well as being used for the M6800 communication mentioned previously is also used to indicate that an autovectored interrupt response is required. This pin is no longer used on the 68020 since synchronous transfers are not supported. It is replaced by AVEC which serves the same purpose as VPA for initiating autovectored interrupts.

Bus arbitration. On the Kaycomp board there is only one possible bus master, the 68000 processor itself. Consequently the platform board design was simplified by leaving out the interface-logic functions required to allow multiple bus masters. The only bus arbitration implemented is that

Fig.2. Two programmable logic i.cs greatly reduce the number of decoding components needed. Both the 68020 and 68881 are pin-grid arrays; their pin references have been omitted for clarity.

required to support co-processor communication. Address strobe signal AS must not be asserted on the host board while co-processor communication cycles are taking place, and so the AS output from the 68020 is gated with the co-processor chip-select signal CPCS produced by the interface logic before being sent to the host board.

Co-processor selection. A co-processor communication cycle is classed as a c.p.u.-space cycle and indicated by all ones on the three function-code lines, as mentioned previously. Whereas the code on address lines A_{19-16} is all ones for an interrupt-acknowledge cycle it is '0010' for a co-processor communication cycle. A co-processor chip-select signal (CPCS) is produced by the interface logic when the code appears.

In conclusion, although designed for the Kaycomp, this platform board should work with any 68000 board provided there can be no bus master on the host other than the processor and that the memory and peripherals can cope with the shorter bus cycle.

CIRCUIT DESCRIPTION

Figure 2 shows the full circuit diagram of the platform board. You can see that the complete circuit consists of the two processors IC_1 and IC_5, two programmable-array logic (Pal) devices IC_2 and IC_3, a t.t.l. decade counter IC_4, a few pull-up resistors, and the adaptor, SKT_1, which connects the board to the host's 64-pin dil processor socket.

The object of this board is that the interconnector, SKT_1, appears to the host board as if it were a 68000, even though a 68020 processor is actually connected. In the case of the majority of the processor pins this means simply connecting a pin on the 68020 to its equivalent pin number for the 68000 on SKT_1.

Address pins A_{1-23} connect straight through, but note that A_0 cannot be routed through as this pin does not exist on a 68000. It is used though in generating the data strobe pulses. The upper address pins of the 68020, A_{24-31}, are ignored as these cannot be used in a 68000 system.

Sixteen of the thirty-two lines of the data bus are also connected straight through, but note that the most significant bit on the 68020, bit 31, should be connected to the most significant bit on the 68000, bit 15, to provide the simplest access mechanism.

Some of the control lines are also connected straight through, namely the interrupt lines, CLK, FC_0, FC_1, BGACK, BR, HALT, RESET, BERR and R/W. An extra connection may also be made to BERR from the co-processor chip-select via a link. This link is inserted on the Kaycomp to allow the firmware to detect the presence or otherwise of the co-processor, which is optional. Remaining processor outputs, FC_2, AS, UDS, LDS, E and VMA are all produced by the interface logic as there is no direct equivalent available from the 68020, and likewise for the two inputs DTACK and VPA.

The interface logic consists of IC_{2-4} and provides two basic functions, control-line modification and M6800 synchronous bus transfers.

Control line modification. A 16L8 PAL device, IC_2, handles the c.p.u.-space bus-cycle decoding. As discussed previously, on the 68000 when all function-code outputs (FC_{0-2}) are at logical one it indicates an interrupt-acknowledge cycle, but on the 68020 it can also indicate a co-processor communication cycle (and other conditions not used in this design). Kaycomp however assumes that if all three lines are one it is an interrupt acknowledge cycle and so it is presented with a modified FC_2 signal which only goes to one if the other conditions are correct for this particular cycle.

These other conditions are that A_{19-16} should also be at logical one and so these address lines, along with the three function-code lines, are all Anded together so that the modified FC_2 signal at IC_2 pin 12 only goes to logical one if all these seven lines are one.

These same address and function-code lines are also used to indicate a co-processor communication cycle which requires the generation of a co-processor chip-select signal (CPCS) to enable the 68881 maths co-processor. Code on address line A_{19-16} in this case though is '0010' and the logic generates a low at IC_2 pin 16 when this condition exists and the function code lines are logical one. Address lines A_{15-13} also indicate which co-processor is being accessed, but as there is only one in this design these are ignored.

When the main 68020 processor is accessing the 68881 co-processor then valid addresses appear on the address bus and the AS line is asserted. It must be ensured though

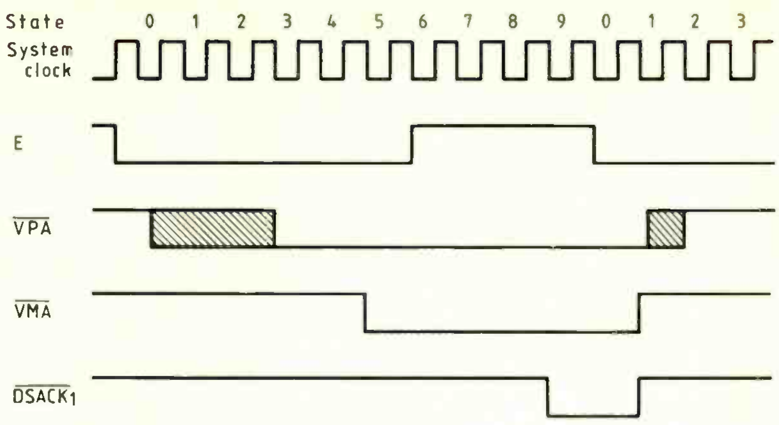


Fig.3. Timing for E-clock, valid memory address, valid peripheral address and data-transfer/size-acknowledge signals.

that the host board does not attempt to place data on the data bus at this time so the AS signal is gated with CPCS so that AS is not asserted if CPCS is.

The remainder of IC_2 is concerned with generating the two data-strobe signals, UDS and LDS. with the 68000 family, unlike some other 16bit processors, each address in the memory map signifies one 8bit byte and not a 16bit word. Consequently, as the 68000 can only access quantities of 16 bits, there is no need for an A_0 signal, the even address byte being considered to be the upper eight bits of the data bus and an odd address byte the lower eight bits. These quantities are strobed in and out of the processor by the upper and lower data-strobe lines (UDS, LDS) respectively.

However, the 68020, with its dynamic bus-sizing capabilities, can access bytes, words, three bytes, or 32bit long words at any base address, even or odd and so the A_0 signal is required and the actual framing of the external data bus connections indicated by the two size signals, SIZ_0 and SIZ_1, and finally strobed by a single data-strobe line DS.

This mechanism is fully explained in the 68020 User's Manual and so I shall just conclude that the equations for generating the UDS and LDS signals which are implemented in part of IC_2 are;

$$\overline{UDS}=DS+A_0$$

$$\overline{LDS}=\overline{DS}+A_0\cdot\overline{SIZ_1}\cdot SIZ_0$$

Remaining modifications of control lines are carried out in IC_3, also a 16L8 Pal device. Production of the E and VMA signals for M6800 synchronous bus cycles will be considered later. Signal AVEC, which is an input to the 68020 indicating that an autovectored interrupt service is required, is taken from the VPA signal from the host board, to which it is equivalent, but also gated with the modified FC_2 signal. This ensures that AVEC is only asserted during an interrupt acknowledge cycle and not during a M6800 cycle which is also indicated by asserting VPA in a 68000 system, this pin serving a dual purpose.

The data-transfer acknowledge signal from the host board, DTACK indicates the termination of a normal bus transfer. There are two equivalent pins on the 68020, $DSACK_0$ and $DSACK_1$, the combinations of codes on the two pins indicating an 8, 16 or 32bit port size. All transfers to and from a 68000 board must be 16bit and this is indicated by $DSACK_0=1$ and $DSACK_1=0$. Consequently $DSACK_0$ is pulled high by a resistor and $DSACK_1$ is driven by DTACK, but Anded with the two other possible sources of $DSACK_1$, co-processor IC_5 and the MC6800 transfer-generation logic also in IC_3.

This leaves us with the production of E and VMA to implement M6800 synchronous bus transfers. Consider how this is done on the 68000. When this type of transfer is required, to access a synchronous peripheral device say, the address decoding asserts low the 68000 VPA input. After recognition of VPA, the 68000 assures that E, which is produced by dividing the system clock by ten with a 60:40 low-to-high ratio, is low by waiting if necessary and subsequently asserts VMA until after E has gone through the next high-to-low transition.

To emulate this, a system clock divide-by-ten circuit is required which is provided by IC_4, and other logic contained in IC_3. The state machine produces the two 68000 signals E and VMA and the $DSACK_1$ signal to the 68020 to terminate the transfer, according to the state table, Table 1. The E clock is produced quite simply from three of the divider outputs, Q_b, Q_c and Q_d.

Table 1. Decoding state table for E, VMA and $DSACK_1$.

Output (Q) d	c	b	a	E	Action
0	0	0	0	0	
0	0	0	1	0	Negate VMA and $DSACK_1$
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	Assert VMA if VPA asserted
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	1	
1	0	0	1	1	Assert $DSACK_1$ if VMA asserted

Two RS type bistable devices are used to produce the two outputs, VMA which outputs on IC_3 pin 17, and $DSACK_1$ which is Anded with other sources of $DSACK_1$ before appearing as an output at pin 18. Both of these bistable devices are reset at the first state, that is when $Q_a=0$ and $Q_c/Q_d=1$, negating (logic 1) the two outputs. At the fifth state, if the VPA input is asserted (logic 0) then the VMA bistable device is toggled, asserting (logic 0) VMA. At the ninth state, if VMA is being asserted, then the $DSACK_1$ bistable device is also toggled, asserting $DSACK_1$. One clock cycle later, after the counter has reset to state zero, both signals are once again reset. Resulting waveform timings are clarified in Fig.3.

CO-PROCESSOR INTERFACE

The 68881 is flexible in how it can be interfaced to the main processor, allowing 8, 16 or 32bit connections. Although an 8 or 16bit bus between the two would have reduced the number of tracks required, clearly maximum processing speed will result if all 32 data bits are used. Thus all 32 data-bus pins on the 68881 (IC_5) are connected to their equivalents on the 68020 (IC_1). The SIZE and A_0 pins of IC_5 are tied to a logical one (V_{cc}) which configures the device in 32bit transfer mode.

Since the 68020 has a built in co-processor interface, other connections are relatively simple: just a few address lines to select the various registers, control lines and a chip-select signal (CPCS) – in fact very similar to interfacing a normal peripheral chip to a processor. Both DSACK pins on the 68020 are driven by the 68881, a low on both indicating to the 68020 a 32bit bus transfer.

The beauty of this system is that all this is totally transparent to the software designer. Attaching the maths co-processor means to the programmer that a new set of instructions and registers have been added to the 68020's. The fact that they are contained in a separate chip is not apparent to the programmer, who just uses the additional instructions as if they were all part of the main processor.

PAL IMPLEMENTATION

When the prototype for this design was first built, it was done using standard t.t.l. for the interface logic, using similar basic circuits to those shown in the blocks for the Pals, Fig.7. Unfortunately it took about 10 t.t.l. devices to implement the interface logic, which made the board rather large and complicated. Programmable-array logic (Pal) devices were therefore used for the final design which reduced the chip count to three and considerably reduced the complexity of the board.

The principle problem with using programmable logic devices can be the equipment required, which could perhaps be difficult to justify for the small development laboratory or educational establishment. I found that with a little time and effort and a Pal distributor who could program the devices from paper it was quite possible to produce the designs with no specialized equipment at all.

PAL CONCEPT

The Pals used on the 68020 upgrade board are one of the simpler types, the 16L8, whose designation indicates logic low or inverted outputs (L) and 16 inputs and 8 outputs. For a 20-pin device, this number of input and outputs means that some pins are dual purpose, programmable as inputs or outputs, as can be seen from the block diagram of the device, Fig.4.

A full explanation of the workings of Pals can be obtained from the various manufacturers' data, but I will briefly explain the basics. A Pal implements the familiar sum of products logic by using a programmable And array whose output terms feed a fixed Or array. Since the sum of products form can express any Boolean transfer function, the Pal's uses are only limited by the number of terms available in the And-Or arrays.

Basic structure of a cell within the array is shown in Fig.5. Working backwards, this shows the Or gate which drives the output, being driven by two And gates, each of which has four inputs connected through programmable fuses to the true and inverted states of two inputs. In an unprogrammed Pal these fuses are intact, but by programming may be selectively blown, opening the circuit to that particular And input, which then assumes a permanent one state.

In practice, the arrays are much larger than this. With the 16L8 device, each And gate has 32 inputs, the true and inverted version of each of the device input pins. Driving each of the eight Or gates there are seven individual And gates. An eighth And gate enables an output inverter-buffer which drives the output pin from the output of the Or gate. A quick calculation reveals a total of 2048 individual fuses. This would be difficult to show by way of a circuit diagram and so a diagrammatic representation is used as is shown in Fig.6.

In this representation, at the point where the horizontal line to the And input (product term) crosses a vertical input line there is a fuse. We need to mark on this diagram crosses where a fuse is to be left intact and no mark where a fuse is to be blown. This 'fuse-plot' can then be handed to the supplier/programmer of the device to pro-

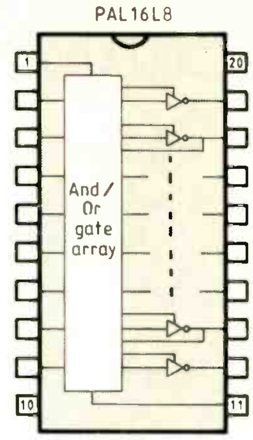


Fig.4. Up to 16 inputs and 8 ouputs are available on the 16L8 programmable-array logic i.c.

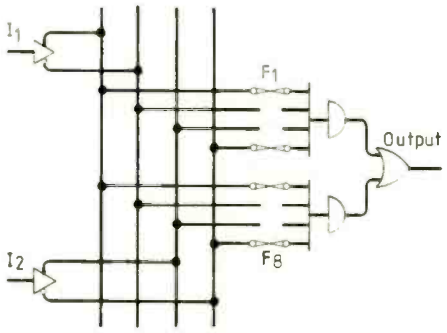
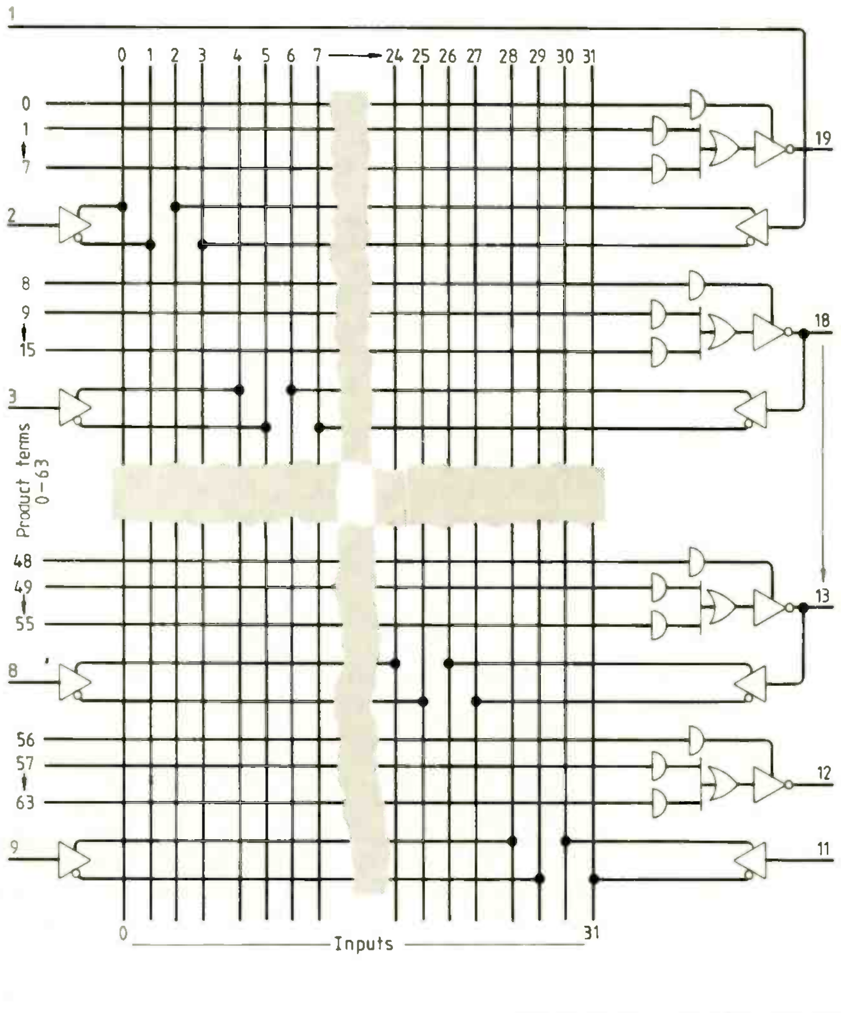


Fig.5. Basic Pal structure for a two-input, one output logic segment.

Fig.6. Logic diagram for the PAL16L8. Eight of the lines on the right can be programmed either as inputs or outputs.



duce a Pal to your custom design. Clearly this will need to be entered into the device programmer manually, which takes time, and hence money. For each of the devices used for this project a charge of around £24 was made. This is a non-recurring cost though as when further devices are required, the first device can be used as a master, saving this charge.

DERIVING THE FUSE-PLOT

We now have a circuit diagram and this has to be translated to a fuse-plot. There are 'schematic capture systems' available to run on various computer systems which can take a circuit diagram and produce the necessary output to the device programmer from which the Pal can be programmed – clearly the best solution, but expensive. A simpler, more readily available software package is a Pal assembler (Palasm) which runs on IBM PC compatibles. This takes the Boolean expressions which define each output and produces the information for programming the device. Even these days though, not everyone has an IBM PC, let alone PALASM, so for this project a manual system had to be devised.

The first question to be answered is can the interface logic be contained in one Pal or if not how many? The main restriction will be the number of inputs and outputs required. I soon saw that more than one would be needed, and after some juggling around the logic was fitted into the 16L8 Pals and one t.t.l. decade counter. The decade counter too could have been implemented in Pal, using a registered output device, such as the

MONITOR ENHANCEMENTS

In order to accommodate the 68020's enhancements and the maths co-processor, the original 68000 monitor with line-by-line assembler has been supplemented with the following commands and modifications.

AS line-by-line assembler. Modifications – 'size' attribute for branch instructions is now 'B', 'W', 'L' for 8, 16 and 32bit offsets respectively, instead of 'S' and 'L'. Default start address is now 400600_{16}. All ram below this address is used by firmware.

CA examine/alter cache address register. Same as An command but for cache register address.

CC examine/alter cache control register. Same as An command but for cache control register. Note that only bits 0-3 are used.

DF examine/alter destination function-code register. Same as An but for destination f.c. register. Only bits 0-2 are used.

Fn examine/alter floating-point data register. Same as An command but for the eight f.p. data registers. Note that this is a 96bit value (24 hexadecimal characters).

FA examine/alter floating-point instruction address register. Same as An command but for the f.p. instruction-address register.

FC examine/alter floating-point control register. Same as An but for the f.p. control register. Only bits 0-15 are used.

FD disable floating-point register printing.

FE enable floating-point register printing.

FS examine/alter floating-point status register. Same as An command but for f.p. status register.

RD register display. Modifications – displays extra registers available on the 68020 and the 68881 registers provided that it is enabled by the FE command.

SC single-step on flow change. Similar to SS command but only stops and displays registers on a program flow change, i.e. when a jump or branch is executed.

SF examine/alter source function-code register. Same as An command but for source function-code register. Note that only bits 0-2 are used.

TC trace on flow change. Similar to TR but only displays registers on program flow change.

VB examine/alter vector base register. Same as An command but for the vector base register.

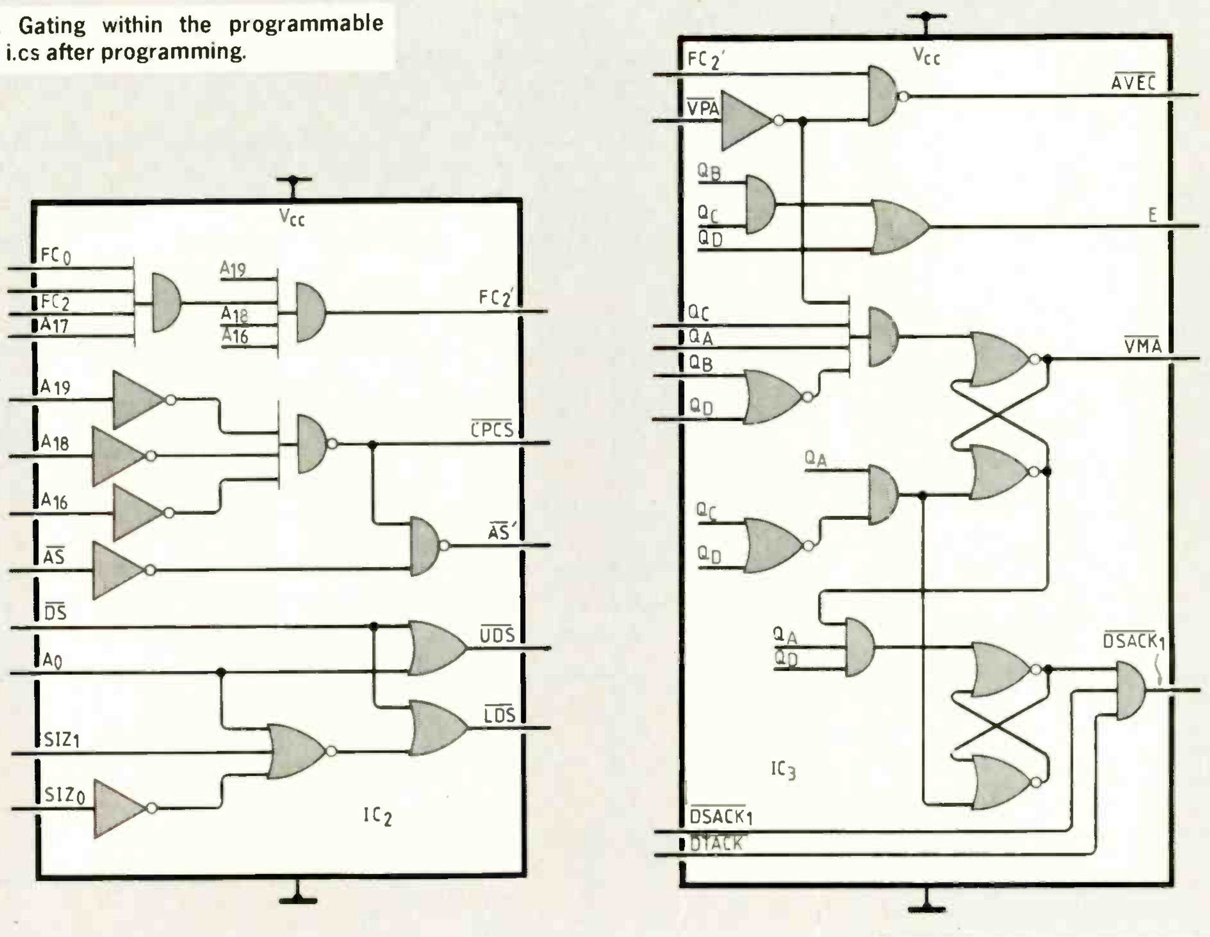


Fig.7. Gating within the programmable logic i.cs after programming.

16R6, but the configuration used proved to be just as simple and more cost-effective in this instance.

The circuit required to produce one of these Pals, IC_2, is shown in Fig.7(a). You can see that 12 inputs and 5 outputs are needed, which is within the capabilities of the device. Next I had to decide which pins were to be allocated to each function. One of the great things about programmable logic is that you can assign the pins to suit your p.c.b. layout, making this easier and more flexible. The allocation chosen is shown in Table 2.

The circuit diagrams we have for each output of the device though is not in a form

Table 2. Allocation of pin functions for Pal IC_2.

I/O	Function	Pin	Pin	Function	I/O
I	FC_0	1	20	V_{cc}	–
I	FC_1	2	19	$\overline{AS}'$	0
I	FC_2	3	18	$\overline{UDS}$	0
I	A_{16}	4	17	$\overline{LDS}$	0
I	A_{17}	5	16	CPCS	0
I	A_{18}	6	15	LDS	I/O
I	A_{19}	7	14	SIZ_0	I
I	$\overline{AS}$	8	13	SIZ_1	I
I	$\overline{DS}$	9	12	FC_2'	0
–	GND	10	11	A_0	I

which reflects the internal structure of the Pal and so the next process is to translate the circuit diagram for each output of Fig.7(a).

Internally, this Pal has four gates in series, the appropriate inputs of each one of which must pass through to produce the output. These are, first either an inverting or non-inverting gate, an And gate, an Or gate, and finally an inverting buffer which may be tri-stated if desired. In this design all used outputs are permanently enabled.

Taking the simplest block first, the UDS output, from Fig.7(a) you can see that this output must go low if both DS and A_0 are low, and high at all other times, i.e. an Or gate. How this is implemented in the Pal is shown in Fig.8. Two of the inputs to the And gate are used and driven from DS and A_0 after each is inverted. Unused And inputs must all be tied permanently high for the gate to function and this is done by blowing all the remaining fuses in that product term.

Thus the And output will be high only if both DS and A_0 are low. To get the required output from this merely requires inversion, the Or gate not being required. If the output from the And function is applied to one input of the Or gate and all the other inputs are low, then the Or gate will act as a non-inverting buffer and have no effect. The output inverter is permanently enabled, producing the required output.

With the circuit diagram in this form, we have the logic function defined in the same manner as the internal structure of the Pal and the fuse plot diagram can now be marked up. Figure 9 shows the section of the fuse plot for the UDS output. The top And gate drives the output buffer which is to be permanently enabled. This is done by blowing all fuses on its input, denoted by no x's on that product term line.

For the next And gate shown, the one that is used, x's on inverted input lines from DS and A_0 indicate these fuses are intact, while all other fuses are blown. The remaining six And gates are not used and must output a logic low to the Or gate. If the fuses for both

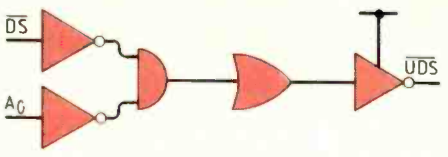
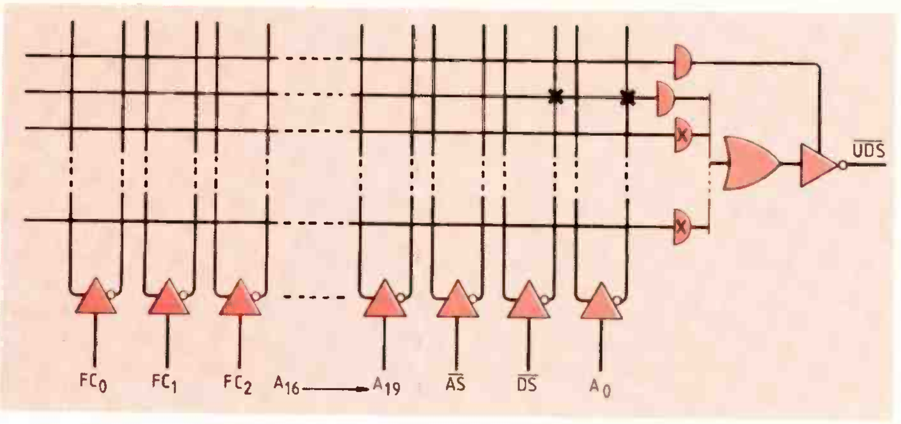


Fig.8. Pal equivalent circuit for the upper data-strobe.

Fig.9. Fuse plot for upper data-strobe signal decoding.



the true and inverse of an input are left intact to an And gate then the output must always be false (low) whatever the state of the input. So all fuses to the unused And gates are left intact and this is indicated by placing an x inside the And-gate symbol.

This output was a fairly simple one, but the LDS output is a little more complicated, Fig.10. You can see that the first output gives the inverse of what is required and so this output, which is also an input, is fed back in and inverted to finally give the required function. The first output pin is left unconnected externally. Figure 11 shows the fuse-plot equivalent of this.

FIRMWARE MODIFICATIONS

In order to accommodate the new processors that can now run on the Kaycomp board, the firmware has been extended to cover the extra features. The assembler and disassembler have obviously had to be extended to cover the extra instructions and addressing modes. And the monitor has had to be altered considerably to enable the extra registers to be accessed individually by various extra commands and also displayed by the 'RD' command.

Since the 68881 maths co-processor is optional, the monitor also has to determine

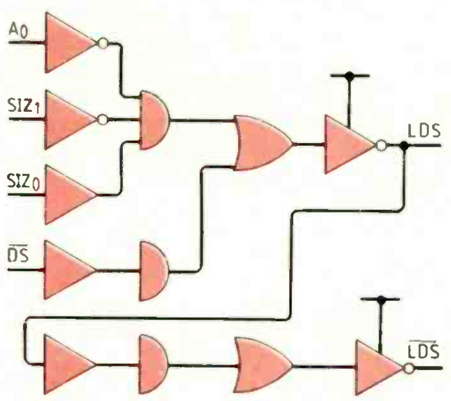
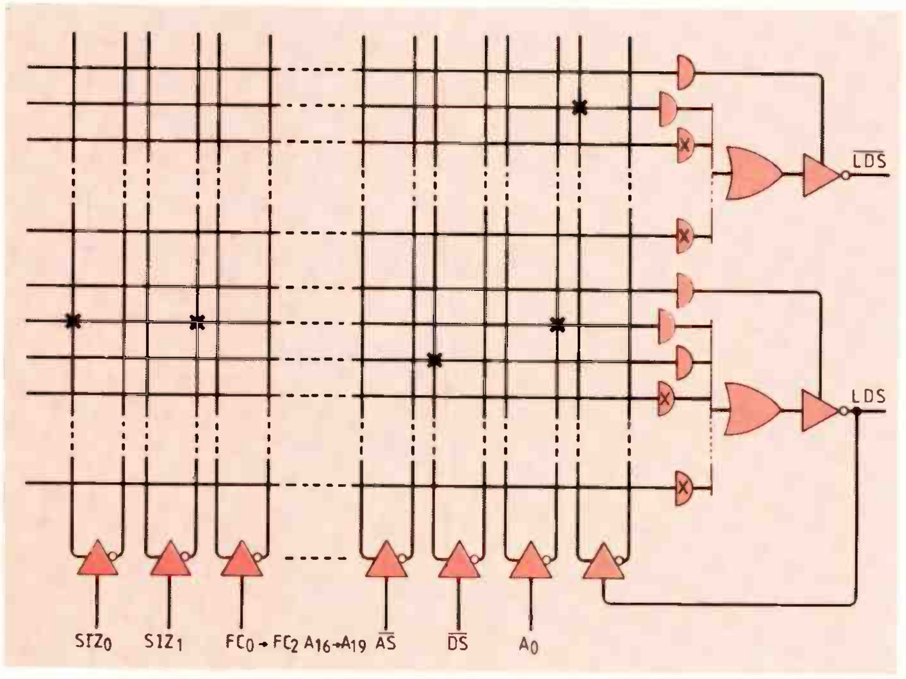


Fig. 10 Pal equivalent circuit for the lower data strobe.

Fig.11. Lower data-strobe output fuse plot.



at reset whether or not it is fitted. If it is not fitted, a link from the co-processor chip-select line to the 68020 bus-error input, BERR, must be fitted (LK_1) which causes a bus error if a co-processor access is attempted. At reset this is intercepted and used to prevent further access by the monitor of the co-processor, instead of causing normal bus-error exception processing.

Otherwise the monitor has been kept the same wherever possible. One alteration is that the 40-column display mode has been removed, the reason being the extra number and size of the registers that have to be displayed make it really unworkable with anything less than an 80-column display. Also consider the maximum length of instruction that now has to be handled by the assembler and disassembler, for instance,

```
00400612 33DO19370012345600056789213300ABCDEF00FEDCBA\
MOVE ([$123456,A0],D1.L,$56789),([$ABCDEF.A1,D2.W]$FEDCBA)
```

which is difficult enough to make clear with an 80-column display!

All these extensions have virtually doubled the size of the Kaycomp firmware package which is now about 60Kbyte. Eproms therefore must be at least 32kbyte each, so 27256 devices must be used.

ASSEMBLY AND INSTALLATION

Because of the use of Pals the board itself is quite simple and requires very little assembly. Parts for this design are available from Magenta Electronics Ltd whose address is given later. These include a high-quality six-layer plated-through-hole p.c.b. which makes the design very compact and reliable. There are only ten components in all to be mounted on the board and their positions are shown by a silk-screened legend. Remember though that the 64-pin board-to-board adapter is fitted on the underside of the board and soldered on the top (component) side. This adapter has pins which are a larger diameter at one end than the other. The larger diameter ends must be soldered into the new board. Also, if the 68881 is not being fitted at this stage, wire together the two holes of LK_1.

To fit the kit, the new 27256 Eproms should be fitted first after altering the links on LK_1 on the Kaycomp to suit 27256 eproms (see November 1985 issue, page 53). Next remove the 68000 i.c. from the Kaycomp and in its place plug in the new board, ensuring it is the correct way round. The boards are now ready for powering up.

Getting the new upgraded Kaycomp working is probably the easiest part of upgrading. The harder part is upgrading the user to be able to program the two new processors, particularly the maths co-processor which requires new ways of working for the programmer but has great potential for real-time mathematical applications[1].

Two books therefore will be essential to the new user. These are the manufacturer's user manuals for the 68020[2] and the 68881[3] processors.

Magenta Electronics is at 135 Hunter Street, Burton-on-Trent, Staffordshire DE15 0AB, tel: 0283 65435.

References
1. Using 68020 co-processors, Burns and Jones, *E&WW* May 1987, pp.535-537.
2. Motorola MC68020 32-Bit microprocessor user's manual, part number MC68020UM/AD.
3. Motorola MC68881 floating-point coprocessor user's manual, part number MC68881UM/AD.

Bob is now Development Manager for computer systems and consultancy company, Barron McCann.

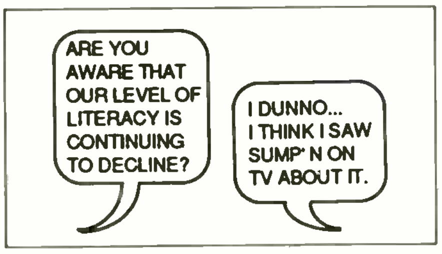
BOOKS

Numerical recipes in C: the art of scientific computing, by William H. Press, Brian P. Flannery, Saul A. Tukolsky and William T. Vetterling. Cambridge University Press, £27.50. Comprehensive handbook of scientific computing in the C language, the follow-up to an earlier work devoted to computing in Fortran and Pascal. Its 17 chapters and 200-odd program examples cover such areas as linear equations, differential equations, interpolation and extrapolation, random numbers, sorting, finding roots, eigensystems, FFTs, statistical methods and data modelling. Program examples are commented and fully explained in the accompanying text. In keeping with the informality of their title, the authors present their material in a sleeves-rolled-up, practical, and very readable style, and are not above tossing in the occasional touch of humour (illustrating one of their sections on C control structures is a program called 'badluk', which calculates occurrences of Friday the 13th when the Moon is full). Useful reading lists follow each section. C programmers will undoubtedly find this book invaluable; and in view of the sheer bulk of information it contains, they will probably think it very reasonably priced too. Hard covers, 755 pages. Recipes and examples are available from the publishers on disc (IBM PC format).

Electronics and computer acronyms (revised second edition) by Phil Brown. Butterworths, £24. Glossary of electronic technology from A for atto (10^{-18}) to Zr for zirconium: 2500 entries covering every branch of the art, including the commonplace, the esoteric and a few surprises. For example, none of us in the office knew until now that BNC (as in connector) stood for Bayonet Nut Coupling. This sent us scurrying to look up SMA – but alas, it wasn't included. One error we spotted is that CTS, in serial communications links, stands for clear-to-send, not clear-to-stand. Useful, if expensive. Hard covers, 272 pages.

Computer Lib, revised edition, by Ted Nelson. Tempus (Microsoft Press, Penguin Books), £14.95. "This book is a multi-user, high-resolution, demand-paged, hardware-software, read-mostly memory, retrieval and display system with real-time interaction, tactile interface, audio and video feedback." Get the idea? Its cheerfully anarchic typography occupies territory somewhere between early *Private Eye* and *The Face*, and the scrapbook-style pages are a diverting mixture of computer facts, fantasy, ideas, opinion, jokes, cartoons, eccentricity, wisdom, whimsy and aphorisms (sample: "Wordstar is the second hardest video game yet made").

THE DECLINE OF MIND


ARE YOU AWARE THAT OUR LEVEL OF LITERACY IS CONTINUING TO DECLINE?
I DUNNO... I THINK I SAW SUMP' N ON TV ABOUT IT.

In fact, it's *two* scrapbooks – the second half, entitled "Dream Machines", is printed upside down, working from the back cover inwards, which means that two readers can enjoy the book at once (this must be the multi-user angle). Besides the humour, there's plenty of serious analysis of the computer world: a favourite target is the industry's *éminence bleue*, IBM, which comes in for a good deal of criticism over its business policies. "A computer cult classic returns", says the blurb-writer, and who could dispute that? Enormous fun to dip in to. Large format paperback, 235×247mm, 305 pages.

• Following the appearance of our Research Notes item, "Building with atoms" (May 1988, page 452), a reader draws our attention to a book by an MIT visiting scholar, Eric K. Drexler, called "Engines of Creation".

This, he writes, discusses the full implications and implementation of this technology, in all fields, from mechanical engineering through computing and telecommunication to medicine.

Nanotechnology, the building of devices using individual atoms and molecules as working parts, is likely to make a major and profound change to humanity. Drexler discusses its use for good or ill. A real age of plenty could result. The medical implications are profound, leading to cures of cellular diseases such as cancer and ultimately to the immortality of the individual and possible revival to youthful good health of those being frozen upon death today.

"Engines of Creation" also discusses possible abuses of the technology, and the implications of its effects on present problems of society. The book is published by Doubleday at just under $18. If any reader has difficulty in getting a copy through the usual channels, a limited number of copies are available at £11.20 each, post free, from John de Rivaz at West Towan House, Porthtowan, Truro, Cornwall TR4 8AX. (Despatch by first class post within 28 hours; your money back if sold out.)