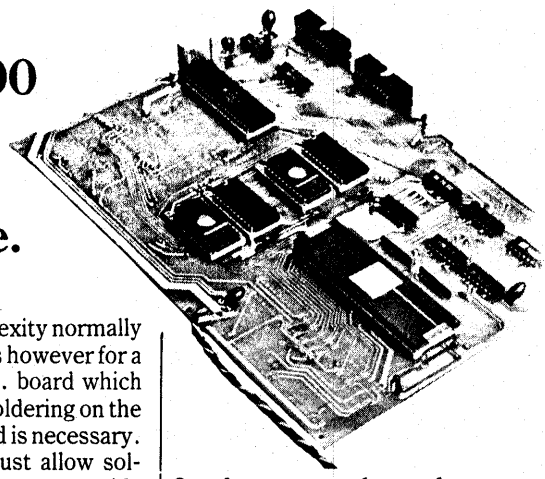
TR 23/2/87.

68000 board

by R.F. Coates

The £100 Kaycomp — Bob Coates, 68000 computer board for engineers, students and enthusiasts — is developed using a terminal and takes a G64 bus interface.



Our front cover shows the 68000 board in its fullest form with G64-bus interface — this is the basic version.

Kaycomp is a low cost computer board using a Motorola 68000 microprocessor with 16-bit data bus. It is designed for use either as an evaluation/educational tool or as the processor board of a larger system, connecting to a wide range of readily available peripheral cards through its G64 interface bus.

Programs can be entered using a terminal to gain access to Kaycomp's 23 function monitor program Kaybug. There's also an optional line-by-line assembler available to speed up program development. Alternatively, the board also links to a host computer with assembler/compiler facilities. Communication software is included in the monitor program.

The 68000 microprocessor has a 32-bit internal structure. Eight data and seven address general-purpose registers are available to the programmer, all 32 bits wide. Its external address bus is 24 bits which gives a linear address range of 16 megabytes, Fig. 1.

Motorola evaluation kits for the original eight-bit 6800 i.c. were available for around £150. There is a similar kit for the 6809 microprocessor but it was not introduced in the UK. For the 68000 microprocessor, Motorola produce a design module costing £1500 — a tenfold increase over the price of a 6800 evaluation kit.

Of course the 68000 design module is a far more complicated product than its older equivalents and designed to allow evaluation on a wide scale. In many applications though this complexity is not necessary and there is certainly a need for a low cost evaluation system.

I designed Kaycomp so that it could be built in its basic form for under £100. In this form it has two RS232 serial interfaces and general-purpose i/o lines provided by a 68681 i.c., a monitor eprom, a small ram and a full 16-bit 68000 i.c. When expanded, the board has 128Kbyte ram, 64Kbyte eprom, two serial interfaces, a 68230 peripheral i/o device and a bus interface which allows connection to standard peripheral cards.

Large systems nowadays can have many processors and direct memory access controllers working together on the same bus to multiply processing speed. I considered that this feature was not essential to learning about and evaluating the processor and leaving it out saved a lot of peripheral logic. If you are interested in the type of work that requires multiprocessing, £1500 won't normally be a problem.

An external bus interface probably isn't essential for training and evaluation either, but I included one to increase the usefulness and versatility of the board. VME bus is the obvious choice for a 68000 processor board but the cost of implementing it is very high. To illustrate, one manufacturer produces a single Eurocard wire-wrap board for prototyping containing just VME interface chips for £600.

My choice was the European G64 bus designed for Motorola eight-bit processors. The interface circuit consists of just three t.t.l. devices. G64 is probably the best supported eight-bit Eurocard bus, with over 200 different cards available from many different manufacturers, but it is not well known in the UK yet. The main UK manufacturer is Syntel of Huddersfield which produces a wide range of processor and peripheral cards, back planes, racking systems, etc.

Kaycomp overview

Figure two illustrates the system. Kaycomp in kit form is double-Eurocard sized, measuring 234 by 160mm, and its p.c.b. is double-sided but to keep costs down, it is not plated through as a board of this complexity normally would be. Layout is however for a plated-through p.c. board which means that some soldering on the top side of the board is necessary. Sockets for i.cs must allow soldering on the component side too.

In order to keep costs down, some systems use a reduced-bus version of the 68000, the 68008, which although internally the same as the 68000 only has an eight-bit data bus and a 20-bit address bus. I decided against using this version. The board accepts either the 68000 or an enhanced version, the 68010, running at up to 10MHz. The 68010 is a virtual memory version of the 68000. This feature cannot be used with Kaycomp, but the 68010 also executes some instructions faster and has some extra ones too.

Memory consists of two eprom and two ram sockets. Two of each byte-wide memory are required to give a 16-bit data width. Links allow eproms with standard

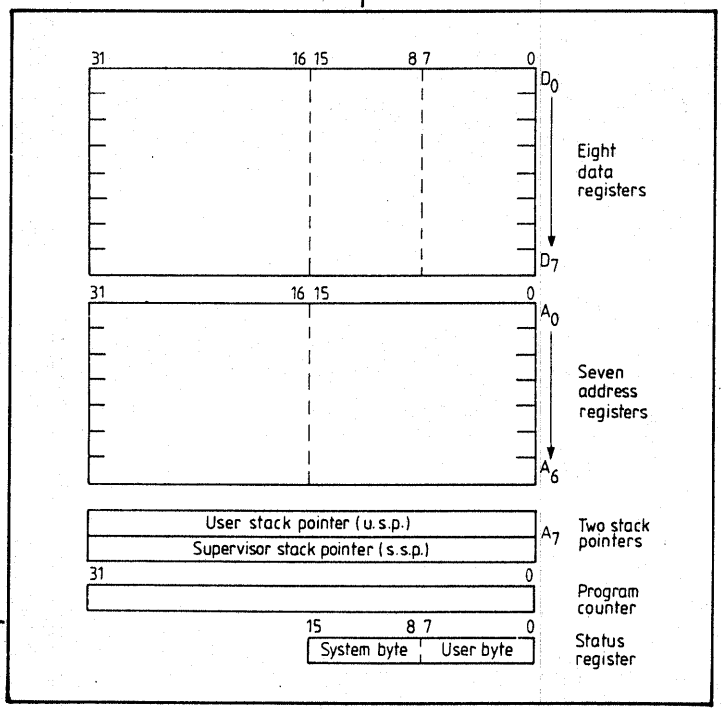


Fig.1. The 68000 has a 24-bit external address bus giving an address range of 16Mbyte.

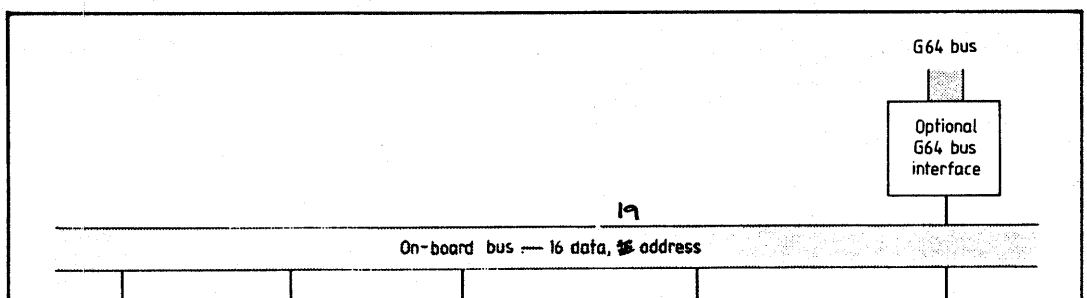


Fig. 2. **Kaycomp uses a full 68000 processor and two peripheral i.cs from the same family. Using these instead of more common 6800 peripherals means higher performance and increases the board's value as an evaluation tool.**

JEDEC pin configurations from 2732 to 27512 to be fitted, giving a range from 8 to 128Kbyte. Note that not all larger eproms conform to the standard pinout, notably those from Mostek.

Ram sockets currently accept either 2 or 8Kbyte static rams, i.e. either 6116 or 6264, to give either 4 or 16Kbytes. The board is laid out though to accept 16 and 32Kbyte devices for when monolithic i.cs become available, which will give up to 64Kbytes of ram. Hybrid 16 and 32K devices are available now but they tend to be expensive. Reasonably priced hybrid 32Kbyte rams consisting of four small-outline 6264 i.cs on a ceramic substrate are produced by Digital Memory Systems, the DMS8832-15PC, and by Hybrid Memory Products, the HMS 62832.

To allow programs to be developed and written on the board, there's a monitor program which fits into two 2732 eproms. This monitor, Kaybug, requires connection of a separate RS232 terminal. If a terminal is not available, many home computers such as the BBC microcomputer have an RS232 port and can be made to act as a dumb terminal. With this in mind, the monitor program can easily be set to produce either a 40 or 80-column display by a keyboard command.

A second RS232 serial port on Kaycomp can be used to connect the board to a host development system or mini/microcomputer. Both RS232 ports come from a 68000 peripheral i.c., the 68681 dual asynchronous receiver/ transmitter or duart. The 68681 internal oscillator requires only a 3.6864MHz crystal. A cheaper 3.579545MHz American colour tv crystal will suffice; data rates will be a little out but still within the required tolerance.

For full-speed operation, the processor requires a separate 8MHz crystal clock but if speed is not important, the duart 3.6MHz clock may be used. If you need to used the serial ports in an application, there's another version of the monitor program available which allows you to develop programs through an external G64 dual serial port card.

Parallel input/output is provided by another 68000-family i.c., the 68230 peripheral interface/timer. This optional i.c. is not used by the monitor and all of its facilities are available for user applications and evaluation.

Finally, there's the optional G64 bus interface which consists of three t.t.l. bus-interface i.cs. There are two sections in the G64 bus memory map, one for memory addresses and the other for peripheral addresses. The peripheral area consists of a 1Kbyte block somewhere in the memory map which is decoded on the processor board. Valid peripheral addresses are denoted by assertion of the VPA signal, which is not to be confused with the 68000 signal of the same name.

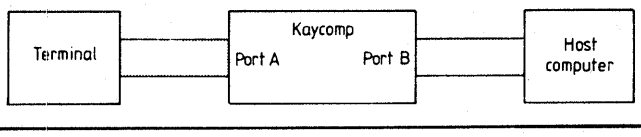


Fig. 3. **Kaycomp can be used by simply connecting a dumb terminal but in this configuration, it is effectively connected in the terminal line from the host computer. In this way, software developed using 68000 assemblers and compilers on the host can be fed directly into the board.**

On Kaycomp, the G64 bus is provided solely for the addition of peripherals. The on-board memory is potentially quite large and capable of operating at much higher speeds than would be possible with memory operating through the interface bus.

Monitor software

Kaycomp's monitor program Kaybug allows you to enter and debug programs and exercise all the facilities on the board. Commands allow you to display/alter memory, set break points and run or single-step trace through programs. Registers can also be altered. Kaybug contains all of the usual monitor features.

At a basic level, Kaybug allows programs to be hand written and typed into memory from a simple terminal using the 'memory open' command. There is an optional line-by-line assembler to simplify this job. If a home computer is used source code can be written, edited and stored using the computer's facilities. When ready, the source code can be sent for processing by the Kaycomp line assembler. Object code is then produced and loaded into ram as each line is entered.

If you have a development system or development facilities on a micro or minicomputer, the second serial port allows program transfer. Kaycomp is then effectively connected in the terminal line from the host computer as shown in Fig.3.

One Kaycomp command allows the board to become 'transparent', i.e., the terminal communicates directly with the host computer as if the board did not exist. Programs can then be written and assembled or compiled in a high-level language according to the 68000 program development software available on the host computer. Another monitor command allows resulting object code to be loaded into Kaycomp's memory in Motorola S-format ready for running. The procedure may vary slightly depending on the host system used but this is a common way of developing programs.

Alternatively, a computer with 68000 'cross-software' can be made to act as both a terminal and development system, Fig.4. A monitor command allows object code to be loaded through the terminal port; the host port is not used.

Before you can understand the

circuit, you need to know a little about the 68000 processor, Fig.5 More detailed descriptions are given in the Motorola Data Manual and the MC68000 Microprocessor User's Manual.

About the circuit

Clock drive. The clock input is a t.t.l. compatible signal which is internally buffered for development of the processor internal clocks. There are 68000 processor versions with clock speeds from 4 to 16MHz faster versions are expected.

Address/data buses. These two buses are fairly straightforward. There are 16 data lines and 23 address lines but there is not an external A_0 address line. Addresses are considered as being byte sizes, i.e. eight bits, and although A_0 is used internally, the address bus is only capable of generating even-number addresses.

Asynchronous bus control. Bus control is a little different to that of previous eight-bit processors in that bus transfers between the processor and memory/peripherals are asynchronous.

On the 6800 for instance bus transfers are controlled by a synchronous timing signal E. This is an equal mark/space ratio signal upon which all bus timings are based. In the case of writing to memory the processor sets up the address bus and read/write signal in the first (low) half of the bus cycle and sets up data to be written in the second half. At the end of the cycle, the E signal returns low and data is latched into the memory.

When reading, the processor presents memory with the address and expects it to have data ready on the bus by the time that the E signal falls to latch the data bus into the processor. This means that the system designer must make sure that memory or peripherals used are capable of operating at the speed required by the processor or, more likely, that the processor clock speed is slow enough to suit the slowest device in the system.

In the 68000, this problem is overcome by using asynchronous bus transfers. The processor sets up the bus in the same way, but it then asserts an address signal called AS and holds the bus until it receives a data transfer acknowledge signal, DTACK, back from the memory or peripheral. DTACK signals from the various system elements are wire-or'd together before entering the processor. This ensures that each part operates at is highest speed.

Peripheral i.cs in the 68000 family produce the DTACK signal but extra circuits are required for this if peripheral devices from other families are used.

Accessing bytes. No A_0 address line is available so some means of implementing byte read/write operations is required. Two signals handle this, upper data strobe UDS for even byte locations and lower data strobe LDS for odd locations. For a normal 16bit word transfer, both signals are asserted.

Figures six and seven summarize the various bus transfers. Figure six shows a read and then a write cycle with no wait states inserted. After setting up the address bus the processor asserts AS, UDS and LDS and then waits for DTACK which it responds to by releasing the three signals. At that point, the addressed device must also release DTACK. If a slow device is addressed, it can be seen that wait states are inserted by the processor after S4 until DTACK is received. Figure 7 shows the action of UDS and LDS when addressing bytes.

68000 peripheral i.c. accesses. Asynchronous bus accesses work fine with 68000 peripheral i.cs but not with the wide range of 6800 peripherals which do not generate DTACK. There are three control pins on the 68000 especially for 6800 peripherals.

If the address decoding circuit asserts valid peripheral address signal VPA instead of DTACK, it indicates to the processor that the device or region addressed is a 6800 family device. The processor then executes the rest of the bus cycle synchronized to a 6800 type E signal as described earlier. It acknowledges this fact by asserting low the valid memory address output, VMA, which is gated with the device's chip-select signal.

The 68000 E signal, with a 40:60 mark/space ratio rather than 50:50, is at one tenth of the clock signal so a processor operating at 10MHz can access 1MHz 6800 peripherals. A synchronous bus access results in a somewhat slower cycle than is possible with asynchronous transfer.

Interrupt control Seven levels of interrupt can be provided for, which ideally would mean seven interrupt pins. To save on pins though, the seven interrupt levels are turned into three-bit binary, the eighth value, all pins high, indicating no interrupt. Normally, these three pins are fed directly from the three possible interrupt sources. Hence only three interrupt levels can be used, one, two and four.

When servicing an interrupt, the 68000 fetches an address from a vector and continues processing from that address. There are two types of interrupt vectoring though, 'auto-vectored' which is similar to that of the 6800, and vectored, where the interrupting device provides a vector number on the data bus in response to the processor executing an interrupt acknowledge

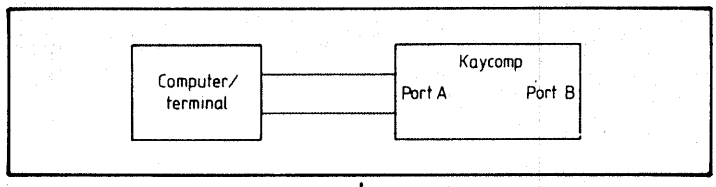
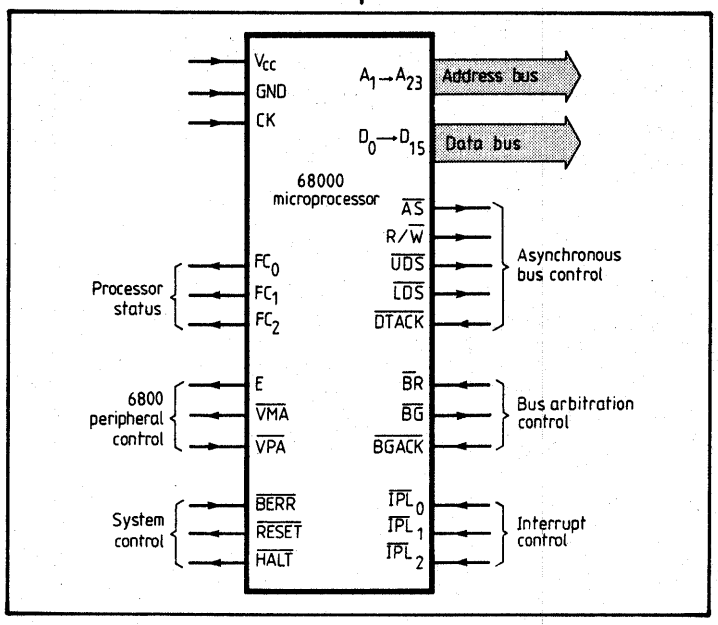
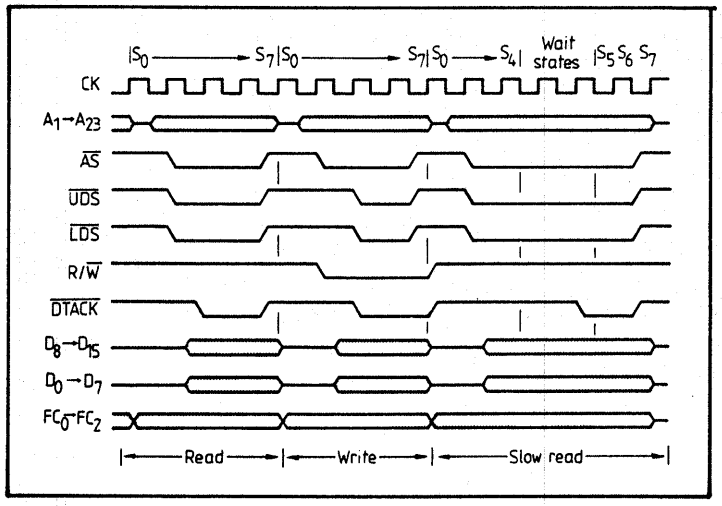


Fig.4. **A computer can be used as both a terminal and host for developing Kaycomp.**

fed from a 0 to 3 line decoder, but on the Kaycomp they are...

Fig. 5. **Input/output signals on the 68000 processor, top, and** Fig. 6, **read-then-write bus transfer with no wait states.**





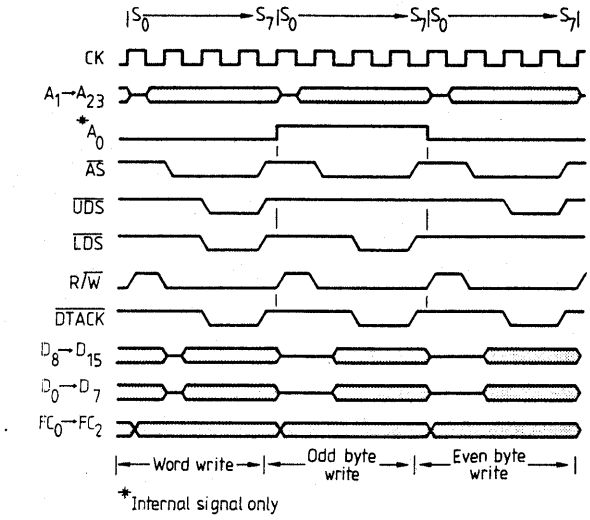


Fig. 7. **Action of upper and lower data strobes UDS and LDS used when addressing bytes. These strobes are needed because the 68000 has no address-line zero.**

cycle. This allows different interrupting devices on the same interrupt level to be serviced by different service routines without polling, which saves time.

Processor status. When processing an interrupt the processor places a unique code on status lines FCO/1/2 of all ones, which is used by Kaycomp to generate an interrupt acknowledge signal, IACK. This signal lets the rest of the board know what is happening. Other states are indicated by the status outputs, Fig.8, but only interrupt acknowledge is used on Kaycomp.

System control. Three signals constitute the system control section, bus error, reset and halt. Bus error, BERR, is not used on Kaycomp. It has two main functions. First, I mentioned earlier that bus cycles are terminated with DTACK. If the circuit does not send this signal, if for example access to non-existent memory location is attempted, the processor stops. A way around this is to have a hardware time-out circuit which generates a bus-error signal if DTACK is not asserted within a given period. A bus-error signal causes exception processing to allow an orderly recovery — hopefully.

Fig. 8. **Function codes indicating the state and cycle type currently executing. These outputs are valid whenever the address strobe is active (low).**

Function code output			Cycle type
FC_2	FC_1	FC_0	
Low	Low	Low	(Undefined, reserved)
Low	Low	High	User data
Low	High	Low	User program
Low	High	High	(Undefined, reserved)
High	Low	Low	(Undefined, reserved)
High	Low	High	Supervisor data
High	High	Low	Supervisor program
High	High	High	Interrupt acknowledge

Components and Support

Individual components complete kits including double-sided p.c.b. and data packs are available from Magenta Electronics, 135 Hunter Street, Burton-on-Trent, Staffordshire DE14 2ST. A kit for the minimum system described is available for £99.

G64 card suppliers include Syntel Microsystems, Queens Mill Road, Huddersfield, HD1 3PG and Thomson Semiconducteur whose UK distributors include Pronto Electronic Systems, 466 Cranbrook Road, Gants Hill, Ilford, IG2 6LE. G64 bus backplanes are available from these and also from BICC-Vero.

Cross-software for various host computers is available from a number of sources, for instance, Microtec Research, Frances Road, Basingstoke, supply 68000 cross-assemblers, Pascal and 'C' crosscompilers for DEC, Data General and IBM PC computers.

Bob Coates will program your pair of eproms (any type) with the Kaybug monitor software for £7 sent to 57 Dalebrook Road, Burton-on-Trent, DE15 0AB. Machine-code listings can be obtained by sending a large s.a.e. marked Kaycomp to our editorial offices.

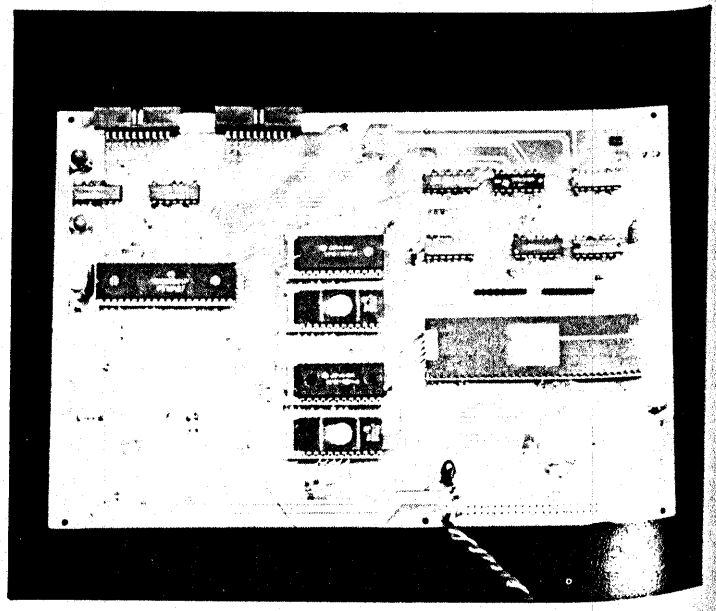
The second use of BERR is in conjunction with HALT. If both are asserted together the processor will attempt to rerun a previous, failed, bus cycle in the hope that it will work the second time. This can be significant in terms of reliability if the processor is controlling say a large plant, but omission of this feature on Kaycomp will probably go unnoticed. If you attempt to access a non-existent location, you'll have to press the reset button.

The HALT pin is bidirectional and the processor can drive it low to indicate a double bus error. Bidirectionality also applies to the reset pin. A reset instruction executed in software causes the reset pin to be driven low for 124 clock periods. All peripheral devices connected to the RESET line are reset.

Taking RESET low externally will have the same effect on peripherals but it will not affect the processor. To reset the processor fully, at power-up for instance, both RESET and HALT must be taken low together externally. If the HALT line is taken low on its own, the processor is held in its current state until the line is released.

Bus arbitration conrol. Three pins, bus request, bus grant and bus grant acknowledge (BR, BG and BGACK) make up this section. These deal with multi-processor/d.m.a. functions which are not available on Kaycomp.

Bob Coates gives a more detailed description of the circuits in his next article.



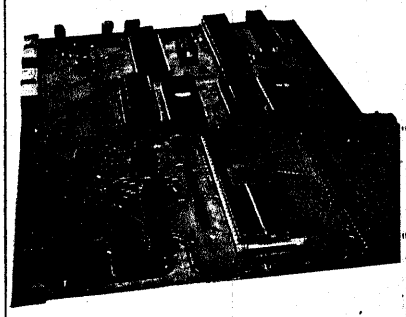
The kit p.c.b. is not a plated-through type; this saves money but requires use of turned-pin i.c. sockets.

FB 23/2/87

68000 board — 2

Bob Coates describes the circuit of Kaycomp — a 68000 microprocessor board with G64-bus option that can be built for £100.

by R.F. Coates



Kaycomp is a low cost computer board using a Motorola 68000 microprocessor with 16-bit data bus. It is designed for use either as an evaluation/educational tool or as the processor board of a larger system, connecting to a wide range of readily available peripheral cards through its G64 bus. This second article describes the circuit.

Address decoding is performed by a three-to-eight-line decoder, IC_9 of Fig.1.(over) The three most-significant address lines A_{21-23} are decoded, splitting the 16M byte memory map into eight 2Mbyte blocks. Five outputs select eprom, ram, 68681 dual universal asynchronous receiver/transmitter (duart), 68230 peripheral interface/timer (p.i.t.) and the G64 bus.

None of these five devices actually requires a 2Mbyte address space; the duart only needs 32 bytes. As a result, each device is repeatedly addressed throughout its 2Mbyte block — addressing for the duart is repeated 65 000 times! This may seem a waste of addressing space but for a small system such as Kaycomp it allows adequate memory capacity while greatly simplifying address decoding. Figure two shows the memory map.

Eprom and ram outputs are further gated with the upper and lower data strobe signals, UDS and LDS, by IC_{10} for defining upper and lower-byte eproms and rams. The three enable inputs of IC_9 also qualify the output strobes.

Address strobe AS allows an output to be enabled only when a valid address appears on the address bus. To ensure that the outputs are only selected when the date bus carries valid data, the two data strobes are combined in IC_6 at pin eight.

Valid data is indicated by one or both of the data strobe signals being low, which occurs later in the cycle than AS. This is to satisfy timing requirements of the 68230 p.i.t. which needs valid data on the chip-select signal leading edge during a write cycle, rather than on the trailing edge transition as with other devices.

Finally, IC_9 pin six inhibits outputs during an interrupt acknowledge cycle during which the processor sets A_{4-23} high. Without the inhibit signal, in the case of user-vectored interrupts output seven would be selected to cause reading of the G64 bus at the same time as the interrupting device is placing its vector on the data bus.

Data acknowledge

To satisfy the requirements of asynchronous bus transfers, an acknowledge signal — DTACK — must be sent by the memory or peripheral back to the processor to inform it that the transfer is complete. If necessary, the processor inserts wait states in the cycle until it receives the acknowledgement.

Peripheral devices in the 68000 family have DTACK open-drain outputs which are directly connected to the processor DTACK input along with a pull-up resistor. Eproms and rams however do not have such an output so an equivalent signal must be created. On more expensive boards DTACK is normally simulated using either a multi-tap delay line or an active delaying device such as a shift register driven by the processor clock to produce a delayed chip-select signal for the DTACK input. There are often different circuits for each type of device and the delay is selectable so it may be set to the optimum required for each type of device on the board.

To keep things simple on Kaycomp, the chip select signals are applied directly to DTACK which means that no wait states are inserted and the memories must be fast enough to allow this. Eprom and ram select outputs of IC_9 are combined in IC_6 at pin eight and then inverted by the open-collector inverter IC_7 at pin eight, pulling DTACK low if either select output goes low.

Outputs one, three and six of IC_9 are not used. If the processor tries to access a vacant part of the memory map no DTACK signal will be generated and so the processor will insert wait states indefinitely. Resetting is necessary to recover the situation.

One output from IC_9 which does not result in DTACK being generated is the G64 bus select signal. This signal requires a synchronous bus transfer. To initiate this transfer, the processor input VPA (valid peripheral address) and not DTACK must be asserted. A ten clock-cycle synchronous transfer then takes place, no acknowledge being required. Output pin seven of IC_9 going low pulls the 68000 VPA input low through IC_8, pin eight.

Memories

In a 16bit system, byte-wide eproms and rams are used in pairs. Memory IC_2 is the lower-byte eprom, IC_3 the upper byte eprom, IC_4 the lower byte ram and IC_5 the upper byte ram.

Lower byte device data buses connect to the processor D_{0-7} lines and upper byte devices to the D_{8-15} lines. Address pins connect to the processor address outputs but as there is no A_0 output, A_1 goes to A_0 on the memories, A_2 to A_1 and so on. Pins 1, 26 and 27 of the eproms go to link one which is wired according to the size of eprom used. Similarly ram pins 23 and 26 go to link one and are also linked to suit the device in use.

Chip enable (CE) pins of each device are driven from the appropriate output of IC_{10}. Output enable (OE) pins connect to the read/write line, inverted at IC_7

2MByte blocks

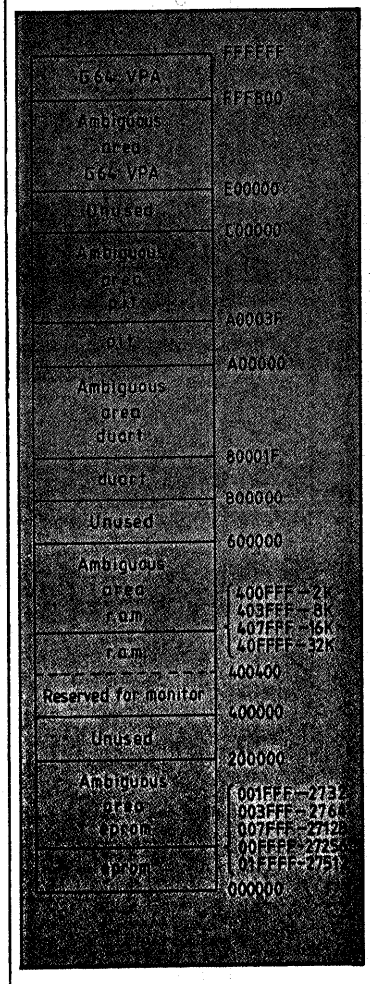
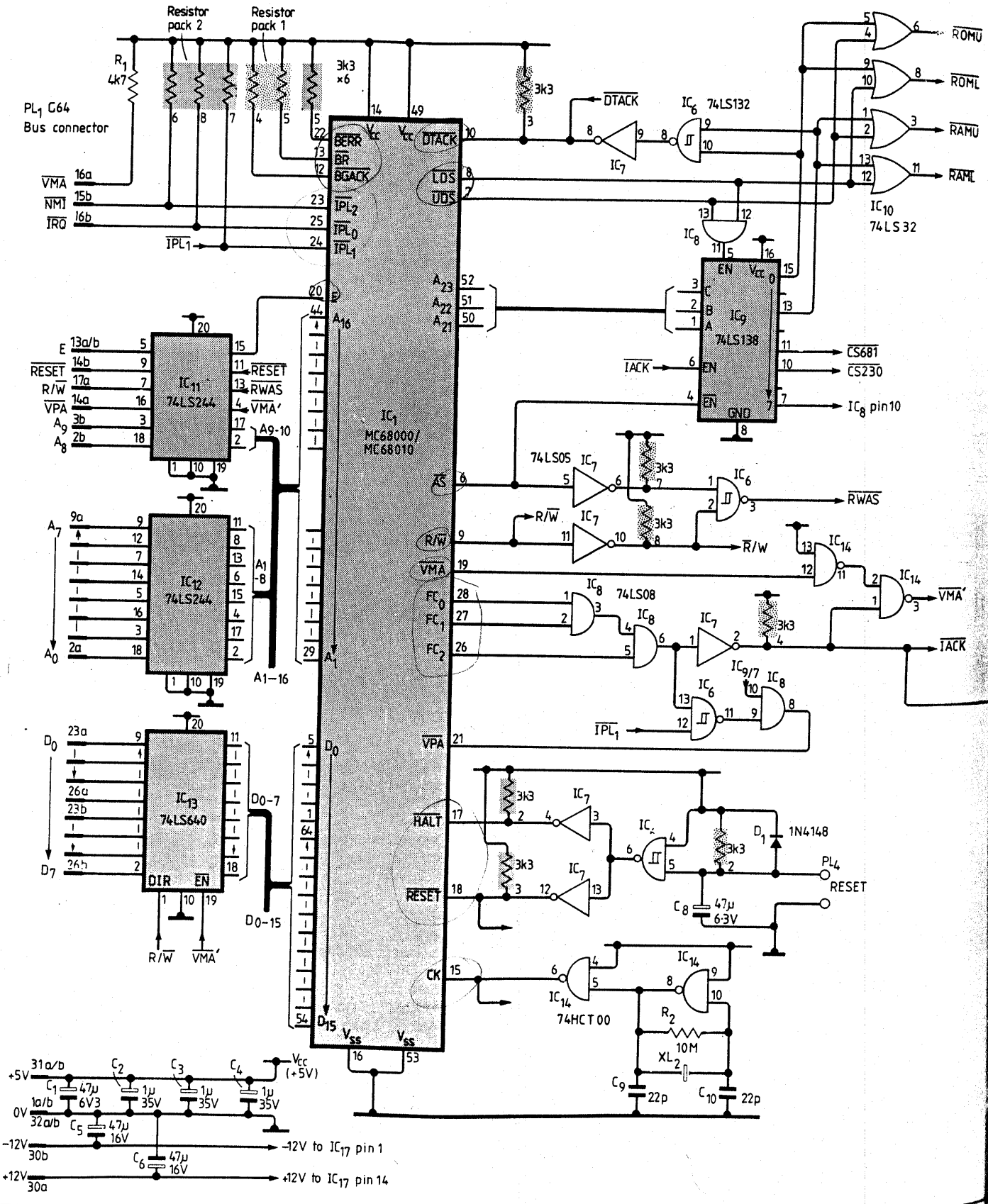
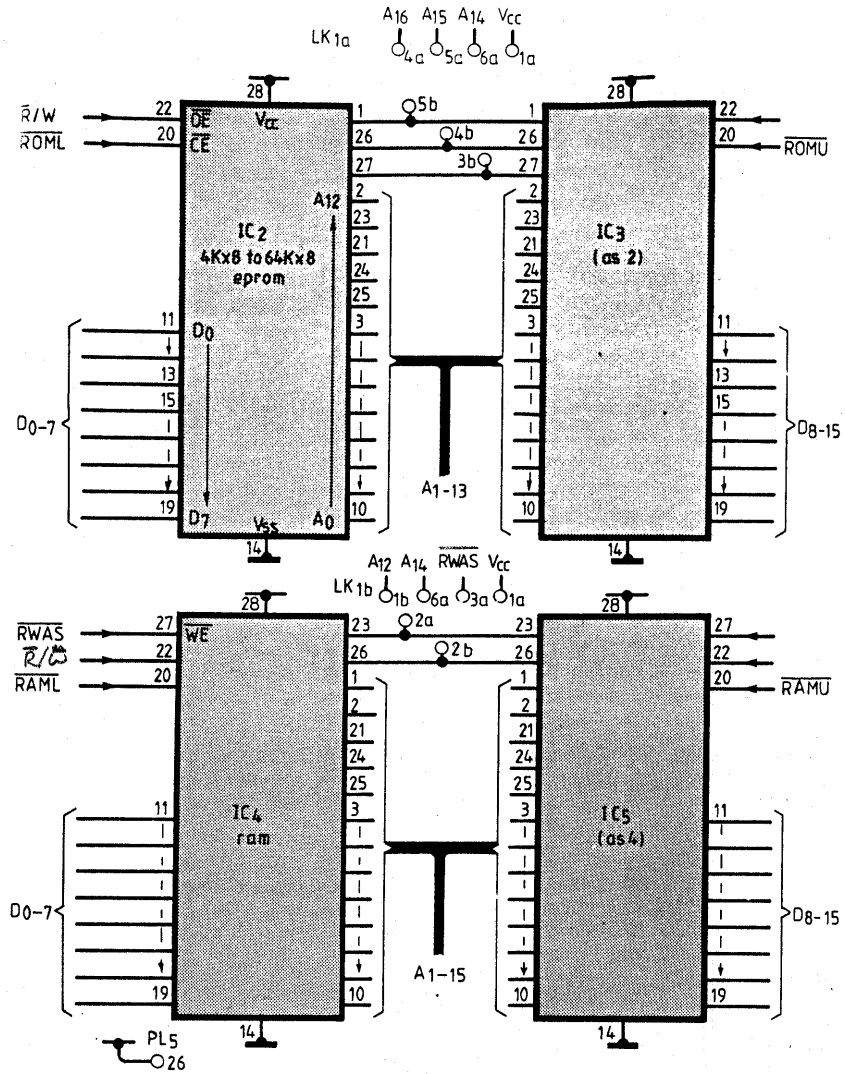


Fig. 2. **Kaycomp system memory map. Simple address decoding means a cheap system allowing sufficient memory for most computer-board applications.**

Fig. 1. **Full circuit of Kaycomp with G64 bus interface and both 68000 peripheral i.cs connected. All these components fit on a double Eurocard sized board included in the kit described last month.**





NOTE :-

Typical configurations

Eprom type

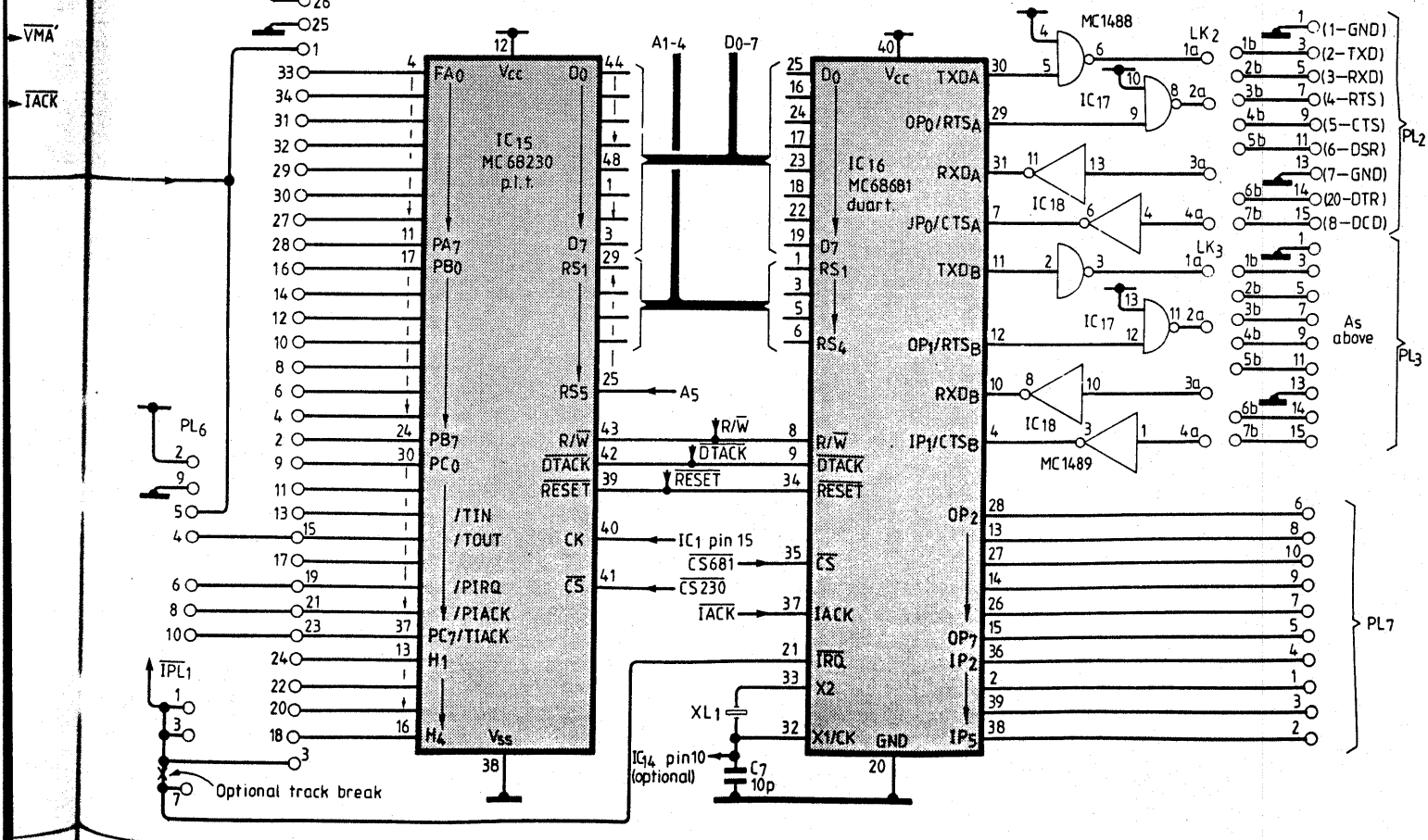
2732	4b – 1a
2764	3b – 5b – 1a
27128	3b – 5b – 1a 4b – 6a
27256	5b – 1a 4b – 6a 3b – 5a

Ram type

2K x 8	(i.e. 6116)	2a – 3a 2b – 1a
8K x 8	(i.e. 6264)	2a – 1b 2b – 1a
32K x 8	(i.e. 8832)	2a – 1b 2b – 6a

Pin-throughs for i.cs not fitted :-

IC_{11}	Pin	10
	"	11
IC_{15}	"	26
	"	27
	"	28
	"	29
	"	39
	"	42
	"	43



pin 10 so that the output buffers are disabled during write cycles. The read/write signal to the memories is the processor read/write output gated with the address strobe.

G64 bus interface

Circuits IC_{11-13} buffer on-board signals to the G64 bus, which can support a wide range of peripheral cards. The G64 bus specification allows a portion of the memory map (normally 1Kbyte) to be dedicated to peripheral cards, the reset of the memory map being available for memory.

Which area is being accessed by a particular bus cycle is indicated by either 'valid memory address' or 'valid peripheral address' being asserted low. These names are not to be confused with the 68000 pins of the same name. To avoid confusion I will call them G64-VMA and G64-VPA .

As Kaycomp is not designed to use the G64 bus for memory expansion, the G64-VMA line is pulled high by resistor R_1. The G64-VPA memory block is only 1Kbyte long so only address lines A_{0-9} are needed. Once again the lack of an A_0 line means that the G64 address lines are driven by the next higher 68000 address line and so the block becomes 2Kbyte in size.

Although G64 bus has 16 data bits, most peripheral cards only use eight bits and so IC_{13} buffers just D_{0-7} to the external bus. As a result, only odd addresses have any meaning when accessing the G64 bus. A byte read or write to an even address causes the processor to use D_{8-15} which are not connected. In the 2Kbyte block, we therefore have 1024 odd addresses on the G64 bus. This may sound a little complicated but it will become clearer later when programs for using the G64 bus are discussed.

The direction of transfer through IC_{13} is controlled by the read/write line and the buffers are enabled by the VMA' signal, which is also used as the G64-VPA signal after buffering by IC_{11}.

Signal VMA' is derived by logic associated with the processor FC_{0-2}, VMA and VPA pins. This section also generates the interrupt acknowledge, IACK, signal. If an address is generated by the processor which accesses the G64 bus, IC_9 pin 7 goes low. This takes IC_8 input pin 10 low, the output of which takes VPA low, initiating a synchronous bus cycle.

During this synchronous cycle, the processor takes VMA low for use as the G64-VPA signal for selecting the relevant G64-bus peripheral device. First though, VMA is qualified by IC_{14} outputs 11 and 3 which only allows VMA if IACK is high that, is, it is not an interrupt acknowledge cycle. This is because an 'auto-vectored' interrupt acknowledge cycle also asserts VMA. The reason for all this gating of signals will become clear later when interrupt handling is discussed.

As mentioned last month, there are two types of interrupt on the 68000, user vectored, where the interrupting device provides a vector number on the data bus, and auto-vectored which is similar to that on the 6800 in which a vector address is fetched from memory. Two interrupt pins, IPL_2 and IPL_0 are connected to G64 bus NMI and IRQ lines. The third, IPL_1, is connected to the duart IRQ output and may be optionally connected to the two interrupt outputs of the p.i.t., PIRQ and TIRQ.

When the processor recognises an interrupt on one of these lines, it starts an interrupt-acknowledge bus cycle. A function code of all ones appears on FC_{0-2} outputs which causes pin six of IC_8 to go high. This signal is inverted by open-collector buffer IC_7 at pin 2 to give a low level IACK during this cycle. When pin 6 of IC_8 goes high, if IPL_1 is also high indicating that this is not the source of interrupt, IC_6 pin 11 and IC_8 pin 8 go low to force VPA low. The same occurs when forcing a synchronous bus cycle.

If VPA is taken low during an interrupt acknowledge cycle, the processor interprets this as meaning a request for an auto-vectored interrupt and not a synchronous bus cycle. It responds by also taking VMA low, which is why VMA' is disabled if IACK is asserted, VMA and VPA serving dual purposes. Thus an interrupt on IPL_2 or IPL_0 (NMI or IRQ) generates an auto-vectored interrupt, but an interrupt on IPL_1 (a 68000 peripheral) does not cause VPA to go low and so generates a vectored interrupt. Why there are two types of interrupt is explained in a later article.

There are three other lines driven on the G64 bus, E which is the 6800 type synchronous clock used to time synchronous bus transfers, read/write and reset.

Halt/Reset

A full processor reset is applied to the 68000 when both halt and reset pins are taken low, either at power-up or when the two reset pins are shorted. At power-up or after opening the reset contacts, the level at IC_6 pin 5 rises to V_{cc} slowly due to charging of C_8 through a resistor. While pin five is below its input threshold voltage, output pin six is high and the two sections of IC_7 hold the halt and reset inputs low. As voltage across C_8 rises, IC_6 output pin 6 will go low, a Schmitt trigger gate here ensuring clean switching. Halt and reset inputs then go high at the same instant allowing the processor to start.

A two-pin printed circuit mounting plug is provided with the kit, which may be connected to a push-switch if required. If 'Kaycomp' is not mounted in a case, squeezing the two pins together provides a crude but effective switch.

Dual uart

The 68681 is a dual asynchronous receiver/transmitter providing two independent serial ports, a data rate oscillator and a number of general purpose i/o pins. Configuration of the serial interfaces and data rate setting is under software control.

Interfacing to the processor is quite simple as the 68681 is a 68000 family peripheral. It is an 8bit peripheral, so only D_{0-7} are used, along with the four lowest address lines connected to register select pins RS_{1-4}, thus the device occupies 32 bytes of memory space (16 registers at odd addresses).

Read/write, DTACK and RESET lines connect straight to the appropriate processor pins, CS connects to output four of IC_9 and IACK is driven from the interrupt acknowledge generating logic. Interrupt request output IRQ is taken to the processor IPL_1 input, thus the duart can generate a level two vectored interrupt.

According to manufacturer's specifications the duart internal data rate generator requires an external 3.6864MHz crystal, XL_1. However, not far away from this frequency is a cheap 3.5795MHz US colour-tv crystal. These have been found to work satisfactorily despite that the data rates are slightly off frequency.

When receiving a serial character the duart, with a clock frequency of 16 times the data rate, first detects the leading edge of a start bit, then counts eight clock periods to align itself in the middle of the start bit. Subsequently every 16 clock periods the duart reads the input level a number of times, up to nine (eight data bits plus one parity bit) according to the data format configuration.

Ideally, each bit is read at its middle point, but with a slow crystal the reading point gradually drifts later in the bit. This is alright provided that the drift doesn't accumulate to the point where it reads over the boundary into the next bit. With a worst-case configuration of eight data bits, one parity bit and one start bit, ten bits are read over 160 clock periods. The distance between the middle of a pulse and the start of the next is eight clock periods, so drift allowable before an error is 8 in 160, or 5%; the difference between the two crystal frequencies is only 2.9%.

This assumes a clean signal which should be the case unless long serial leads are used or the environment is electrically noisy. The kit printed circuit board accepts both colour tv and HC18/U crystals.

Crystal output at pin 32 may also be used to drive the processor clock if processing speed is not important. For this, XL_2, C_9, C_{10} and R_2 are omitted and a wire link used to connect IC_{16} pin 32 to IC_{14} pin 10. Gate IC_{14} does not now have to be an HCT version. An LS version will suffice.

On the peripheral side of the i.c. there are two each of serial outputs and inputs plus eight general purpose t.t.l. outputs and six general purpose t.t.l. inputs. Optionally two of the inputs, $IP_{0,1}$, may be used as RS232 CTS (clear to send) inputs and two of the outputs, $OP_{0,1}$, as RS232 RTS (ready to send) outputs. Outputs and inputs for each serial interface are buffered by an RS232 line driver IC_{17} and line receiver IC_{18} respectively.

There is one problem when using RS232 — every computer, terminal and modem manufacturer seems to have its own interpretation of the standard and one

continued on page 64

REFERENCES
1. Bennett, J.C., Barker, K., and Edeko, F.O. A new approach to the assessment of stereophonic sound system performance, 1984. To be published in *JAES*
2. Harwood, H.D. Stereophonic image sharpness, *Wireless World*, vol.74, no.7, 1968, pp 207-211.
3. Leakey, D.M. Stereophonic sound system, *Wireless World*, vol.66, no.4, 1960, pp 154-160.
Dutton, G.F. Assessment of two-channel stereophonic reproduction performance in studio monitor rooms, living rooms and small theatres, *JAES*, vol.10, no.2, Apri1962, pp 98-105.
Blauert, J. Spatial Hearing. MIT Press, 1983.
de Boer, K. The formation of stereophonic images. *Philips Technical Review*, vol.5, no.2, 1966, pp 51-66.
Bauer, B.B. Phasor analysis of some stereophonic phenomena, *JASA*, vol.33, 1961, pp 1536-1539.
Makita, Y. On the directional localization of sound in stereophonic sound field, *EBU Technical Review*, no.73, 1962, pp 102-108.
4. Harita, Y. Improving stereo at low frequency, *Wireless World*, 1983, pp 60-62.
Buijs, H. Binaural recording and loudspeakers, *Wireless World*, 1982, pp 39-42.
Kitzer, N.J.W. Multiple loudspeaker array using Bessel coefficients, *Electronic Components and Applications*, vol.5, no.4, 1983, pp 200-205.
5. Bose, A.G. Sound Recording and Reproduction, *Technology Review*, 1973, pp 25-33.
Gerzon, M. NRDC Surround Sound System, *Wireless World*, 1977, pp 36-39.
6. Berkovitz, R.A., and Gundry, K.J. British Patent No.1 522 135, 1978.
Bauck, J.L., and Copper, D.H. On Acoustical Specification of natural Stereo Imaging, A.E.S. Preprint no.1649, 66th Convention, 1980.
7. de Boer, K. Stereophonic sound reproduction, *Philips Technical Review*, vol.5, 1940, pp 107-114.
Bracewell, R.N. The Fourier Transformer and its Application, McGraw-Hill, 1978.

broader on-axis images with increases in frequency.

Listening tests

Practical tests have been carried out to validate the theoretical predictions made above. The tests involved subjectively determining image width using the geometric arrangement in Fig.1. The tests were carried out in an anechoic chamber with reverberation time of less than 0.25 seconds for all frequencies down to 125Hz. The signal used was $\frac{1}{3}$-octave band limited pink noise produced by a random noise generator in conjunction with a bandpass filter set (Bruel & Kjaer type 1402 and 1611). Each loudspeaker cabinet housed a single type 8P unit produced by Goodmans Loudspeakers Ltd. Ten subjects took part in the tests.

Each subject, occupying a central position, was asked to keep a fixed head position and looking directly toward the stage centre. The listener was then told to state the location of the image and its width using the dimensions on a bar placed along the stage width. The tests were carried out using 250, 500, 1250Hz, as centre frequencies of the $\frac{1}{3}$-octave signal.

Average practical results of image width versus image position in terms of stage width as shown in Figs 2b, 3b and 4b.

Comparison between theoretical and practical results shows good agreement. The practical curve in Fig.2b for the central frequency of 250Hz shows that image width increases as image is displaced away from stage centre. This is in good agreement with theoretical predictions in Fig.2a. Fig. 3b, 500Hz, shows an almost constant image width. This compares very well with the theoretical results in Fig.3a. At high frequencies, 1250Hz, Figs 4a and b, both practical and theoretical results show that image width decreases as the image moves away from stage centre.

How to overcome image broadening

Image broadening will always exist in a two-loudspeaker system. This is because the quality of the plane-wave signal deteriorates with increase in frequency. At low frequencies the broadening of the image may not be adversely perceived because the image width factor is considerably less than the −20dB level which corresponds to the minimum perceptible change in the effective source distribution[1]. However, at high frequencies the image width factor exceeds the −20dB level and image definitions will worsen and can now be obviously perceived.

To overcome image broadening it is necessary to increase the quality of the plane-wave component of the reconstructed field as frequently increases. The best way to achieve this is to use an array of speakers. The number of speakers in such an array will depend on how much of the high frequency band one needs to correctly reproduce.

An array of speakers is generally regarded as an excellent form of stereophonic sound reproduction[5]. However, the cost and inconvenience of having many speakers makes this approach uninviting. Quite a lot of methods of improving image quality have been proposed while retaining the convenient two speakers systems[4–6]. While these methods may help to improve the accuracy of localization and naturalness of stereophonic images, they do not solve the fundamental problem of image broadening. The use of video cassettes, which have the potential for storing many audio channels may help reduce the cost of having an array of speakers and thus facilitate the use of such a system which is the only real way of solving the problem of image broadening, or indeed of overcoming the general problem of fidelity and usable listening area in stereophonic sound reproduction.

68000 board

continued from page 54

manufacturer's computer won't couple directly to another's terminal without juggling of connections. Hence the proliferation of 'break-out' boxes to help when configuring new arrangements, with leds to show what's happening and patch-links to let you try every combination until it works! Rather than add another interpretation to the standard I have added link areas two and three to allow any signal to be connected to any pin on the interface connector as required.

Plugs two and three for the serial ports are 20-way insulation-displacement type plug headers. Pins have been chosen so that if the standard RS232 25-way D-type connector is used, say on an external panel of a box that Kaycomp is fitted in, then if insulation displacement type connectors are used at each end, a straight through ribbon cable may be used between the two.

On the circuit diagram, pin numbers shown against plugs two and three are those of the 20-way connector while those in brackets give corresponding 25-way D-type pin numbers and their function. Only pins 1 to 8 and 20 are used, 1 and 7 going to 0V, the remainder to link areas two and three. Finally, the remaining general purpose i/o pins (IP_{2-5} and OP_{2-7}) are brought straight out to plug seven, a 10-way insulation displacement connector plug header.

Construction is discussed in the next article.

Notes

Well-intentioned but over-zealous proof reading on behalf of our typesetters led to some anomalies in this article in the October issue.

In the first column on lines 31 and 39, 68000 should read 6800. The same misprint is found on page 53 in the first column on line 26 under the heading 'About the circuit', and in lines 42, 45 and 51 of the next column.

Initially in the article, it is erroneously stated that Kaycomp can have 128Kbyte ram and 64Kbyte eprom, but it is clear from the rest of the text that the correct specification is 128Kbyte rom and 64Kbyte ram. In the third column on page 53, the sentence beginning 'Normally, these three pins...' has a section missing from it. It should read 'Normally, these three pins are fed from an 8-to-3-line encoder but on Kaycomp they are fed directly from the three possible interrupt sources.' Finally, in Fig. 2, the bus has 19 address lines and not 16.

Price of the board with line-by-line assembler added to the monitor program is £109 inclusive.

Two hybrid static rams suitable for Kaycomp memory expansion were mentioned last month, the DMS8832 and the HMS62832. These are manufactured by Digital Memory Systems, PO Box 84, Walton-on-Thames and Hybrid Memory Products of Weymouth Road, West Chirton Industrial Estate, North Shields NE2 97TY.

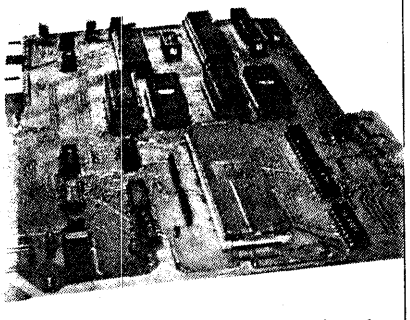
Sockets suitable for top-side soldering are Augat 1800 series, Jermyn 18000 series and Robinson-Nugent ICE series. Augat 510-AG91D terminal strips, Augat 700 series terminal carriers and Jermyn 8500 series terminal carriers are also suitable.

IB 23/2/87

by R.F. Coates

68000 board — 3

Kaycomp's 68000 peripheral interface/timer option adds three eight-bit ports for input and output.



Kits mentioned in the first 68000-board article now include plated-through-hole p.c.b.s instead of ordinary double-sided ones for the same price of £99 inclusive. A line-by line assembler is included for a further £14. See October and November issues for details.

Kaycomp accommodates two 68000 peripheral i.cs — an MC68681 dual asynchronous receiver/transmitter described last month and an optional MC68230 peripheral interface/timer. The 68230 provides digital i/o interfacing and a programmable timer, similar to the older 6821 p.i.a. or 6522 v.i.a. but much more powerful. The processor interface is similar to that of the duart, but there are no IRQ and IACK pins, and the clock for the timer is an external input which in Kaycomp is driven from the processor clock. There is also an extra register-select line RS_5 driven by address line A_5.

On the peripheral side 28 i/o pins are arranged in three ports of eight lines, PA_{0-7}, PB_{0-7}, PC_{0-7} and four handshake lines, H_{1-4}. All of these lines and the 5V supply are brought out to a 34-way connector, plug five, and are available to the user.

Some port C pins serve either as normal i/o lines or as timer/interrupt control pins, the function being selected under software control. Four of these function as the interrupt request output and acknowledge input for the timer and peripheral i/o sections of the device. Line PC_3 doubles as timer-output signal TOUT which can be used to generate an interrupt and PC_7 timer interrupt-acknowledge input TIACK. If interrupts are to be used on the i/o side, PC_5 is the peripheral interrupt request output PIRQ and PC_6 is the peripheral interrupt acknowledge signal PIACK. These four pins are not connected on the board to any interrupt pin or the IACK signal. If they are to be used, links can be made on P_6 as required.

There are certain points to be noted if using interrupts on the p.i.t. First, the IRQ outputs are not open drain but normal mos outputs and so cannot be wire-or'd with other interrupt sources, such as the duart. This means that the two p.i.t. interrupt sources TOUT and PIRQ cannot be used together on the same interrupt line. Also the p.i.t. requires that IACK only goes to the input that has caused the interrupt and not the other one, so only the one being used should be connected.

For the majority of Kaycomp applications these restrictions are not a problem It is of course possible to use one or both of interrupt lines IPL_0 and IPL_2 if they are not being used by the G64 bus. Connections between NMI/IRQ and $IPL_{2,0}$ of the processor are wire links instead of p.c.b. tracks to allow these links to be made across the board.

These restrictions on p.i.t. interrupts could not be tolerated on larger systems where it may be essential to have many devices connected to one interrupt level. This means extra arbitration logic, for instance open-collector buffers on the outputs so that they may be wire-or'd together, and gating to direct the IACK signal only to the interrupting source.

Rather than increase the complexity and cost of Kaycomp plug six, as well as being a patching area, has all relevant interrupt signals and the 5V supply brought to it. This location can be fitted with a 10-way header plug for connecting to a small separate board which implements the required arbitration logic to give full interrupt facilities. All the necessary signals are also available on plug five for convenience.

The duart IRQ output is directly connected to IPL_1 but there is a small link between two diagonally running parallel tracks just above plug five which if broken makes the duart IRQ and IPL_1 signals available separately at plug six.

Power

Requirements for the fully populated Kaycomp board are +5V at 0.7A, and positive and negative 12V rails at 30mA each. If the serial interfaces are not being used, the 12V supplies can be omitted.

Power is supplied through the G64-bus connector from the bus backplane. If the external bus is not being used then the connector could be omitted and supplies from an external power source wired directly to the board.

Circuit and component options

In its basic form, Kaycomp can be used with just the processor, ram, eprom and the serial ports, the processor clock being derived from the duart clock. A wire link is fed from the pad next to pin 32 of IC_{16} to pin 10 of IC_{14} for the clock signal. Components R_2, C_9, C_{10} and XL_2 are not fitted.

Having a processor clock of about 3.6MHz means that the slowest 68000 processor and 450ns-access or faster memories may be used. Gate IC_{14} may be either an l.s.t.t.l. type or a t.t.l- compatible high-speed c-mos device such as the 74HCT00. If a separate processor crystal is used or anticipated, IC_{14} must be a high-speed c-mos type.

I recommend that you use sockets for the larger i.cs like the processor, peripheral i.cs and memories. Remaining i.cs may be soldered in or socketed.

If your p.c.b. has plated-through holes then any type of good-quality socket may be used. If not there are a number of turned-pin socket types available that allow soldering on both sides of the board using a fine soldering iron tip. The easiest types to use are sock

rminal-carriers which consist individual sockets with no insulator mounted on an aluminium 'dummy' i.c. After soldering, the dummy i.c. is removed, leaving the individual pins in place to form an i.c. socket.

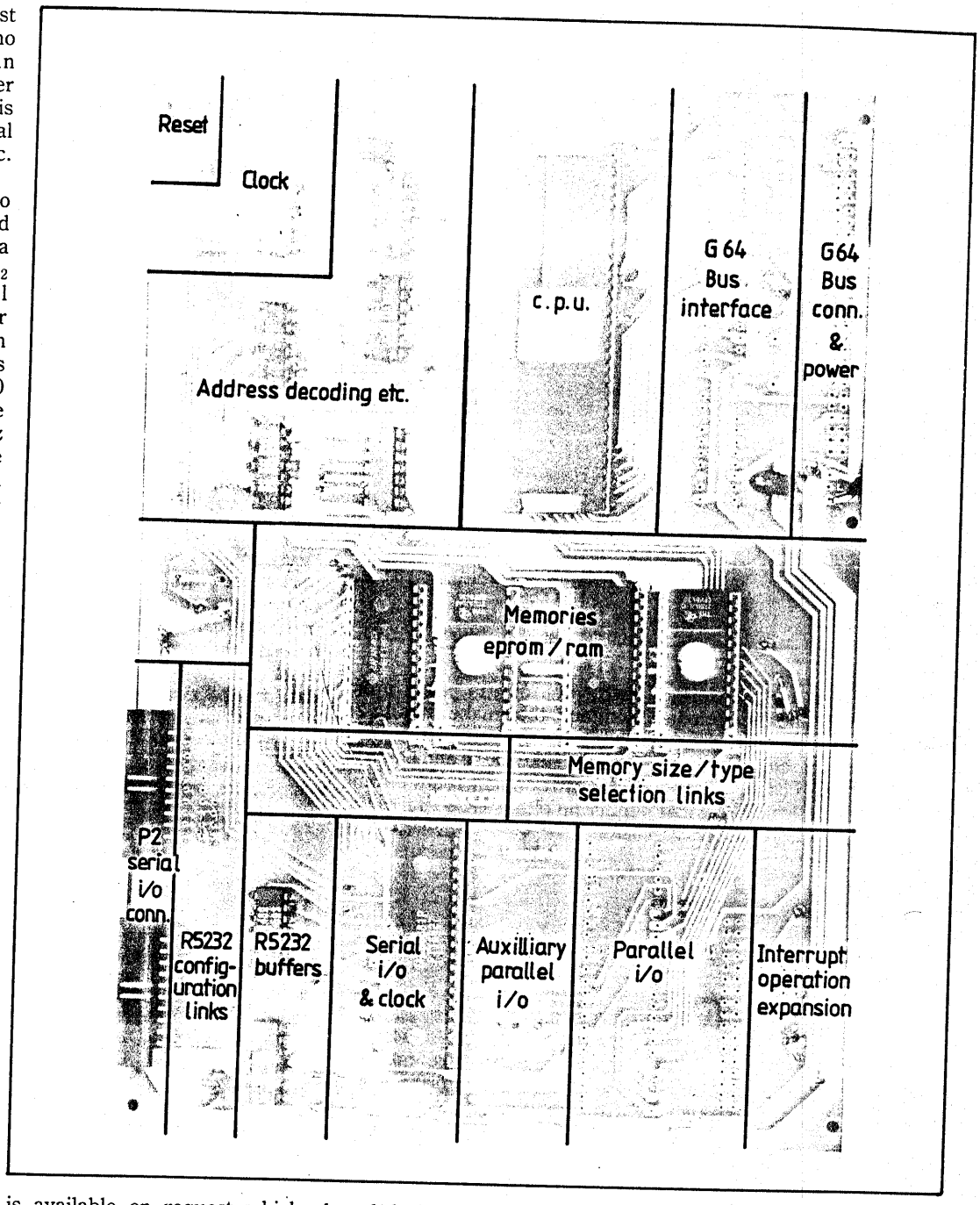
If crystal XL_2 is to be used to take full advantage of the speed of the board then IC_{14} must be a 74HCT00 and C_9, C_{10} and R_2 must be fitted. Crystal frequency is chosen to suit your needs and the speed selection of the processor and memories used. At present 68000 microprocessors are available with 4, 8, 10, 12.5 and 16MHz maximum clock rates. The board was designed to be used with an 8MHz crystal, but the clock oscillator will work down to 2MHz. In practice the boards have been found to work up to 10MHz satisfactorily.

The 68000 peripheral devices will work with any clock speed, as the asynchronous bus transfer technique means that access to these chips will be slowed down to suit them, but the memories must be capable of operating at the appropriate speed. Maximum access times of devices that may be used at various crystal frequencies in nanoseconds are 660 for 4MHz, 290 for 8MHz and 230 for 10MHz. C125 optimum

The 'Kaybug' monitor fits into two 2732 eproms, but these are most commonly available with 450ns access times. This though is the worst case and in practice, 450ns devices tried in prototype boards have always worked satisfactorily at 8MHz although this cannot be guaranteed, of course.

If peripheral i/o is required then IC_{15}, the MC68230, must be fitted. A socket should be used here, but 48-pin sockets are not very common so it may have to be built up using socket strips or by using two 24-pin sockets mounted end-to-end. A connector for the i/o lines will also be required, this being a 0.1in 34-way straight p.c. mounting plug.

Three i.cs, IC_{11-13}, resistor R_1 and the bus connector need fitting if use is to be made of the G64 bus. It is also worth noting that if the G64 bus interface is fitted it is possible to use the board without the two on-board serial ports, allowing IC_{16-18} and X_1 to be omitted. A special version of the Kaybug monitor is available on request which instead of using the on-board ports for terminal and host communication uses a G64 dual serial interface card instead, the Thomson EFS-SI01.



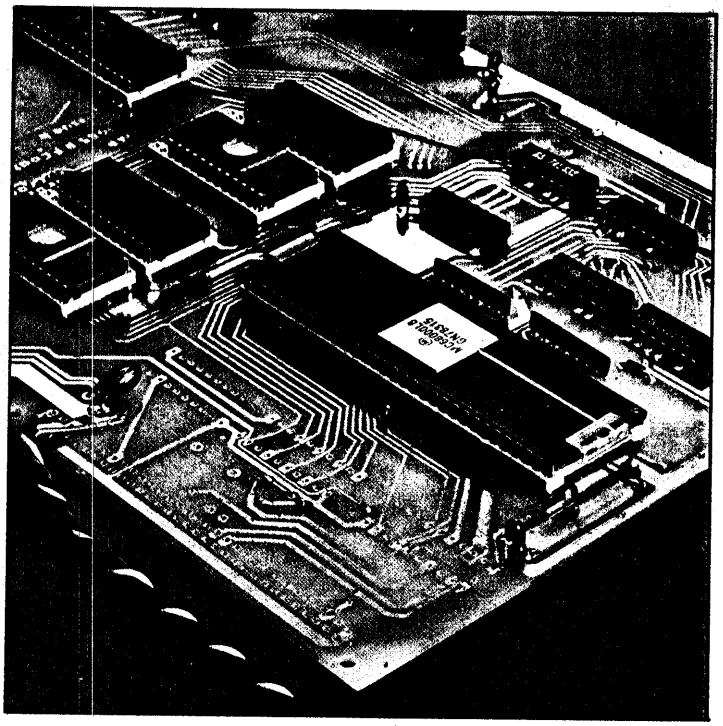
All resistor and resistor network values quoted are typical and are not critical, as are all capacitors with the exception of C_7, C_9 and C_{10}. Holes are provided at each corner for monting the board if required, or of course the board will fit a standard 6U-high Eurocard rack. In this case a Vero KM6 card front panel may be fitted using the two front mounting holes.

Construction is straightforward provided that the printed circuit board designed for the project is used. Using strip board or wire-wrap could prove a false economy and it may not be possible to achieve correct operation due to the high speeds involved on the board.

These construction details assume that you are building the fully populated board. After inspecting the bare p.c.b., there are three insulated cross-board links which may need to be fitted. If G64 NMI and IRQ lines are to be applied to IPL_2 and IPL_0, then fit wire links A-B and C-D respectively. These letter references are on the p.c.b. component position diagram supplied with the kit. Components IC_1 and IC_{12} fit over the top of the links to hold the wire in place. If a separate processor clock crystal is not needed, a wire link should connect point E next to IC_{16} to the holes for C_{10} nearest to IC_{14}.

Next, optional connectors PL_{1-3} and the i.c. sockets are fitted. The memory sockets should be fitted in the order IC_3, IC_5, IC_2 and IC_4. Sockets for $IC_{1,16}$ and optional IC_{15} are fitted in the same manner.

Now fit connectors P_{5-7}, as required. If you do not intend to change the memory types or the serial port configuration then wire jumpers can be used on links one to three. If changes are anticipated, fit wire-wrap pins instead. Resistors,



capacitors, resistor networks, D_1, PL_4 (reset) and the optional crystals come next.

Finally, gating chips $IC_{6\text{-}10,14}$, serial interface i.cs $IC_{17,18}$ and optional bus interface devices $IC_{11\text{-}13}$ should be soldered in, completing the board.

If power is derived from the G64 backplane, PL_1 is not fitted and the power leads are connected in the holes of PL_1 as follows; 0V to 32a and 32b, +5V to 31a and 31b, +12V to 30a (nearest board edge) and −12V to 30b.

Getting started

Before inserting the expensive i.cs, power up the board to make sure that there are no power-rail short circuits. Connections must be made on links $LK_{1\text{-}3}$. The first link needs to be configured according to the memory types being used, as shown on the circuit diagram in last month's article. The second and third links may be more of a problem. You will need to take into account requirements of the terminal to be used.

The Kaybug monitor program uses transmit data and receive data lines without handshaking, RTS output being permanently asserted, but the handshaking input and output lines are available. As a minimum requirement, only pins 1a and 3a may need to be connected. If the terminal receives data on pin three and transmits on pin two, which is the most common, then make links 1a-2b and 3a-1b. If it's the other way round, make 1a-1b and 3a-2b.

Many terminals require some input on either DCD, pin eight, or CTS, pin five. A suitable signal is available at pins 2a of the second and third links. It may require some trial and error to get things right. The most common configuration is 1a-2b, 2a-4b and 3a-1b.

Components $PL_{2,3}$ are 20-way 0.1in insulation-displacement connector plugs; 25-way D-type connectors are normally used for RS232 interfaces. Pin numbers of $PL_{2,3}$ though correspond to appropriate pin numbers on a D-type connector. If a terminal lead is made up using ribbon-cable connectors with a 20-way 0.1in insulation displacement connector at one end and a 25-way D-type i.d.c. (male or female to mate with the terminal) at the other, pin one on the 20-way connector coincides with pin one on the 25-way one.

Circuits $IC_{1\text{-}5,16,17}$ can now be fitted in their sockets. Make sure the upper and lower byte eproms are in their appropriate sockets.

If a Thomson EFS-S101 card is being used for serial communication instead of the on-board duart then this too will have to be configured to suit the terminal and the data rate set as described in the manufacturer's manual. Address switches on the EFS-SI01 card J3 and J4 should be set to F and E respectively so that the board occupies addresses FFFFF1 to FFFFF7.

Port P_1 is used for the terminal and P_2 for the host system if required. With both types of serial interface characters have eight data bits, no parity bit and one stop bit, which the terminal should also be set to.

At switch on, if the EFS-SI01 card is being used you should be greeted with a sign-on message and prompt on the terminal if all is well. With the stand-alone board though nothing will happen.

Kaycomp's serial interface has no data-rate selection hardware, this function being performed in software. At power-up (or reset) before anything can be sent to the screen, the monitor software must determine the terminal data rate. This is done by the operator hitting the carriage return key repeatedly.

The monitor reads the characters in, stepping through the available rates until the carriage return code is correctly interpreted. It then holds the data rate at that value and sends the sign-on message and prompt to the terminal. Data rates available with the keybug monitor are 19200, 9600, 4800, 2400, 1200, 600, 300, 150, 110 and 75 baud.

The second serial port on Kaycomp works identically to the first and may be used for connection to a host computer. Data rate for this port is set at reset to be the same as port A but it is possible to alter this by software if required. On the external serial card, data rates are set individually by jumpers and so it is possible to set the two ports differently.

Care should be taken if the host port is at a higher rate than the terminal as the host could send a stream of data faster than it could be transferred to the terminal. Although no hardware handshaking is supported by the monitor, it supports XON/XOFF protocol to suspend output which allows slow terminals such as those with smooth scrolling to be used at high rates.

Software for the 68000-board monitor is discussed in the next article.

Events

22 November
Marconi — triumph of a non-conformist: Royal Institution lecture, Royal Institution, Albermarle Street, London W1. 21h. Tel. 01-493 6470.
25 November
Propagation above and below the waves: IEE colloquium.
Improving prediction methods for radio services in u.h.f. and v.h.f., bands: IEE lecture.
26 November
The Janet project (networking in universities and colleges) IEE colloquium, 10h.
27-28 November
Measurements for telecommunications transmission systems. IEE/IMA/IoP/IERE International conference at the IEE Savoy Place.
27-29 November
ITAME: International test and measurement exhibition Olympia 2, London. Tel. 0280 815226 or 0799 26699.
29 November
Real-time expert systems for process control: IEE colloquium, University of Salford. 1030h.
UK research into robotics and automatic manufacture: IEE/IMechE/SERC seminar at Savoy Place.
3 December
Impact of i.k.b.s. on designers: IEE/Design Council discussion meeting, 14h.
3-4 December
Satellite communications: Conference at the Tara Hotel, London. Details from Online, Tel. 01-868 4466.
4 December
Electrostatic damage in semiconductor devices. IoP/IEE one-day meeting at Imperial College, London. Tel. 01-235 6111.
4-5 December
Electric and magnetic fields in medicine and biology: International conference at Savoy Place.
5 December
Automatic test technology for electronic devices and boards. IEE colloquim, 14h.
History of sound broadcasting: IEE lecture.
6 December
Energy sources for portable telecommunication equipment: IEE colloqium.
9 December
Cross polar cancellation techniques: IEE colloquium.
10 December
EMP protection: IEE colloquium, 1030h.
What is MAP? IEE lecture.
12 December
The application of real-time Basic and Forth to control systems: IEE colloqium.

IB 23/2/87

68000 board-4

by R.F. Coates

With the board now working and talking to the terminal Bob Coates turns to his Kaybug monitor software.

The monitor has been made as friendly as possible. It prompts the user for the next entry line where necessary and gives a 'help' listing which shows all the available commands with a brief description of each. Even the occasional user should not need to refer to the full description having once studied it.

A total of 38 different commands is available, each one invoked by a simple two-letter code. Any further information required is prompted for. The commands allow listing of memory blocks, altering memory, loading object code from host, setting breakpoints, altering register contents, running programs, single-step tracing through programs and so on.

The monitor accepts either upper or lower-case letters. Printing on the terminal screen is formatted to suit 80-column terminals but a command allows an alternative 40-column format.

There is a collection of system calls (sub-routines) which a user's program may call upon. Most are concerned with reading the keyboard and writing to the terminal screen, but others include b.c.d. to binary conversions and a 32-bit divide.

Unless otherwise stated, addresses and values entered are eight-digit hex numbers. The last eight are taken if more than eight are entered and leading zeroes are inserted if there are fewer then eight. Addresses must begin on word boundaries (even addresses).

The command set

An — *examine/alter address register:* the value in any of the target program's processor registers may be examined and altered if required. This command applies to the address registers a_0 to a_6 (a_7 is the stack pointer).

The command is invoked by keying the letter A followed by the appropriate number from 0 to 6. The current contents are then displayed (zero after a reset). The user is asked to enter either a new value followed by <cr> or just <cr> to leave the contents unaltered.

AE — *Ascii entry:* if text strings are to be included in your program, this command allows them to be entered more easily than is possible with the *MO* command.

It asks for the starting address and then waits for Ascii characters to be entered. It stores them as a string of bytes in memory. All Ascii characters and control codes are stored, apart from the break/hold characters and the following —

Backspace: does not store but backspaces the memory pointer and terminal cursor to allow corrections to be made.

Control-C: stores 'carriage return' with bit 7 set ($8D_{16}$).

Control-L: stores 'line feed' with bit 7 set ($8A_{16}$).

Control-N: stores null (00), the terminator required by *pdata1/pdatam*. Displays as \0.

The terminator control code set by *ST* is used to exit the *AE* command, the address following the end of the string being displayed.

AS — *line-by-line assembler:* This is an optional extension to the monitor.

It accepts assembler source code a line at a time, assembles it and stores the resulting object code in memory.

When this command is invoked, the display shows the default starting address which is 400400_{16}. If a different address is required, the new address may be entered in hex, preceded by a dollar sign.

Kaycomp specification

Processor	Motorola MC68000 or MC68010
Data bus	16 bits
Clock speed	Up to 10MHz
Eprom	Up to 128K-bytes
Ram	Up to 64K-bytes
Serial i/o	Two RS232 ports, 75 to 19200 bit/s
Parallel i/o	26 i/o; six input; six output
External bus	G64-bus (peripherals)
Dimensions	233.4mm × 160mm (double Eurocard)
Power	+5V at 0.7A, +12V and −12V at 30mA
Firmware	Comprehensive monitor Line-by-line assembler
Cost	Bare board: £18.90; with eproms and monitor, £34.40; with assembler too, £48.40; from Magenta Electronics, 135 Hunter Street, Burton-on-Trent, Staffordshire DE14 2ST. All prices include v.a.t. Postage costs 60p extra.

Each assembler source line should be terminated by a carriage return, e.g.

moveq #10,d3<cr>

After the terminating carriage return, the line is assembled, object code stored in memory and the line displayed as follows,

400400 760A moveq 10#,d3
400402

indicating the address, the object code produced and the original input line.

Lastly the address is incremented and displayed on the following line to await the next entry.

All standard Motorola instruction mnemonics are accepted as well as the five pseudos *$nnnnnn, data, ascii, asciz* and *end.*

$nnnnnn alters the working address to nnnnnn. The address entered must be even.

data allows bytes of data to be entered, e.g.

data 0,1,$10,32

will enter hex bytes 00, 01, 10, 20 into memory.

ascii allows Ascii text strings to be entered. The pseudo *ascii* must be followed by one space. All keys entered, up to the terminating <cr>, are then stored in memory as Ascii characters.

asciz: as *ascii* but also appends a null byte to the string, as required as terminator by *pdata1.* Note that if an odd number of bytes are entered with the *data, ascii* or *asciz* pseudos, then if an instruction mnemonic follows, its code will automatically be aligned on the next word boundary.

end terminates the assembler command and returns the monitor prompt. The <del> key will also abort the assembler, as with other commands.

Instructions for the 68000 can be specified as one of three sizes: byte (.b), word (.w) or long-word (.l). The required

size specification is appended to the mnemonic, e.g.

eor.b d3, (a6)

If no size is specified then 'word' is assumed.

Some instructions allow only one size. In such cases, no size may be specified.

Most of the addressing modes are as standard Motorola assembler format, but some variations should be noted (see box, right).

Since the assembler works line-by-line it has no way of resolving labels. Branch destinations must therefore be entered either as an absolute address,

bra $400426

or as a p.c. relative displacement,

bra *+26

(branch forward 26 bytes)

The displacement may be either short (8 bits) or long (16 bits) and is specified by appending .s or .l to the mnemonic. If no size is specified, .l is assumed:

bra.s $400426

(one-word instruction) or

bra.l $400426

(two-word instruction).

Addressing mode details

Indexed
The size subscript for the indexing register is optional, 'word' being the default.

Absolute word/long
This may be entered either as an absolute address or as a displacement from the current address. It must be suffixed with wither .w or .l to indicate absolute word or absolute long modes. So

400400 clr $400420.l and 400400 clr *$20.l

are equivalent instructions (*means current program counter contents).

P.c. relative, indexed
The format of these is similar to 'register indirect with displacement' and 'indexed', but with 'pc' replacing the register. The interpretation of the displacement is different though:

movea 10(pc), a6

means the displacement will be the relative offset between address 10 and the current address.

movea *+10(pc), a6

means the displacement will be the relative offset between the current address + 10 and the current address: in other words, 10.

In the case of the line ...

lea string, a6

(which loads a6 with the relative address of label 'string' at assembled address of $40043C$_{16}$, the line would be entered as lea $40043c(pc), a6

Bn — *breakpoint display/set:* up to four separate breakpoints may be set within a program, B1, B2, B3 and B4. If, say, B1 is entered, the terminal displays the address where this breakpoint is set. The user may either hit just <cr> to leave the breakpoint set at the current address; or else enter a new address where the breakpoint is to be set, removing any old breakpoint that may be set. If an address of zero is entered then no new breakpoint is set, just the old one removed.

CM — *compare two blocks of memory:* this command compares two blocks of memory and prints any differences between them. It asks for three addresses to be entered: the start address of the first block, the end address of the first block and the start address of the second block. It then compares the memory blocks byte-by-byte and prints a list of addresses (first block address given) where bytes do not match.

CN — *continue execution after break:* when a breakpoint is set at a particular address in ram with the *Bn* command, the original instruction at that address is saved and replaced by a Trap instruction with vector number 15 to 12 for B1 to B4 respectively.

When this instruction is executed the processor jumps back into the monitor, first displaying the current status of all the registers. If it is now desired to resume execution from this point, the *CN* command will re-start the program by re-inserting the original instruction at the breakpoint address and running that instruction. Thus the breakpoint is automatically cancelled by this command.

CV — *convert between hex and decimal:* this useful programming aid will convert numbers from hexadecimal to decimal and vice-versa.

To enter a number in hexadecimal form, prefix it with the letter X. The largest number that may be converted is 99999999 or its hexadecimal equivalent.

Dn — *examine/alter data registers:* this is exactly the same as *An*, but works on data registers d_0 to d_7.

FI — *fill block of memory with word.* The start and end address of the memory block are requested first, then the value of the data word (up to four hex digits). The memory block is then filled with this word.

No checks are performed to see whether the data has been stored correctly.

GO — *go and execute target program:* this is the command to start running your target program. It asks for the start address of your program, then sets all the processor registers to the target values (as shown by *RD*). It then loads the program counter with the entered address, transferring control to your program, in which it will remain until a breakpoint, say, returns it to the monitor.

HE — *help:* gives a brief description of all commands.

LH — *Load S file from the host into memory:* this is used to download an object-code file created by a host computer's assembler or compiler into the Kaycomp's ram area ready for running. The host system must be capable of outputting the object file in the standard Motorola S2 Ascii format.

The *LH* command is very similar to the *TM* command, the Kaycomp going 'transparent' and the terminal being able to talk to the host computer. In addition, though, it looks at the information coming back from the host for data in S file format, which contains Ascii encoded object code plus address and checksum information. If it finds this, it will decode it and store it at the designated address in memory.

If an error occurs during loading — if a byte is not stored correctly or if there is a checksum error — then the message

*** LOAD ERROR ***

will appear and the command will be aborted.

If the S file is loaded in correctly, the end-of-file marker will cause the command to exit without error. Alternatively, an exit from the command can be forced with <control-T> or whatever the terminator has been set to by the *ST*.

If a breakpoint is set in memory at a point where *LH* loads new data, then the breakpoint is cancelled.

The procedure then for using this command is first, key *LH*; key in the host system's command for dumping an object file in S format; then wait until the file is loaded (it is displayed as it is received).

LT — *Load S file from terminal:* this is the same as *LH* except that it receives the S format file from the terminal rather than the host port.

After this command has been entered, the computer being used as terminal must exit terminal mode and be made to download the S file.

LW — *locate data word in memory:* this will search for a particular data word in a memory block.

It asks for memory block start and end addresses and then for the four-digit hex data word to be located. It will then print a list of all addresses in the block where the data word is found.

MO — *memory open/modify:* this command allows you to examine and modify memory a word at a time.

It first asks for the address where you wish to start. It then displays that address and its current contents.

The user may choose to alter the contents by entering a new four-digit hex value.

The line should be terminated by one of the following keys:

(a) <space> will cause the new data word, if entered, to be stored in memory. The command will then step on to the next memory word.

(b) <^> (upward arrow on some terminals) causes data to be stored and then the previous word opened.

(c) <cr> causes the data to be stored and the command exited.

If a new data word is stored, a check is made to ensure that it is stored correctly (i.e. addressed to ram and not to eprom).

If it has not been stored,
*** NO CHANGE ***
is displayed and the command exited.

MV— *move memory block:* allows a block of memory to be moved up or down in memory by as small or large a shift as required.

It asks for the start and end address of the memory block to be moved and then the new start address it is to take. No checks are performed to see whether the bytes moved are correctly stored.

PR — *Print memory block:* prints on the screen the contents of the specified block of memory.

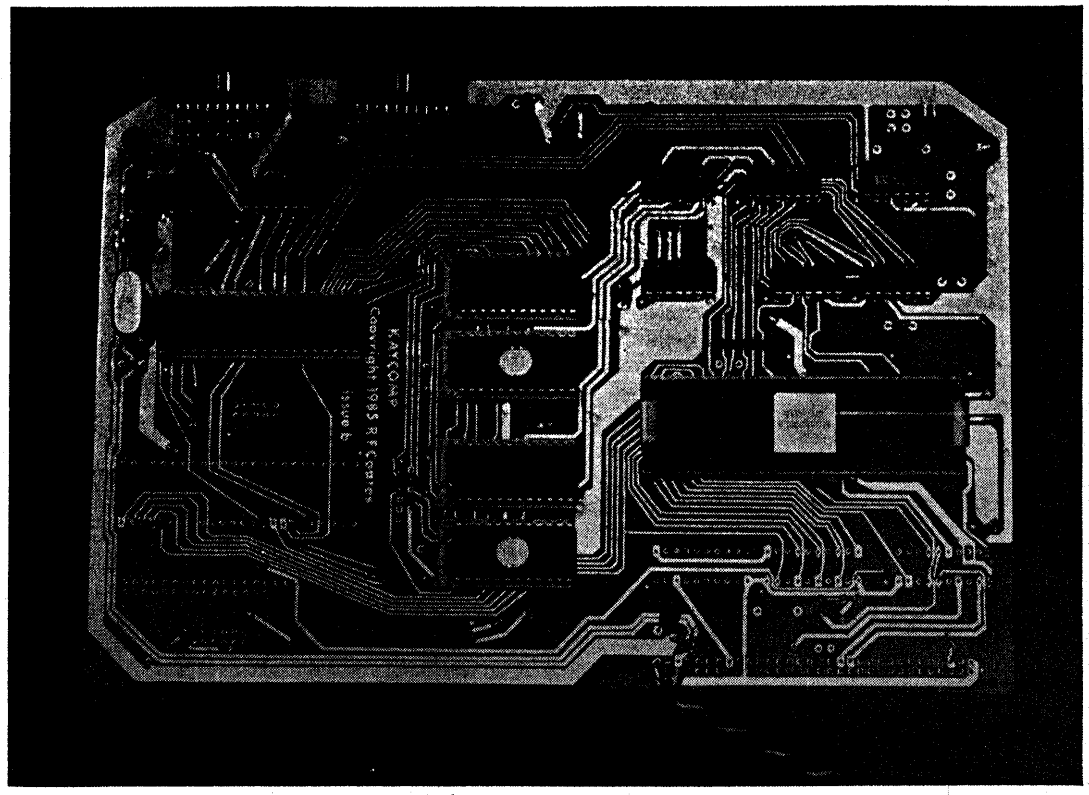
It asks for the start and end addresses of the data block (which need not be at word boundaries as the start address is rounded down to the start of a 16-byte block and the end address rounded up). In asking, it displays the value last used and re-uses it if you hit just <cr>.

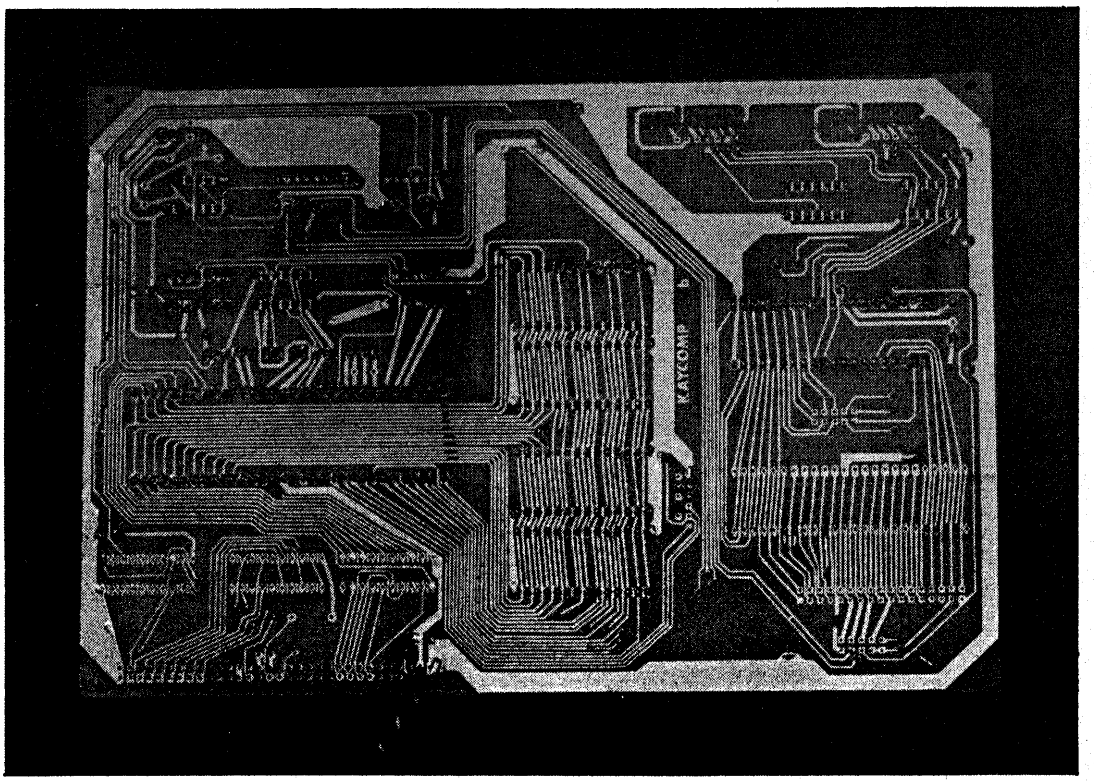
The data is formatted into word-sized groups. Eight words are displayed to a line in the 80-column mode (see *TC*), four words per line in 40-column mode. On the right of each line is displayed the Ascii equivalent character of each byte in the line (a dot if there is none).

Control-S and control-Q may be used to control the output, <del> to abort it.

RD — *Display target program's registers:* this command is called automatically when a breakpoint is encountered. It shows the contents of all the 68000's registers. The values displayed are those which the registers will contain if a program is started using *GO* or re-started using *CN*.

With the exception of the status register (SR), all are 32-bit registers and are displayed as eight-digit hex numbers. 'A7' refers to the supervisor stack pointer, as Kaycomp always operates in supervisor mode to allow the user unhindered access to all instructions except when a user's program switches the processor to the user mode.





The status register is only 16 bits in size. If it were displayed as four hex digits, some mental agility would be needed to discover the state of each individual bit. The status register is therefore displayed as 16 individual bits: a 1 indicating bit set, 0 bit clear. Each allocated bit is labelled by a legend above.

The program counter (PC) is the address where the breakpoint trap instruction was encountered.

SR — *examine/alter status register:* equivalent to *An* but operating on the status register.

It accepts and displays a four-digit hex value; so the particular bit pattern required must be converted to hex before being entered.

Correction: in the December article, the ninth line on page 38 should begin, "If power is *not* derived..."

ST — *set terminator.* The *LH, LT* and *TM* commands may, by default, be terminated by keying <control—T>. However, this particular character may have some function on the host computer you happen to be using and so there might be no way to send this code through from terminal to host.

To overcome this problem, the command *ST* allows the terminator to be changed to some other control code. This may be any Ascii control code (00 to $1F_{16}$) other than carriage-return ($0D_{16}$).

TC — *toggle column mode.* The displays associated with some commands assume that an 80-column terminal is being used and make full use of the screen width available.

This presents a problem where the terminal has fewer than 80 columns, for instance, a home computer with only 40 columns. But the command *TC* causes 40-column print formatting to be used in all subsequent operations. Entering *TC* again will switch back to 80-column mode.

TR — *trace target program:* when your program does not work and you cannot find where it is going wrong, it can be helpful to step through it one instruction at a time.

TR allows this by executing the target program and after each instruction displaying the register contents as with *RD*. It will continue this until a breakpoint is hit or control is passed back to the monitor, but control—S can be used to suspend execution and display.

TR asks first for the address where tracing is to be started; then for the address where the program is to be started from.

The two addresses may be the same, but are prompted for separately, so that the trace display may be started after code which is known to be working and where tracing is not required.

Note that if the addresses are different the processor still single-step traces from the very start; but the display being only when the trace start address is encountered. Up to that point, therefore, the program will run considerably faster as there is no display taking place, but it will not run at full speed.

TM — *transparent mode.* A host computer may be connected to Port B and used for storing programs on its storage medium and assembling them. This command makes a transparent connection between the terminal and host allowing the terminal to work the host directly.

Control-S, control-Q and <del> are not intercepted but passed on to the host. The only character not transferred is the terminator, normally control—T, unless altered by *ST*.

WB — *write bytes to a memory location.* The *MO* command works fine for altering memory but is inconvenient when you wish to talk to peripheral devices, such as the PI/T or a d-to-a converter. But *WB* allows bytes to be written to any address, odd or even. It does not advance the address neither does it read back the data to check that the data stored correctly. D-to-a converters, for instance, normally cannot be read, only written to: so checking would serve no purpose.

The command asks first for the address of the byte to be operated on, then waits on a new line for successive pairs of hex digits to be entered. Each byte is stored at the chosen address.

The command is exited by <cr> or <del>.

To be continued

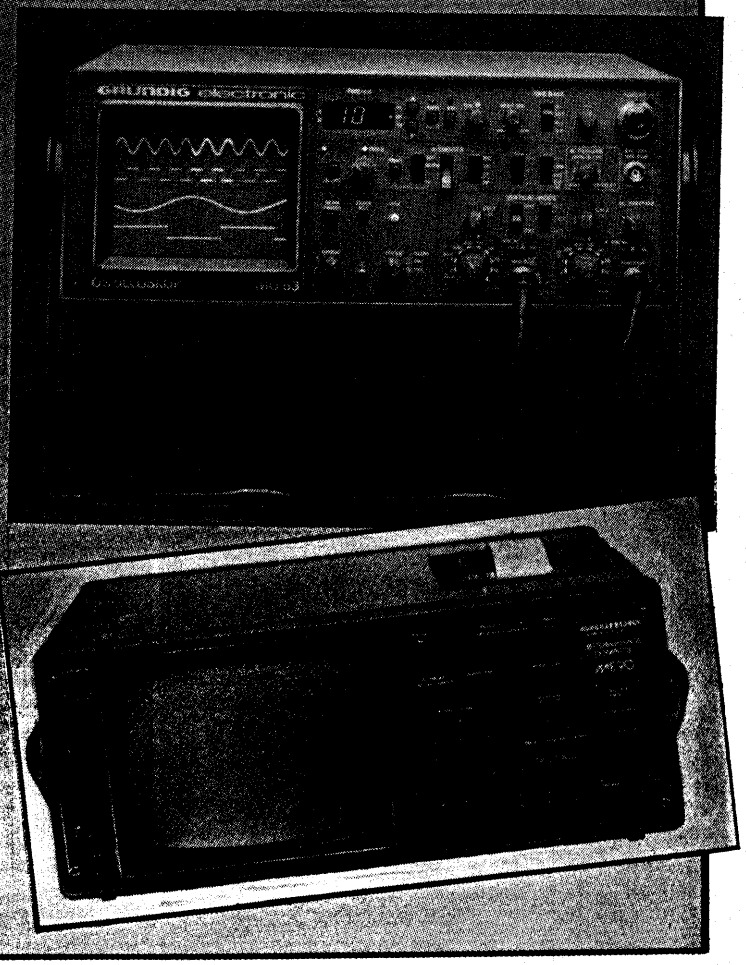
GRUNDIG AND ELECTRONIC BROKERS

Grundig AG, well known in the UK as a manufacturer of domestic electronic equipment, also possesses a reputation for test and measurement instruments. In a recent move to obtain better penetration of the UK market for t. and m., the company has appointed Electronic Brokers as exclusive UK distributors of, initially, eight instruments from the Grundig range. Instruments covered by the agreement include oscilloscopes, video pattern generators and an automatic field-strength meter.

Top of the oscilloscope range is the M053, a 50 MHz dual-channel instrument with what Grundig call a 'user-friendly' time-base control, which automatically sets the time-base speed to suit the signal and displays the time-setting digitally. It offers all the usual X and Y deflection facilities of an instrument in this class, with the difference that the delayed B time-base is separately triggered.

ME90 is a microprocessor-controlled test receiver/field-strength meter covering from long-wave radio frequencies to 862 MHz television. The receiver can store 100 channel frequencies, which are push-button selected, and will carry out automatic level measurement with error correction. Frequency, channel and level are indicated digitally and printed out with time and date. By programming the print operation, the instrument can be left unattended.

Grundig and the Philips company co-operate closely, Philips having a 31.6% holding in the German company. The general manager and financial controllers are from Philips, but report to the Grundig board. Grundig is still in competition with Philips.



Grundig instruments now handled by Electronic Brokers include the M053 oscilloscope and ME90 intelligent field-strength tester.

IB 23/2/87

by R.F. Coates

68000 board — 5

Bob Coates continues his description of the Kaybug monitor software with a look at its system calls

The monitor offers 26 system calls or subroutines for user programs to draw upon. These handle keyboard input and screen output functions but also include number base conversions and a 32-bit divide routine.

The calls are invoked by setting up the appropriate entry conditions and then executing a trap #11 instruction followed by a word-sized number indicating which call is being made.

All calls except *renter* when completed return to the user program and continue with the next instruction, as if a jsr instruction had been used.

Table 1 lists the calls available in version 1.0 of the monitor along with the number that follows the trap#11 instruction.

A more detailed description of each call is now given.

renter: this call terminates a users' program and transfers control back to the monitor, giving the monitor prompt.

Entry conditions: none.

Exit conditions: not applicable.

outch: outputs a character in d_0.b, then checks whether 'break' <del> or 'hold' <control—S> characters have been received. If 'break' received, does not return to caller but aborts user program and returns to monitor. If 'hold' received, waits until a different character is received before returning to caller.

Entry conditions: d_0.b = Ascii character to be output.

Exit conditions: all registers preserved.

sendch: outputs character in then returns to caller without checking for 'break' or 'hold'.

Entry conditions: d_0.b = Ascii character to be output.

Exit conditions d_0.b. undefined, other registers preserved.

outs: outputs a space character. Uses *outch* (see comments about 'break' and 'hold').

Entry conditions: none.

Exit conditions: all registers preserved.

spaces: outputs a number of space characters as indicated by d_0.w. Uses *outch*.

Entry conditions: d_0.w = number of spaces required, 1 to 65535.

Exit conditions: all registers preserved.

pdata1: outputs Ascii string pointed to by a_6 and terminated by a null byte (00). Sends all bits of each character unmodified. Uses *outch*.

Entry conditions: a_6 = address of start of string.

Exit conditions: d_0.b = 0; a_6 = address of string terminator + 1; all other registers preserved.

pdatam: As *pdata1*, but takes account of the display mode currently in use.

Characters with bit 7 clear, i.e. normal Ascii characters, are always output. But the outputting of characters with bit 7 set depends upon the display mode selected.

In 80-column mode, characters with bit 7 set are ignored and there is no output. In 40-column mode, characters have bit 7 stripped and are output as normal characters.

Entry conditions: a_6 = address of start of string.

Exit conditions: d_0.b = 0; a_6 = address of string terminator + 1; all other registers preserved.

query: outputs a '?' and sounds bell/buzzer/bleep. Uses *outch*.

Entry conditions: none.

Exit conditions: all registers preserved.

outhex: outputs contents of lowest four bits in d_0 as an Ascii hex character. Uses *outch*.

Entry conditions: d_0 = hex value.

Exit conditions: d_0 = undefined; other registers preserved.

out2hx: outputs contents of lowest eight bits in d_0 as two Ascii hex characters. Uses *outch*.

Entry conditions: d_0 = hex value.

Exit conditions: d_0 = undefined; other registers preserved.

out4hx: outputs contents of lowest 16 bits in d_0 as four Ascii hex characters. Uses *outch*.

Entry conditions: d_0 = hex value.

Exit conditions: d_0 = undefined; other registers preserved.

out8hx: outputs contents of d_0 as eight Ascii hex characters. Uses *outch*.

Entry conditions: d_0 = hex value.

Exit conditions: d_0 = undefined; other registers preserved.

crlf: outputs a carriage return and line feed. Uses *outch*.

Entry conditions: none.

Exit conditions: all registers preserved.

getch: waits for character to be received from input.

When character received, checks whether 'break' or 'hold': if 'break', does not return to caller but aborts user

Table 1

Call no.	Name	Function
0000	renter	Exit users' program and re-enter monitor
0001	outch	Output character with pause check
0002	sendch	Output character without pause check
0003	outs	Output a space
0004	spaces	Output 'd0.w' spaces
0005	pdata1	Output string
0006	pdatam	Output string according to column mode
0007	query	Print '?' and sound bell
0008	outhex	Convert 'd0' nibble to ASCII and print
0009	out2hx	Print two hex characters in 'd0'
000A	out4hx	Print four hex characters in 'd0'
000B	out8hx	Print eight hex characters in 'd0'
000C	crlf	Print carriage return and line feed
000D	getch	Get character with pause check
000E	readch	Get character without pause check
000F	gethex	Get hex character
0010	conhex	Convert character to hex
0011	gpch	Get and echo character
0012	get2	Get two ASCII characters
0013	hexin	Get string of hex characters in
0014	getadr	Get address
0015	range	Get start/end addresses allowing previous values
0016	strend	Get start and end addresses
0017	binbcd	Binary to bcd conversion
0018	bcdbin	Bcd to binary conversion
0019	divide	32-bit two's complement divide

program and returns to the monitor. If 'hold', waits for another character to be received (other than 'break' or 'hold') before returning to caller. Returns with Ascii character in d_0.b, bit 7 forced clear.

Entry conditions: none.

Exit conditions: d_0.b = 7-bit Ascii character; all other registers preserved.

readch: waits for character to be received from input and returns with it without checking for 'break' or 'hold'.

Entry conditions: none.

Exit conditions: d_0.b = 7-bit Ascii character; all other registers preserved.

gethex: gets Ascii character and converts it to hex, returning with hex value in d_0 and 'c' bit in status register clear. If character was not a hex character, i.e. not 0-9 or A-F, then returns instead with Ascii code and 'c' bit set.

Uses *getch*: see comments about 'break' and 'hold'.

Entry conditions: none.

Exit conditions: (hex character) d_0.b = hex equivalent of character, s.r. 'c' bit = 0; (non-hex character) d_0.b = 7-bit Ascii character, s.r. 'c' bit = 1. All other registers preserved.

conhex: as *gethex*, but does not get character from input. Enter instead with Ascii character in d_0.b.

Entry conditions: d_0.b = 7-bit Ascii character.

Exit conditions: (hex character) d_0.b = hex equivalent of character, sr 'c' bit = 0; (non-hex character) d_0.b = 7-bit Ascii character, sr 'c' bit = 1. All other registers preserved.

gpch: gets character from input and echoes to output. Returns with it in d_0.b; if lower case, converting it to upper case. Uses *getch*.

Entry conditions: none.

Exit conditions: d_0.b = 7-bit Ascii character (upper case); all other registers preserved.

get2: gets two Ascii characters. Calls *gpch* twice and assembles the two characters in d_0.w.

Entry conditions: none.

Exit conditions: d_0.w, b_{15}—b_8 = first Ascii character; b_7—b_0 = second Ascii character.

hexin: gets string of characters from keyboard, converting them to hex and building in d_1. Returns after a non-hex character with the last eight hex characters entered in d_1, leading zeroes being inserted if necessary. Register d_0 contains the non-hex terminating character and d_2 the number of hex digits entered. Uses *gpch*.

Entry conditions: none.

Exit conditions: d_0.b = non-hex terminating character; d_1.l = eight hex characters; d_2.l = number of hex characters entered; all other registers preserved.

getadr: prompts operator to enter an address. Returns with address in a_0 and number of characters entered in d_2. Rejects odd addresses. Uses *gpch*.

Entry conditions: none.

Exit conditions: a_0 = address entered; d_2 = number of characters entered.

range: prompts user to enter a beginning and end address, giving the option of re-using the last used values (as used by the *PR* command). If values entered, updates ram registers *begadr* and *endadr* with new values. If just <cr> entered, registers left unmodified.

For Kaybug version 1.0, *begadr* is at 400000_{16} and *endadr* at 400004_{16}.

Entry conditions: none.

Exit conditions: *begadr* and *endadr* updated if new values entered; all registers preserved.

strend: prompts user to enter block start and end addresses (as used by *LW* command). Returns with start address in a_2 and end address in a_1.

Entry conditions: none.

Exit conditions: a_1.l = end address; a_2.l = start address; all other registers preserved.

binbcd: converts a 27-bit binary number in d_0 to eight packed b.c.d. digits in d_1, with 'c' bit in status register clear.

If input value exceeds maximum allowed ($5F5E0FF_{16}$) then d_1 is undefined and the 'c' bit is set.

Entry conditions: d_0.l = 27-bit binary number.

Exit conditions (valid input): d_1 = 8-digit packed b.c.d. result, s.r. 'c' bit = 0; (invalid input) d_1 = undefined, s.r. 'c' bit = 1. All other registers preserved.

bcdbin: converts an 8-digit packed b.c.d. number in d_0 to binary equivalent in d_1.

Entry conditions: d_0.l = eight-digit packed b.c.d. number.

Exit conditions: d_1.l = binary result; all other registers preserved.

divide: integer-divides the number in d_0 by the number in d_1. Result is in d_4 with remainder in d_5.

All numbers are in 32-bit two's complement binary. Sign conventions used are:

dividend	divisor	result	remainder
+	+	+	+
+	−	−	−
−	+	−	−
−	−	+	−

Entry conditions: d_0 = dividend; d_1 = divisor.

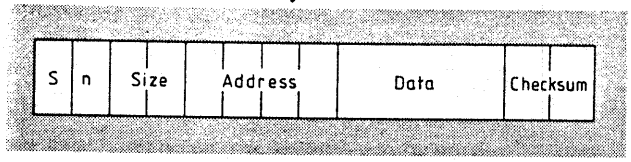
Exit conditions: d_4 = result; d_5 = remainder; all other registers preserved.

Exception handling

The way in which interrupt exceptions are handled by the 68000 is to go to a specified location according to the exception designation, fetch a long

Motorola S-format files

This format provides a means of Ascii-encoding binary object files for transfer between systems.



S is simply the Ascii code for the letter S.

n is the record type: 0 = header, 1 = data (16 bit address), 2 = data (24-bit address), 9 = end of file.

size indicates the number of bytes (hex) in this record (address, data and checksum).

address is four or six hex characters for record types 1 and 2 respectively. For types 0 and 9 it is 0000.

data is the block of ascii encoded binary data. Each record typically contains a 16-byte block, but may be more or less.

checksum is the one's complement of the sum of the size, address and data.

Each record is followed by carriage return and line feed. This example shows the S-file produced for the object code for example 1 (to be given in the next article).

```
S00900006578616D3120FA
S2144004007000720020 7C0000010052800601000A45
S20840041055886 0F670
S9030000FC
```

The header record holds the name of the object file as it was on the development system which produced this, 'EXAM1'.

Table 2

Assignment		Ram address	
Level	1	400016	Autovector interrupts
Level	2	40001C	
Level	3	400022	
Level	4	400028	
Level	5	40002E	
Level	6	400034	
Level	7	40003A	
Trap	#0	400040	Trap instructions
Trap	#1	400046	
Trap	#2	40004C	
Trap	#3	400052	
Trap	#4	400058	
Trap	#5	40005E	
Trap	#6	400064	
Trap	#7	40006A	
Trap	#8	400070	
Trap	#9	400076	
Trap	#10	40007C	
Vector	64	400082	User interrupts
Vector	65	400088	
Vector	66	40008E	
Vector	67	400094	
Vector	68	40009A	
Vector	69	4000A0	
Vector	70	4000A6	
Vector	71	4000AC	

word (four-byte) address from it and re-start processing from that address.

There is a slight problem with the Kaycomp in that these exception vector addresses are in the eprom space and so a user program cannot alter them.

To circumvent this difficulty, for each interrupt vector that may be used on the Kaycomp, a six-byte block of ram is allocated. The exception vector in eprom contains this address and so processing will jump to this location.

The six-byte space allows the user program to insert a 'jump absolute, long' instruction, the jump destination address being that of the user's exception service routine.

The exception processing sequence will thus be
1. Get vector and go to that address.
2. Execute jump instruction at that address.
3. Continue processing at jump destination address.

Ram 'jumpers' are provided for all autovector interrupts and eight user interrupt vectors are also catered for. These are for vector numbers from 64 (40_{16}) to 71 (47_{16}) and would be used by 68000 peripherals interrupting on IPL_1.

Trap instructions with vector numbers from 0 to 10 may also be used in user programs. These are handled in the same manner as interrupts. Trap numbers 11 to 15 are reserved for use by the monitor.

A complete list of these 'jumpers' is given in Table 2.

The remaining assigned system exceptions, such as Bus Error, Address Error etc, cannot be intercepted by the user program but can cause a return into the monitor with an appropriate message displayed.

To be concluded.

(May 1986)

The DMS8832 memory module for this project is available under the type number EDH8832C (with the suffix –15PC for the 150ns version) from MicroCall, Thame Park Road, Thame, Oxfordshire; tel. 084421-5405.

by Tom Ivall

D.B.S. — a plan in search of some users

As David Withers mentioned in the December issue (p.75) the most recent ITU pronouncement on direct broadcasting by satellites came at last year's World Administrative Radio Conference. This meeting, WARC 85, formally approved the d.b.s. plan previously worked out in detail at RARC 83, a regional conference specially convened in 1983 for ITU Region 2, which broadly means the Americas.

A similar situation occurred in 1979 when the big WARC 79 world conference formally agreed to the d.b.s. plan put together for ITU Region 1 (Europe, Africa, USSR) and Region 3 (Far East and Australasia) at WARC 77, which had been entirely devoted to satellite broadcasting. As a result we now have a complete world-wide plan — in frequencies, channels, powers, orbital positions etc. all signed and sealed and just waiting for someone to start using it for full broadcasting operation.

The Japanese came near to starting the first d.b.s. service in early 1984 when they launched a satellite from their Tanegashima Space Centre and placed it in geostationary orbit in their allocated orbital position of 110°E. But, as was reported in *E&WW*, faults rapidly developed in the transponders' travelling-wave tubes and the whole project had to be downgraded to an experimental status.

Meanwhile, we have seen the unexpected development of what has become known as 'semi-d.b.s.' or 'quasi-d.b.s.'. Transmissions from communications satellites, originally intended purely for distribution of television and sound programme feeds within the broadcasters' own networks, have become, willy-nilly, a sort of broadcasting in themselves. When cable tv systems use these programme feeds the signals are in any case 'broadcast' to a number of widely-spaced head-end receiving stations. And when reception of these transmissions by individuals was legalized, the comsats thereby became broadcasting stations in an even fuller sense, while still doing the job they were originally intended for. D.J. Standen's article in the December 1985 issue describe this 'quasi-d.b.s.' phenomenon in detail.

International planning for satellite systems has actually been under way for two decades or more. The ITU held its first conference on extra-terrestrial radio communication as long ago as 1963. This was when comsats were first beginning to establish themselves as practical relay stations. The second World Space Telecommunications Conference followed in 1971. Then came WARC 77 and