The 68000 and its interface

Alan Clements introduces the way in which a 68000 chip is interfaced to other system components

The 68000 represents the new generation of mature microprocessors. It is mature because it is powerful both in terms of its facilities and its computational throughput, and yet it is neither difficult to program nor to design systems around. This paper provides a simple introduction to the way in which a 68000 is interfaced to the other components of a microcomputer system. As the 68000 has so many facilities, only the basic details of its interfacing capabilities are provided.

microprocessors pin functions 68000

BASIC PIN FUNCTIONS OF THE 68000

The 68000 has 64 pins which may be grouped together as shown in Figure 1. There are nine logical groupings of pins: power supply and clock, address bus, data bus, asynchronous bus control, synchronous bus control, bus arbitration control, system control, function code, and interrupt control. Each of these groups will be dealt with in turn.

The current trend in microprocessor systems design is to use an asterisk to denote that a signal is active-low, so that what was once written as $\overline{\text{HALT}}$ is now written as HALT*. Furthermore, in the past a signal was often said to be 'forced high' or 'forced low' to effect a particular action. This meant that the reader had to remember whether the signal was active-high or active-low before he could figure out what was happening. Today, the term 'asserted' is used to indicate that a signal is put in a state which will cause its named action (eg HALT, RESET, STOP) to take place. Conversely, 'negated' means that the signal is placed in the opposite state in order to stop the named action. The reader does not have to know the actual physical state of an input or output when he is reading about the function of that line.

Each pin of the 68000 can be classified as an input, an output, or a dual-function input/output pin. When designing interfaces to the 68000, it is necessary to know the electrical characteristics of the processor's pins. Table 1 lists the pins of the 68000 and defines their electrical nature. A pin labelled I/O can act as an input or an output — but not at the same time. All outputs are labelled TS (tristate), TP (totempole) or OD (open drain) outputs.

Department of Computer Science, Teesside Polytechnic, Middlesbrough TS1 3BA, UK

Presented at *Applying the 68000 family*, City Conference Centre, London, UK, 30 October 1984

POWER SUPPLY AND CLOCK INPUT

In common with most other digital logic elements found in microprocessor systems, the 68000 requires a single +5V power supply. Two V_{CC} (ie +5V) pins and two ground (ie 0 V) pins are provided. This reduces the voltage drop between the V_{CC} terminals of the chip and the V_{CC} conductors within the chip itself.

The clock input is a single-phase TTL-compatible signal from which the 68000 derives all its internal timing. As the 68000 uses dynamic storage techniques internally, the clock input must never be stopped or its minimum or maximum pulse widths violated. Current versions of the 68000 have maximum clock rates between 4 MHz and 12.5 MHz. Basic read or write accesses require four clock cycles.

ADDRESS BUS

The address bus is provided by A_{01} to A_{23}, permitting 2^{23} 16-bit words to be uniquely addressed. The processor uses the address bus to specify the location of the word it is writing data into, or reading data from. Like several other processors, the 68000 treats all input/output transactions exactly like read/write operations, because it has no explicit input/output mechanism in either hardware or software. Because of its tristate outputs, the address bus can be controlled by a device other than the CPU under certain conditions. Whenever the 68000 is interrupted, it uses address lines A_{01}, A_{02} and A_{03} to indicate the level of the interrupt being serviced. During this so called acknowledge phase, address lines A_{04} to A_{23} are set to a logical one level.

0141-9331/84/07324-14 $03.00

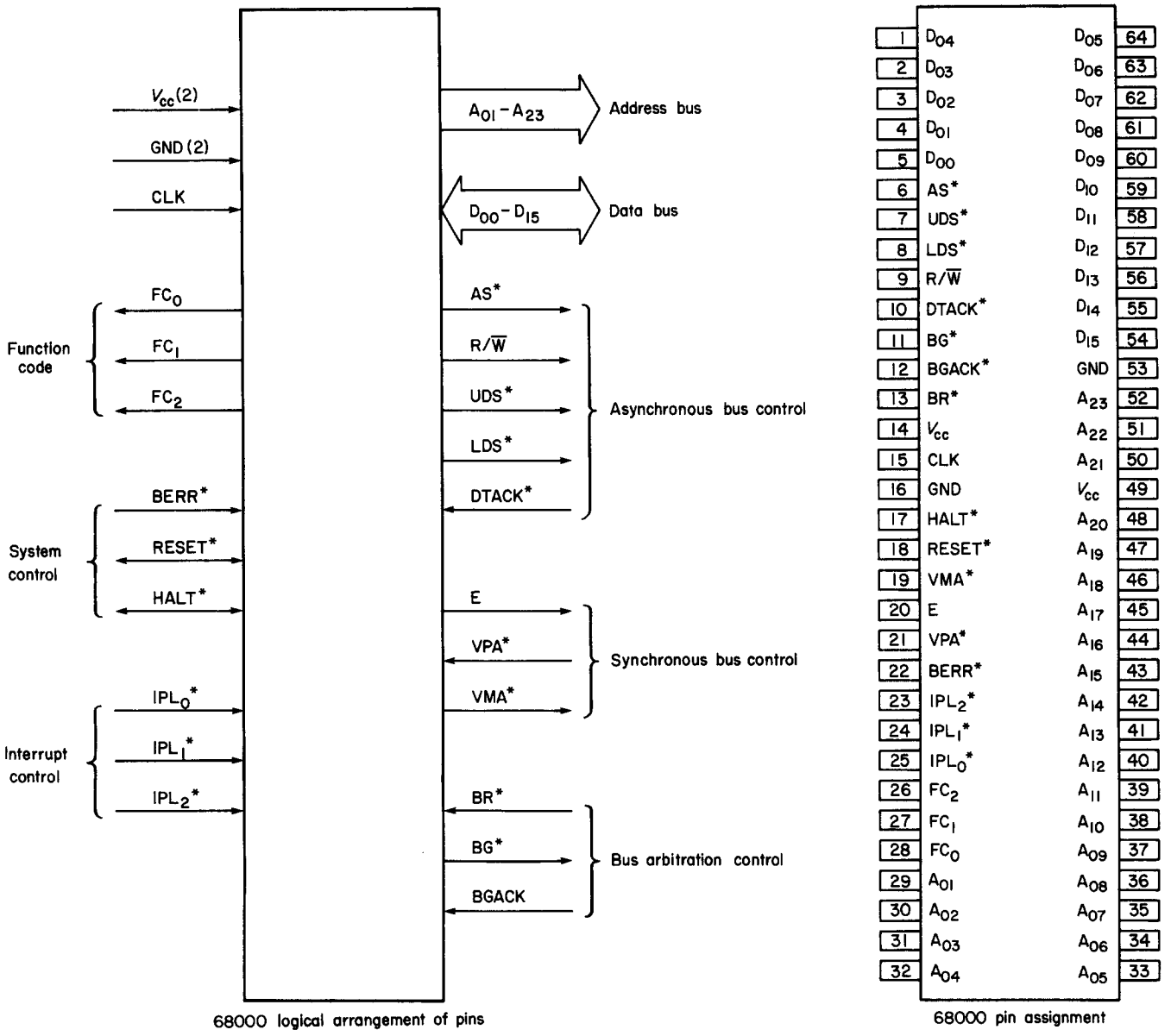

Figure 1. The pinout of the 68000

DATA BUS

The data bus is 16 bits wide and transfers data between the CPU and its memory and peripherals. It is bidirectional, acting as an input during a CPU read cycle and as an output during a CPU write cycle. The data bus has tristate outputs which can be floated to permit other devices to access the bus. When the CPU executes an operation on a word, all 16 data bus lines are active. When it executes an operation on a byte, only D_{00} to D_{07} or D_{08} to D_{15} are active. During an interrupt acknowledge cycle, the interrupting device identifies itself to the CPU by placing an interrupt vector number on D_{00} to D_{07}.

ASYNCHRONOUS BUS CONTROL

One important difference between the 68000 and many other microprocessors is the 68000's ability to carry out asynchronous data transfers between itself and memory or peripheral components. Asynchronous data transfers between the CPU and memory (or peripherals) are controlled by five signals: address strobe (AS*), upper and lower data strobes (UDS*, LDS*), read/write (R/$\overline{W}$), and data transfer acknowledge (DTACK*). In order to understand the nature of asynchronous data transfers, it is worth looking at synchronous data transfers first.

In a synchronous data transfer, the processor provides an address and some form of timing signal. Figure 2 demonstrates a simple synchronous data transfer — a CPU read from memory. At point *A*, a read cycle begins with the falling edge of the clock. At *B* the CPU generates an address corresponding to the memory location being accessed.

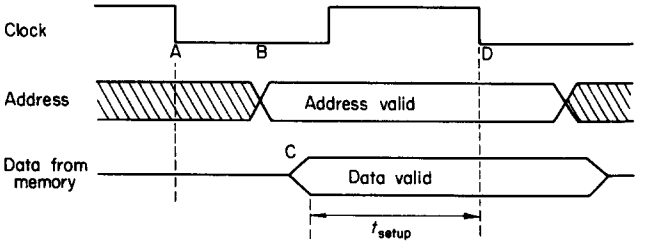

Figure 2. Synchronous data transfer

Table 1. The input/output characteristics of the 68000's pins; TS = tristate output, TD = totempole output, OD = open-drain output

Signal name	Mnemonic	Type	Output
Power supply	V_{CC}	Input	—
Ground	GND	Input	—
Clock	GLK	Input	—
Address bus	A_{01}–A_{23}	Output	TS
Data bus	D_{00}–D_{15}	I/O	TS
Address strobe	AS*	Output	TS
Read/write	R/W	Output	TS
Upper data strobe	UDS*	Output	TS
Lower data strobe	LDS*	Output	TS
Data transfer acknowledge	DTACK*	Input	—
Enable	E	Output	TP
Valid memory address	VMA*	Output	TS
Valid peripheral address	VPA*	Input	—
Bus request	BR*	Input	—
Bus grant	BG*	Output	TP
Bus grant acknowledge	BGACK*	Input	—
Bus error	BERR*	Input	—
Reset	RESET*	I/O	OD
Halt	HALT*	I/O	OD
Function code output	FC_0, FC_1, FC_2	Output	TS
Interrupt priority level	IPL_0*, IPL_1*, IPL_2*	Input	—

At C the memory yields its data for the CPU to read. At *D* the current cycle ends with the falling edge of the clock. The time between C and *D* is called the data setup time of the CPU and is the time for which the CPU demands that the data be valid before the end of the cycle. In this arrangement the clock must allow enough time for the memory to access its data. If sufficient time is not allowed and the setup time is violated, the data obtained by the CPU may be invalid.

An asynchronous data transfer is rather more complex as can be seen from Figure 3. At point *A* the processor generates a valid address. This leads to an address strobe being asserted at *B*. When the memory detects the address strobe, it places data on the data bus which becomes valid at point *C*. The memory then informs the processor that it has valid data by asserting a data acknowledge signal at point *D*. The processor detects that the data is now ready, reads it, and negates its address strobe to indicate that it has read the data (point *E*). The memory then negates its data acknowledge signal to complete the cycle. Below is a brief description of the asynchronous data transfer control signals of the 68000.

- AS*: The address strobe is active-low and indicates that the contents of the address bus are valid.

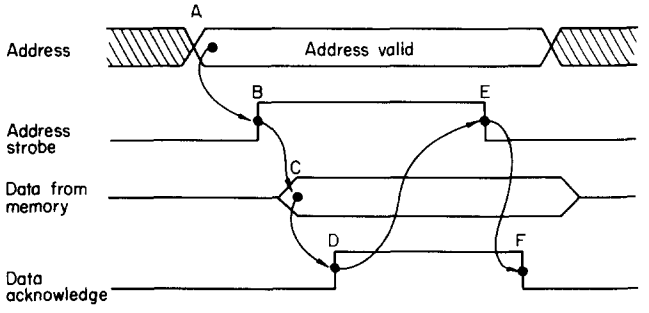

Figure 3. Asynchronous data transfer

- R/$\overline{W}$: The R/$\overline{W}$ (read/write) signal provided by the 68000 determines the nature of a memory access cycle. Whenever the CPU is reading from memory R/$\overline{W}$ = 1, and whenever it is writing to memory R/$\overline{W}$ = 0. If the CPU is performing an internal operation R/$\overline{W}$ is always true. That is, R/$\overline{W}$ is never in a logical zero state unless the CPU is executing a write to a memory location or a peripheral.
- UDS* and LDS*: The 68000 accesses memory via a 16-bit wide data bus. However, special provisions have to be made to enable it to access a byte of data instead of a word. When the 68000 accesses a word, both UDS* and LDS* are asserted simultaneously. If it wishes to access a single byte, UDS* is asserted if it is the upper byte (D_{08} to D_{15}), or LDS* if it is the lower byte (D_{00} to D_{07}). Table 2 defines the relationship between UDS*, LDS*, R/$\overline{W}$ and the data bus.
- DTACK*: The active-low data transfer acknowledge input to the 68000 is generated by the device being accessed and indicates that the contents of the data bus are valid, and that the 68000 may proceed. When the processor recognises that DTACK* has been asserted, it completes the current access and begins the next cycle. If DTACK* is not asserted, the processor generates wait-states until DTACK* is asserted, or until an error state is declared.

SYNCHRONOUS BUS CONTROL

The 68000 also has a built-in provision for synchronous transactions between itself and memory or peripherals. Strictly speaking, the synchronous bus control group of signals is not needed — all data transfers may take place asynchronously. The synchronous bus control group has been included entirely to simplify the interface between the 68000 and peripherals designed for

Table 2. The control of the data bus by UDS* and LDS*

R/$\overline{W}$	UDS*	LDS*	Operation	$D_{08} - D_{15}$	$D_{00} - D_{07}$
0	Negated	Negated	No operation	Invalid	Invalid
0	Negated	Asserted	Write lower byte	Note 1	Data valid
0	Asserted	Negated	Write upper byte	Data valid	Note 2
0	Asserted	Asserted	Write word	Data valid	Data valid
1	Negated	Negated	No operation	Invalid	Invalid
1	Negated	Asserted	Read lower byte	Invalid	Data valid
1	Asserted	Negated	Read upper byte	Data valid	Invalid
1	Asserted	Asserted	Read word	Data valid	Data valid

Notes 1 and 2: During a write to a byte the processor places a copy of the data being written onto both bytes of the data bus. Thus, if a byte is written to D_{00} to D_{07}, a copy of this byte is also placed on D_{08} to D_{15}. Motorola does not guarantee this feature on all future versions of the 68000.

use with the 6800, 6809 (or 6502) 8-bit synchronous-bus microprocessors. That is, this group of signals makes the 68000 look like a 6800 to certain types of peripheral. Three signals are included in this group

- VPA* (valid peripheral address)
- VMA* (valid memory address), and E (enable)
- VPA*: The active-low valid peripheral address input is used by a device to indicate to the 68000 that a synchronous peripheral is being accessed. When the processor recognises that VPA* has been asserted, it initiates a synchronous data transfer by means of VMA* and E.
- VMA*: This is an active-low output from the 68000 and indicates to the peripheral being addressed that there is a valid address on the address bus. The assertion of VMA* by the CPU is a response to the assertion of VPA* by an addressed peripheral.
- E: The enable output from the 68000 is a timing signal required by all 6800-series peripherals, and is derived from the 68000's own clock input. One E cycle is equal to ten 68000 clock cycles. The E clock is non-symmetric: it is low for six clock cycles and high for four. There is no defined phase relationship between the processor's own clock and the E clock. The E clock runs continuously, independently of the state of the 68000.

A synchronous data transfer is effected by detecting an access to a 6800-series peripheral and then asserting the processor's VPA* input. This must be done by user-supplied hardware. The 68000 then asserts VMA* and E which are used to select the peripheral.

BUS ARBITRATION CONTROL

When the 68000 has control of the system address and data buses it is said to be the bus master. Modern microcomputer systems include a mechanism whereby other microprocessors (or DMA controllers) can also take control of the system bus. The 68000 has three pins dedicated to bus arbitration control: bus request (BR*), bus grant (BG*), and bus grant acknowledge (BGACK*). 'Arbitration' is the term used to describe the sequence of events which take place when a number of potential masters request the bus simultaneously and one of them must be selected as the next bus master. The logic necessary to perform the arbitration does not form part of the 68000 and must be designed to suit the user's own application. Some 68000 users employ the processor's bus arbitration control signals to facilitate the design of dynamic memory refresh circuitry.

- BR*: All devices capable of being a bus master may drive the active-low bus request input with open-drain outputs. Whenever a device wishes to take control of the bus, it first asserts BR*, signalling its intent to the 68000.
- BG*: The 68000 asserts its active-low BG* (bus grant) output in response to the assertion of the BR* input. This indicates to the potential bus master that the current bus master is going to release control of the bus at the end of the current bus cycle.
- BGACK*: Bus grant acknowledge is an active-low input to the 68000 and indicates that some other device has now become the bus master. A potential master must not assert BGACK* until the following four conditions have been satisfied.
  - a bus grant has been issued by the current bus master
  - the address strobe, AS*, is inactive (ie negated) indicating that the microprocessor is not using the bus
  - data transfer acknowledge is inactive indicating that neither memory nor peripherals are using the bus
  - bus grant acknowledge is inactive indicating that no other device is still claiming bus mastery

In any system implementing a multimaster arrangement, some logic is necessary to arbitrate between competing bus masters. A 68000 microcomputer without other devices capable of taking the role of bus master does not need the bus arbitration control lines. Under these circumstances both BR* and BG* are permanently connected to a logical one level.

SYSTEM CONTROL

The 68000 has three active-low control inputs which are used to reset or halt the processor, or to indicate to the processor that a 'bus error' has occurred.

Whenever power is first applied to the 68000 (or any other microprocessor) it must execute some initialization process in order to start up in an orderly manner. It may also be reset while it is running. However, this action is taken only when a system crash has occurred and no other mechanism can be used to regain control of the processor. Although many 8-bit

microprocessors have a front-panel reset control, I feel that a sophisticated microprocessor such as the 68000 should be relatively difficult to reset by the user. An 8-bit microprocessor frequently has a simple, single-user, single-task operating system and resetting it causes no great harm. A more sophisticated 16-bit processor, on the other hand, may have a multiuser or multitasking operating system. In this case a manual reset by one user may cause untold harm to another user or task.

A microprocessor is said to be halted whenever it temporarily ceases to perform useful calculations. A microprocessor does not halt in the normal plain language sense of the word. It goes into an idle state (rather like an automobile in neutral) and relinquishes control of the bus. Some microcomputer systems use the halt state to allow other processors to control the system bus.

- RESET*: The active-low reset input of the 68000 forces it into a known state on the initial application of power. For correct operation during the power-up sequence, RESET* must be asserted together with the HALT* input for 100 ms — an incredibly long time by most microprocessor standards. At all other times, RESET* and HALT* must be asserted for ten clock periods. When a reset input is recognised by the 68000, it loads the system stack pointer, A_7, from memory location zero ($00 0000), and then loads the program counter from address $00 0004.

RESET* can also act as an output from the 68000 under certain circumstances. Whenever the processor executes the software instruction RESET, it asserts the RESET* pin for 124 clock cycles. This resets all external devices (ie peripherals) wired to the system RESET* line, but does not affect the internal operation of the 68000.

- BERR*:The active-low bus error input is used by the microcomputer system to inform the 68000 that something has gone wrong with the bus cycle currently being executed. It may be argued that this feature is one of the attributes distinguishing the 68000 from all 8-bit microprocessors and some 16-bit microprocessors. The provision of a BERR* input permits the 68000 to recover gracefully from events that would spell disaster to other processors.

Sometimes an access is made to a memory location which is either faulty or nonexistent. The latter case may occur when a spurious address is generated due to a software error, or it may be that the actual memory in the system is less than the operating system 'thinks'.

Whenever external logic detects such an anomaly, it asserts BERR*. The precise nature of the action taken by the 68000 on recognising that BERR* has been asserted is rather complex and is also dependent on the current state of the HALT* input. The 68000 will either try to repeat (ie rerun) the faulty cycle, or will generate an exception and inform the operating system of the bus error.

- HALT*: Like the RESET* input, HALT* is bidirectional and serves two distinct functions. In normal operation HALT* is an active-low input to the 68000. When asserted by an external device, HALT* causes the 68000 to stop processing at the end of the current instruction. Then all control signals are made inactive and all tristate outputs floated.

The recommended use of the HALT* input is to permit the 68000 to execute a single instruction at a time. If HALT* is negated, the 68000 will recommence normal operation. However, if HALT* is asserted early in the first cycle of the instruction sequence, the processor will be forced into a halt state after a single instruction has been executed. By negating HALT* just long enough to permit the processor to execute a single instruction, the 68000 can be stepped through a program instruction by instruction. This can be used to debug a system.

Whenever the 68000 finds itself in a situation from which it cannot recover (the so-called double bus error), it stops and asserts HALT* to indicate what has happened.

FUNCTION CODE

In principle a microprocessor simple reads instructions from memory, interprets them, and operates on data either within the processor itself or within the memory system. In practice the operation of the processor is rather more complex because it may have to interact with external events through the interrupt mechanism. Moreover, the processor accesses different types of information in memory: instructions, data, the stack etc. There are many occasions when it would be helpful to know what the computer was up to.

This information is called function or status information, and is provided by microprocessors (directly or indirectly) in varying amounts. For example, the Intel 8080A multiplexes status information on its data bus for a part of a cycle. The 68000 has three processor status outputs, FC_0, FC_1, and FC_2, which indicate the type of cycle currently being executed. The function code becomes valid at the same time as the address strobe (AS*) indicates a valid address. As a matter of fact, it is not always necessary to use the function code provided by the 68000 to build a working microcomputer. Equally, the function code can be used to enhance the operation of the system. Table 3 shows how FC_0, FC_1, and FC_2 are interpreted.

Of the eight states in Table 3, three are marked 'undefined, reserved'. This is Motorola's way of telling us that these states may be reassigned in future versions of the 68000. Function code output FC_2 distinguishes between two modes of operation of the 68000: supervisor and user.

Table 3. Interpreting the 68000's function code output

Function code output			Processor cycle type
FC_2	FC_1	FC_0	
0	0	0	(Undefined, reserved)
0	0	1	User data
0	1	0	User program
0	1	1	(Undefined, reserved)
1	0	0	(Undefined, reserved)
1	0	1	Supervisor data
1	1	0	Supervisor program
1	1	1	Interrupt acknowledge

It can be seen from Table 3 that the 68000 is always in one of two states: user or supervisor. The concept of user and supervisor states does not exist for 8-bit microprocessors or for some 16-bit devices. User and supervisor states have a meaning only in the world of multitasking systems, where a number of different programs are running concurrently. The supervisor state is said to be the state of highest privilege, and certain instructions may be executed only in this state. In general, the supervisor state is closely associated with the operating system, while the less privileged user state is associated with user programs running under the operating system.

By restricting the privileges available to user state, individual programs are capable of causing less havoc if they crash. The supervisor state is in force when the S-bit of the processor status word is true. All exception (interrupt and reset) processing is performed in the supervisor state, regardless of the state of the processor before the exception occurred. Consequently, the 68000 always powers up in the supervisor state. A change from supervisor to user state can be carried out under program control, but it is impossible to move from the user to supervisor state by any sequence of instructions. Only by the generation of an exception can a transfer from user to supervisor mode by made.

Table 3 also shows how it is possible to determine whether the processor is accessing program or data. The region of memory containing data is called 'data space' and the region containing instructions 'program space'. The meaning of the word 'space' in this context is closer to the mathematician's use of the word (eg vector space) than to the everyday meaning.

The advantage of dividing memory space into program and data spaces is that it becomes possible to prevent a program from corrupting the data space of another program by detecting any access to program space which would corrupt the program.

The function code denoted by $FC_0 = FC_1 = FC_2 = 1$ is called interrupt acknowledge, and is used as an indication that the 68000 is currently acknowledging an interrupt.

INTERRUPT CONTROL

Three interrupt control inputs (IPL_0*, IPL_1*, IPL_2*) are used by an external device to indicate to the 68000 that it requires service. These interrupts are encoded into eight levels (0 to 7). Level zero has the lowest priority and indicates that no interrupt is requested. Level seven is the highest priority interrupt. The status register contains three bits, I_2, I_1, and I_0, called the interrupt mask, which determine the level of interrupt that will be serviced.

An interrupt request indicated by a 3-bit code on IPL_0*, IPL_1*, IPL_2* will be serviced if it has a higher value than that currently indicated by the interrupt mask bits in the status register. A level-7 interrupt is handled rather differently because it is always serviced by the 68000.

Many peripherals capable of generating an interrupt have only a single interrupt request output. Consequently, most 68000-based microcomputer systems must use a priority encoder circuit to convert up to seven levels of interrupt request into a 3-bit code, which can then be fed into IPL_0* to IPL_2*.

THE TIMING DIAGRAM

The timing diagram represents the most fundamental transactions between a processor and its external environment. Traditionally, the timing diagram has been used to illustrate the detailed operation of a microprocessor or a memory component. A timing diagram shows the relationship between the signals involved in a read/write cycle and time. Although the timing diagram is an educational tool, because it presents visually the relationship between a number of signals, it is principally a design tool. It enables an engineer to match components of different characteristics so that they will work together.

In recent years the timing diagram has been supplemented by what may best be called a 'protocol diagram' or 'timing flowchart'. The protocol diagram is an abstraction of the timing diagram which seeks to remove all detail in order to provide only the most essential information to the reader. The read and write cycles of the 68000 will be explained in terms both of protocol flowcharts and timing diagrams.

THE 68000 READ CYCLE

This section considers the sequence of events taking place when the 68000 reads a word from memory using its address and data buses in conjunction with the asynchronous group of bus control signals. The 68000 can read either a 16-bit word or an 8-bit word in a single read cycle. As there is very little difference between these operations, only a word operation is described.

Figure 4 gives the protocol flowchart for a 68000 read cycle. Any read cycle involves two parties: the reader and the read. The reader is the 68000, and is represented by the bus master in Figure 4. A bus master is the active device that is currently controlling the system bus and at any instant there may be only one bus master. There may be several 68000s in a system, but only one may be the master at a time. Equally, a device other than a CPU may simulate a 68000 to gain control of the bus. The lefthand side of the diagram displays the actions carried out by the master (the 68000). Each block is labelled by the words in its top line. The numbered lines below the header describe the sequence of actions carried out by that block.

The righthand side of the diagram displays the actions carried out by the slave during the transfer of information. The slave is, of course, the memory being accessed by the master. The protocol diagram is read from top to bottom so that the action 'Address the slave', carried out by the master, is followed by the slave with the action 'Input the data'. Note that actions within boxes may, or may not, take place simultaneously.

What is lacking from this diagram are precise timing relationships, and details of critical events. For example, in the block labelled 'output the data', it is the action of asserting DTACK* which allows the master to continue with the action 'Acquire the data'. This is not evident from Figure 4, and therefore the diagram does not tell the whole story.

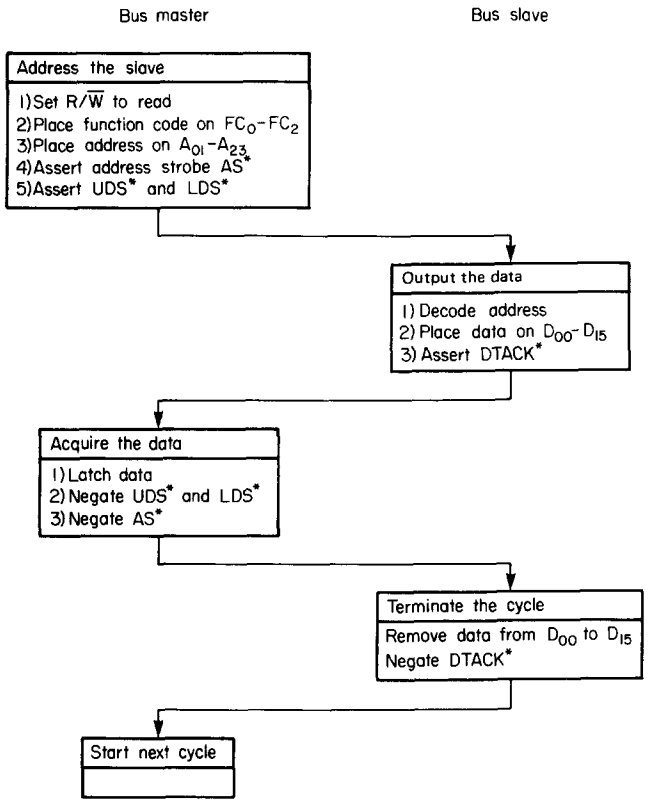

Figure 4. Protocol flowchart for a read cycle

The essential feature of a 68000 asynchronous read cycle is the interlocked handshaking procedure taking place between the master and the slave. A read cycle starts with the master indicating its intentions by setting up an address and forcing R/$\overline{W}$ true. By asserting AS*, UDS* and/or LDS*, the CPU is saying, 'Here's an address from which I wish to read the data'. The slave detects the valid address strobe (AS*) together with the data strobe(s), and starts to access the data. It asserts DTACK*, informing the processor that it may proceed. DTACK* is the handshake from the slave to the processor, and acknowledges that the slave has (or is about to have) valid data available. The microprocessor systems designer must provide suitable circuitry to generate the appropriate delay beween the start of a read (or write) cycle and the assertion of DTACK*. If DTACK* is not asserted, the master will theoretically wait forever. The 68000 has provision for dealing with the failure of a slave to complete a handshake by asserting DTACK*. When the master recognises DTACK*, it terminates the cycle by negating the address and data strobes. This invites the slave to terminate its actions by removing data from the bus and negating DTACK*.

A highly simplified version of a 68000 read cycle is presented in Figure 5. Each machine cycle consists of a minimum of four clock cycles, and is divided into eight states labelled S_0 to S_7. All machine cycles start in state S_0 with the clock high, and end in state S_7 with the clock low. The machine read cycle may be extended indefinitely by the insertion of wait states (each of one full clock cycle duration) between clock states S_4 and S_5. This allows the 68000 to be operated with any mixture of fast and slow memory or peripherals.

Figure 5 is designed to show the relationship between the 68000's asynchronous bus signals, and between these signals and the states of the clock. During the first state, S_0, all signals are inactive with the exception of R/$\overline{W}$, which becomes true (ie read) for the remainder of the current machine cycle. In the following description of the 68000, all times given are for the 8 MHz version, unless stated otherwise.

In state S_1 the address on A_{01} to A_{23} becomes valid and remains so until state S_0 of the following cycle. In state S_2 the address strobe, AS*, goes low, indicating that the contents of the address bus are valid. At this point it is tempting to ask why we need AS*, as the falling edge of S_2 can be used to indicate that the address is valid. The answer to this question lies in the variations between different versions of the 68000. In the 12.5 MHz version, it is possible that AS* will not go low until state S_3. It is not the relationship between the clock and the 68000's signals that matters to the designer. It is the relationship between the signals themselves.

In a read cycle, the timing specifications of the upper and lower data strobes (UDS* and LDS*) are the same as AS*. The falling edge of UDS* and/or LDS* initiates the memory access and at the same time, or after a suitable delay, triggers a data transfer acknowledge, DTACK*. Remember that it is up to the designer of the microcomputer system to provide logic to control DTACK*. The delay between a data strobe going low and the falling edge of DTACK* must be sufficient to guarantee that there is enough time to access the memory currently being accessed. If DTACK* does not go low at least 20 ns before the end of state S_4, wait states are introduced between S_4 and S_5 until DTACK* is asserted.

The assertion of the data strobe causes memory to be accessed, and data to appear on the data bus. In Figure 5 this happens in state S_5, although the actual time depends on the access time of the memory being accessed.

During the final state of the current machine cycle, S_7, both AS* and LDS*/UDS* are negated, and the data latched into the deep 68000 internally. The negation of these strobes causes the memory to stop putting data on the data bus, and to return the bus to its high impedance (floating) state. DTACK* must be negated after the strobes have been negated.

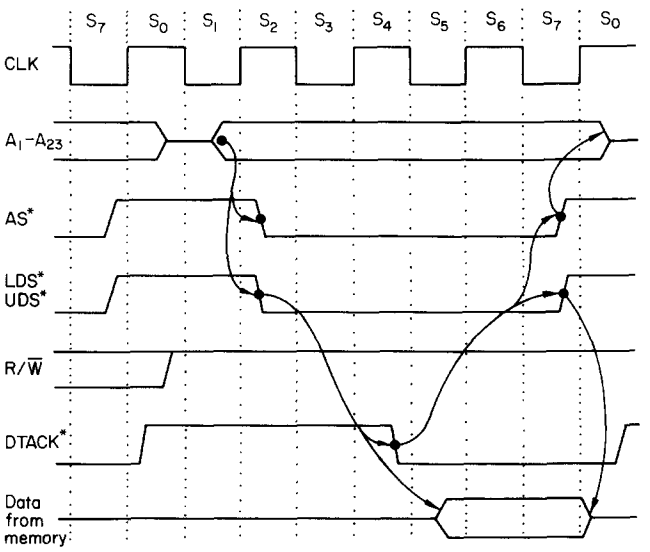

Figure 5. A simplified version of the 68000 read cycle timing diagram

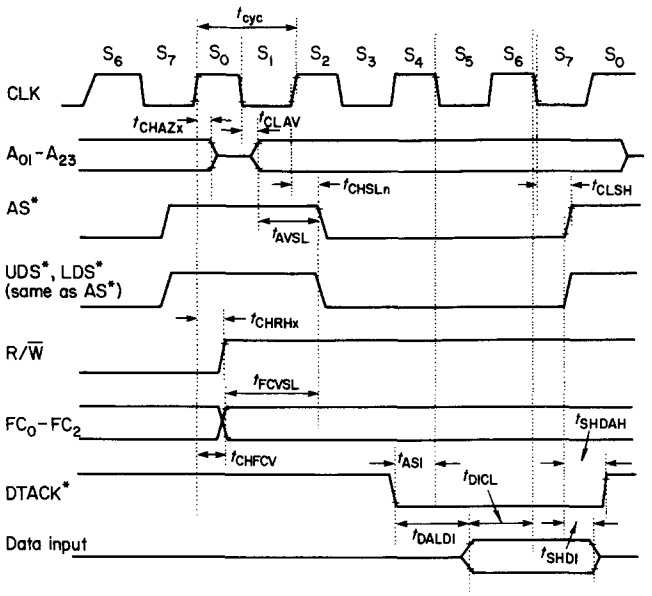

Figure 6. A more detailed version of the 68000 read cycle timing diagram

The address bus is floated in the following S_0 state and the read cycle is now complete.

The microcomputer designer needs to know the restrictions placed on his design by the timing diagram of a microprocessor. Figure 6 provides a more detailed read cycle timing diagram of the 68000. Table 4 gives the value of some of the read cycle timing parameters for the 68000L8.

The 68000 clock input is specified by three parameters:

- its period, t_{cyc}, must not be less than 125 ns (for a 8 MHz clock) or more than 500 ns
- the maximum limit is determined by the way in which the 68000 stores data internally as a charge on a capacitor
- if the 68000 is not clocked regularly, internal data is lost, leading to unpredictable behaviour of the processor

Limits are also placed on the times for which the clock may be in either a high or a low state. Table 4 reveals that the clock input should have an approximately symmetrical waveform with equal up and down times.

The address bus is floated within t_{CHAZx} s (80 ns max) of the start of S_0. No more than t_{CLAV} s (70 ns max) from the start of S_1, the new address is placed on the address bus. The address strobe, AS*, is asserted no less than t_{AVSL} s (30 ns min) after the address has stabilized. This is a key parameter, because if the designer uses AS* to latch the address, he must choose a device with a setup time less than t_{AVSL}.

R/$\overline{W}$ is set high at the beginning of a read cycle no more than t_{CHRHx} s (70 ns max) after the start of state S_0, and stays high for the remainder of the current cycle. In practice, this means that the designer can forget about R/$\overline{W}$ during a read cycle, as it is true well before the other parameters are valid and remains true until well after they have changed.

The 68000 puts out its function code no more than t_{CHFCV} s (70 ns max) after the start of state S_0, and no sooner than t_{FCVSL} s (60 ns min) before AS* is asserted. Consequently, the function code behaves like an address, and can be latched by AS* at the same time as an address.

The key parameter governing DTACK* is its setup time, t_{ASI} (20 ns min) before the falling edge of state S_4. If DTACK* is asserted before its minimum setup time, the next state will be S_5. If DTACK* does not meet this setup time, the processor introduces wait states after S_4, until DTACK* is asserted at least t_{ASI}s before the falling edge of the next 68000 clock input.

The data from the memory being accessed is placed on the data bus and must satisfy setup and hold times similar to the input of any D flip-flop. The data must be valid at least t_{DICL} s (15 ns min) before the beginning of state S_7.

CONNECTING THE HM6116P RAM TO A 68000 CPU

As an example of how the 68000 read cycle parameters

Table 4. Basic read/write cycle timing parameters (ns) of the 68000L8; DS = UDS* or LDS*

Parameter name	Symbol	Min	Max
Clock period	t_{cyc}	125	500
Clock width (low)	t_{CL}	55	250
Clock width (high)	t_{CH}	55	250
Clock high to address bus high-impedance	t_{CHAZx}		80
Clock low to address valid	t_{CLAV}		70
Address valid to AS* valid	t_{AVSL}	30	
Clock low to AS*, DS* high	t_{CLSH}		70
Clock high to R/$\overline{W}$ high	t_{CHRHx}		70
Clock high to FC valid	t_{CHFCV}		70
FC valid to AS*, DS* low	t_{FCVSL}	60	
Asynchronous input DTACK* setup time	t_{ASI}	20	
AS*, DS* high to DTACK* high	t_{SHDAH}	0	245
Data in to clock low setup time	t_{DICL}	15	
DS* high to data invalid (data hold time)	t_{SHDI}	0	

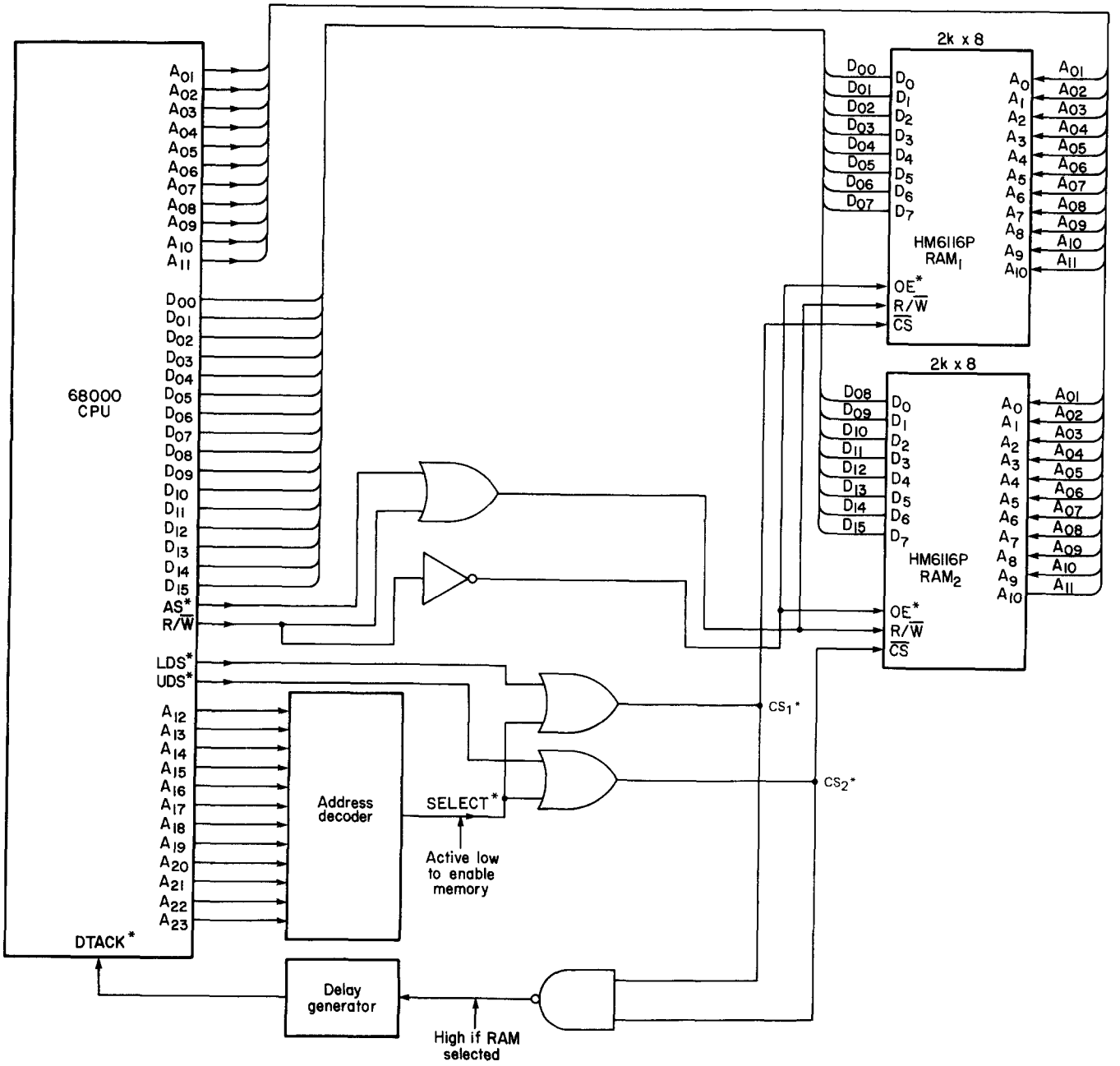

Figure 7. Connecting the HM6116P-4 RAM to a 68000 CPU

affect its operation, consider the interface between the 68000 and a typical static RAM. Figure 7 shows how two HM6116P 2k × 8 RAMs can be connected to a 68000 CPU. This circuit will work, although most microcomputer systems isolate memory components from the 68000's address and data buses by means of buffers or data bus drivers. The data bus of the 68000 is connected directly to the data buses of the HM6116Ps. RAM_1 is connected to D_{00} to D_{07}, and RAM_2 to D_{08} to D_{15}.

Address lines A_{01} to A_{11} from the 68000 are connected to the address inputs of the two HM611P RAMs. The address inputs of the RAMs are wired in parallel, so that the same location is accessed in each chip simultaneously.

The R/W input of each RAM is connected directly to the 68000's R/$\overline{W}$ output via an OR gate strobed by AS*. Each OE* is connected to the processor's R/$\overline{W}$ output via the inverter. It is only the active-low chip select, CS*, inputs of the two RAMs that are treated differently. Before dealing with CS*, a little has to be said about address decoding.

Address lines A_{01} to A_{11} of the 68000 select one of 2k unique locations within the RAMs. The higher-order address lines A_{12} to A_{23} define 2^{12} or 4k possible blocks of 2k (note that 4k blocks of 2k words = 8 M words). In order to uniquely assign the 2k words of RAM to one of these 4k possible blocks, address lines A_{12} to A_{23} must take part in a decoding process whereby only one of the 4k possible values spanned by these address lines is used to generate CS*.

The simplest possible address decoder is formed from a 13-input NAND gate, whose output is active-low only when all address inputs are true. Thus, the SELECT* output of the NAND is asserted whenever an address in the 2k word (ie 4 kbyte) range $FF F000–$FF FFFF appears on the address bus.

Table 5 shows how the two signals UDS* and LDS*

Table 5. Generating CS_1* and CS_2* from the 68000's data strobes

SELECT*	UDS*	LDS*	CS_2*	CS_1*	Operation
1	X	X	1	1	No operation
0	0	0	0	0	Word read
0	0	1	0	1	Upper byte read
0	1	0	1	0	Lower byte read
0	1	1	1	1	No operation

NB X = don't care (may be 1 or 0)
1 = true (positive logic)
0 = false (positive logic)

from the CPU are combined with the SELECT* signal from the address decoder to generate CS_1* and CS_2*.

READ CYCLE CALCULATIONS

Having described the 68000's read cycle and a possible connection between the CPU and memory, the next step is to determine whether the CPU/RAM combination violates any timing restrictions.

The principal timing parameter of the RAM is its access time t_{AA}, which must be sufficient to meet the data setup time of the CPU (ie t_{DICL}). Figure 8 relates the essential features of the 68000's timing diagram to those of the HM6116P RAM. From the falling edge of S_0 to the falling edge of S_6, three full clock cycles take place, a total time of $3t_{cyc}$. During this time, the contents of the address bus become valid (t_{CLAV}), the memory is accessed (t_{AA}), and the data setup time met (t_{DICL}). Thus, the total time for this action is given by $t_{CLAV} + t_{AA} + t_{DICL}$. Putting the two equations together we get

$$3t_{cyc} > t_{CLAV} + t_{AA} + t_{DICL}$$

or

$$t_{AA} < 3t_{cyc} - t_{CLAV} - t_{DICL}$$

or

$$t_{AA} < 3 \times 125 - 70 - 15 \quad \text{(all values ns)}$$
$$< 290 \text{ ns}$$

The RAM must have an access time of less than 290 ns to work with the 68000L8 at 8 MHz. As the quoted value of t_{AA} for the HM6116P is 200 ns, the access time criterion is satisfied by a reasonable margin. It is interesting to consider what the demands on t_{AA} would have been, if a 12.5 MHz version of the 68000 had been used. The value of t_{AA} is now given by

$$t_{AA} < 3 \times 80 - 55 - 10$$
$$< 170 \text{ ns}$$

The HM6116P RAM cannot be used at 12.5 MHz without the addition of any wait states.

The next criterion to be considered is the value of the data hold time ($t_{SHDI} = 0$ ns minimum) required by the CPU following the rising edge of AS*. There is no problem here, because it can be seen from Figure 6 that the address does not change until the start of state S_0 in the next cycle, which means that the data from the RAM will be valid (nominally) throughout state S_7. Following the rising edge of AS*/UDS*/LDS* the data bus drivers are turned off in the RAM.

However, the data bus driver will not be floated instantly, and the data hold time of 0 ns will be met.

The control of CS* presents no problem. As CS* is derived from SELECT*, and LDS*, it is asserted very early in a read cycle, approximately 10 ns (t_I) after the falling edge of AS*. This turns on the data bus drivers in the RAM early in the cycle, although the data is invalid until after the RAM's access time has been met. At the end of a read cycle, CS* is negated when AS* rises no more than t_{CLSH} (70 ns) after the falling edge of state S_6. The data bus is floated no more than $t_{CLSH} + t_2 + t_{CHZ}$ s after the start of S_7. The low to high transition of the address decoder output occurs t_2 s after the negation of AS*. For a 68000L8 and 6116P-4 combination with $t_2 = 10$ ns and $t_{CHZ} = 60$ ns, the guaranteed turn-off time measured from the end of S_6 is $70 + 10 + 60 = 140$ ns.

As the duration of S_7 is nominally 62.5 ns, the data bus may not be floated until up to 77.5 ns into the following S_0. Fortunately, the next access does not begin until S_2, and so there is no chance of bus contention occurring. That is, the next access must not try to put data on the data bus until all the data bus drivers have been turned off following the current cycle. The write cycle of the 68000 is very similar to its read cycle and will therefore not be dealt with here.

A MINIMAL CONFIGURATION USING THE 68000

People occasionally ask, how few chips it takes to build a microcomputer with a 68000 CPU? In some ways this is an unfair question, because it tries to pin down the 68000 to a largely spurious figure of merit (ie a minimum chip-count design). This question takes no account of performance and is based on a rather dubious assumption that low chip count is related to low cost or ease of construction. Having given this warning, I am now going to look at a low chip-count 68000 microcomputer. My motives are twofold. I wish to demonstrate which pins of the 68000 are essential to a simple microcomputer and which pins can be 'forgotten about' in a minimal design. Second, it is sometimes necessary to produce a really small system, either as a teaching aid to illustrate the processor, or as a stand-alone controller.

While it is possible to design a 68000 microcomputer subject to the constraint of a minimum chip count, this is a rather pointless exercise, as the addition of one or two extra chips may result in a vastly

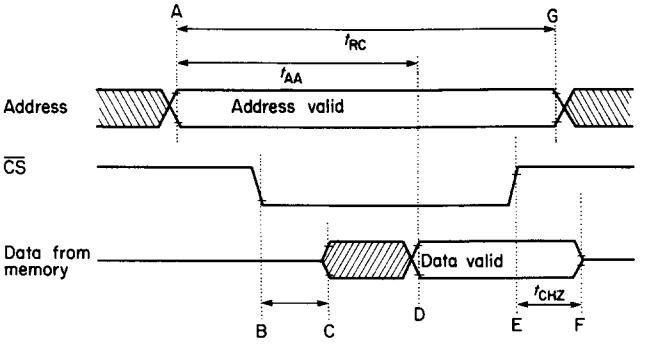

Figure 8. The timing diagram of a 68000 and HM6116P-4 combination

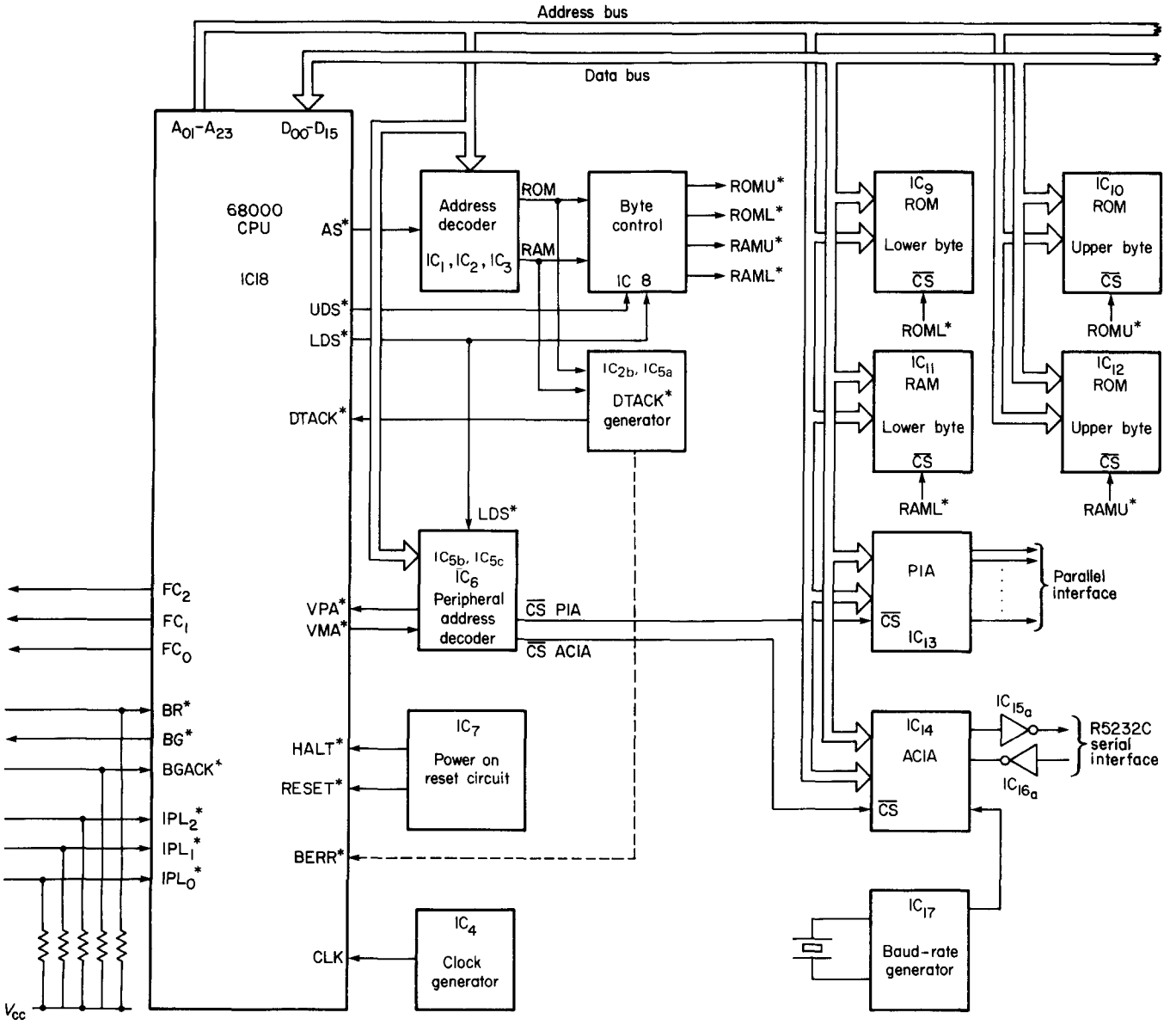

Figure 9. The block diagram of a minimal 68000-based microcomputer

increased level of performance. Instead, I intend to design a system subject to the following constraints.

- The microcomputer is to be used in a standalone mode and requires only a power supply and an external terminal.
- It is intended to be used as a classroom teaching aid to demonstrate the characteristics of the 68000.
- It must have a 16 kbyte EPROM-based monitor.
- Its speed (ie clock cycle time) is of little or no importance.
- It must have at least 4 kbytes of read/write memory.
- It must have at least one RS232C serial I/O port and one parallel port.
- It must be possible to expand the memory and peripheral space of the microcomputer later.
- Interrupts and multiprocessor capabilities are not needed, but again it should be possible to add them later.

The first step in designing our minimal system is to consider the major components, the ROM, RAM and peripherals. The ROM is provided by two 8k × 8 components, the RAM as two 2k × 8 devices and the peripherals as a 6821 peripheral interface adaptor (PIA) and a 6850 asynchronous communications interface adaptor (ACIA). Figure 9 shows how they are arranged in the microcomputer module.

The next step is to consider the memory and peripheral support circuitry. Clearly, the 16 kbytes of ROM and the 4 kbytes of RAM have to be selected out of the 68000's 16 Mbytes of memory space. The actual location of these devices within this space is largely unimportant, as long as the reset vectors are located at $00 0000. Consequently, the 16 kbytes of ROM are situated at $00 0000 to $00 3FFF.

The circuit diagram of the control circuitry of the minimal single board computer is given in Figure 10. Address decoding is carried out by three integrated circuits: IC_{1a}, IC_{1b}, IC_{2a} and IC_3. These divide the memory space in the region $00 0000 to $01 FFFF into eight blocks of 16 kbytes. The first three consecutive blocks at the upper end of the memory space are devoted to ROM, RAM and peripherals respectively.

Whenever the Y_0* or Y_1* outputs of IC_3 go active-low, signifying the selection of ROM or RAM, the output of NAND gate IC_{2b} goes high. This is complemented by open-collector invertor IC_{5a} to become the processor's DTACK* input. Note that no delay is applied to

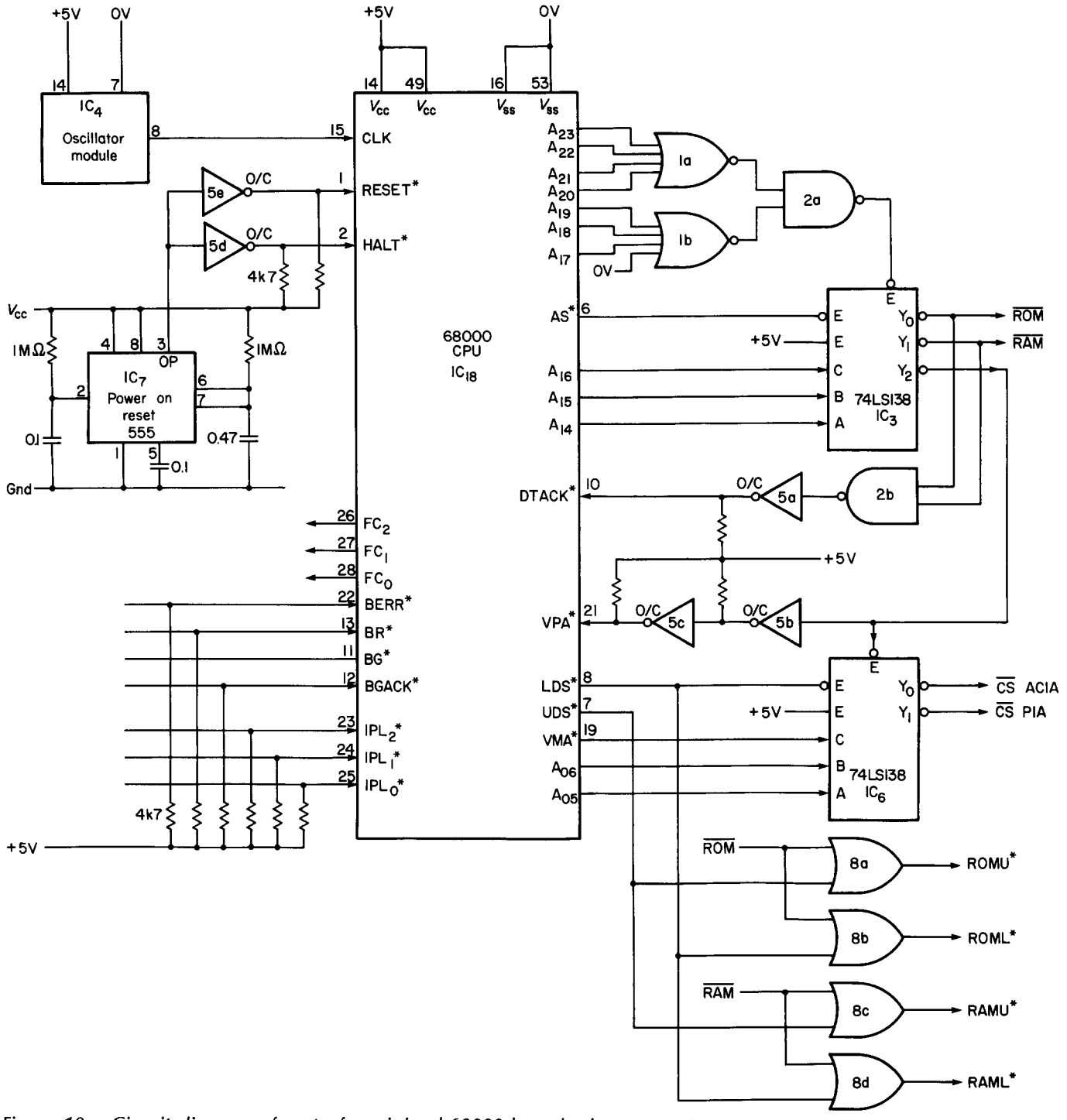

Figure 10. Circuit diagram of part of a minimal 68000-based microcomputer

DTACK*, so we must match the processor to its memory carefully.

The Y_2* output of IC_3 goes active-low whenever a peripheral is addressed. This is buffered by IC_{5b} and IC_{5c} to permit the VPA* input of the CPU to be driven by an open-collector gate. In this way other open-collector outputs may drive VPA* if they are added later. Y_2* is further decoded by IC_6† to generate peripheral chip selects for the PIA and ACIA.

The power-on-reset circuit forces RESET* and HALT* low when the system is initially switched on. A monolithic DIL clock generator chip supplies the processor with its clock signal.

In this application the interrupt request inputs, IPL_0* to IPL_2*, are pulled up by resistors to their inactive state. The function code outputs, FC_0 to FC_2, are not required and are left unconnected. Finally, both the bus request (BR*) and bus grant acknowledge (BGACK*) inputs are pulled up into their inactive-high states by resistors. The bus error input (BERR*) is not used and is also pulled up by a resistor. The 6850 ACIA requires its own clock which is supplied by baud-rate generator IC_{14}. Its serial inputs are buffered by a line transmitter, IC_{16}, and its outputs by a line transmitter, IC_{15}.

In all, this minimal 68000 system contains 18 integrated circuits. It would work as it stands and can be expanded to become a more sophisticated system.

† IC_6 is enabled by VMA* and LDS*. This means that a peripheral is synchronised to a 68000 synchronous cycle operation (triggered VPA* being asserted), and that the CPU must address a lower byte to select a peripheral.

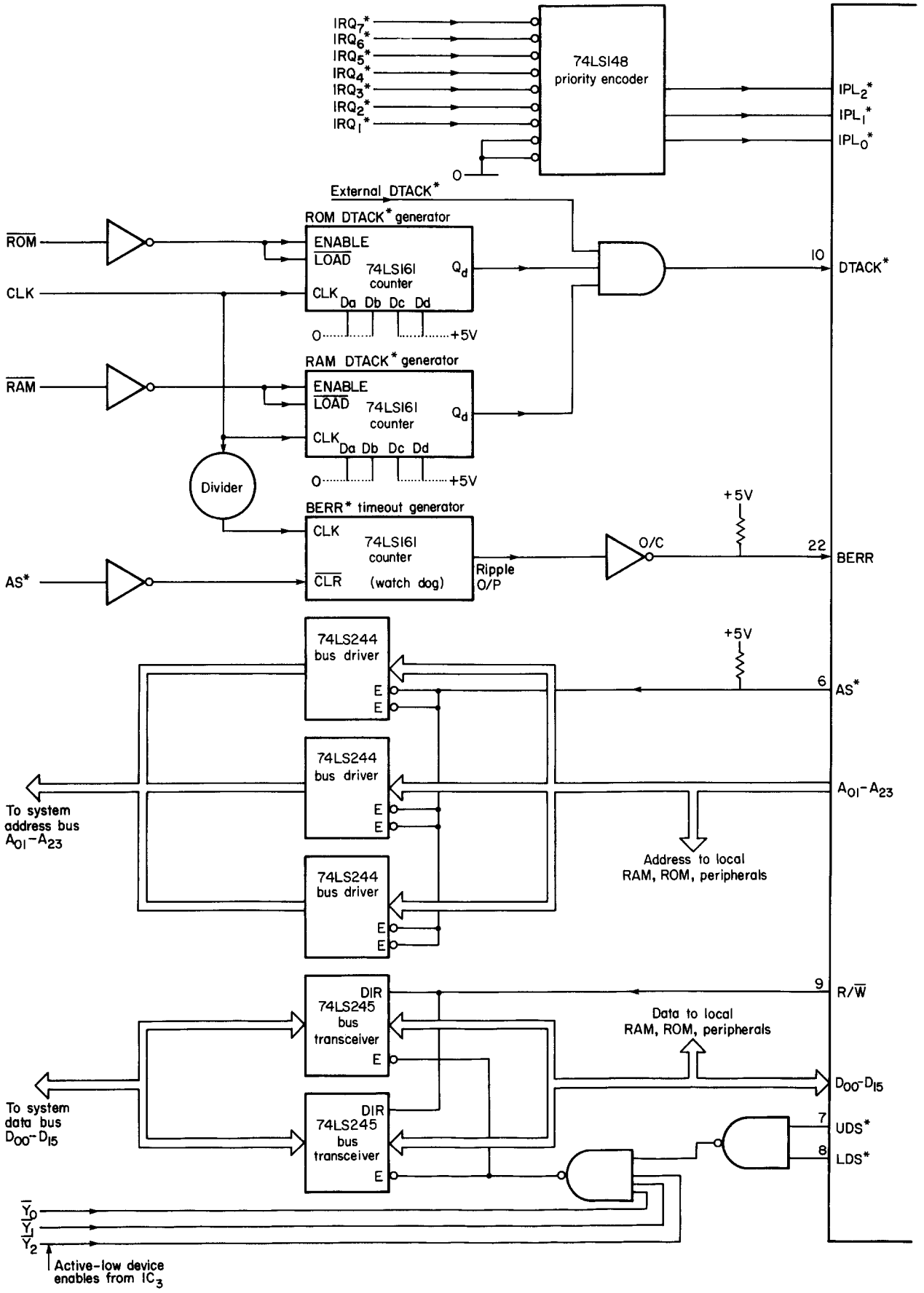

Figure 11. Turning the minimal microcomputer into a general-purpose SBC

A CRITIQUE OF THE MINIMAL COMPUTER

The minimal computer of Figures 9 and 10 is practical, but only just. It lacks various features whose inclusion costs little in terms of the chip count, but which considerably enhance the system. Some of the areas in which the minimal computer can be improved are as follows.

Control of DTACK*

As it stands, the circuit of Figure 10 provides a poor implementation of the DTACK* input to the 68000. Two problems have not been considered. The first concerns the operational speed of the processor. If the CPU is to run at its maximum rate and moderately fast RAM is used, it is necessary to delay DTACK* only when the slower EPROM-based read-only memory is accessed. Figure 11 shows how individual DTACK* delays can be generated, one for RAM accesses (if necessary) and one for EPROM accesses. The second problem concerns the possibility of accesses to unimplemented memory. If a read or write access is made to memory not decoded in Figure 10, the DTACK* input is not asserted and the processor will hang up indefinitely. In Figure 11 a watchdog circuit is used to overcome this difficulty. When AS* is asserted, a timer is triggered. The timer is reset by the rising edge of AS*. If DTACK* is not asserted, the timer is 'timed-out' and the BERR* input to the 68000 is asserted to indicate a bus error. This allows the processor to proceed.

Control of interrupts

While it is not necessary to operate the 68000 or any other processor in an interrupt-driven mode, it is worthwhile providing some form of interrupt facility in a general-purpose digital computer. Figure 11 shows how seven levels of interrupt request input can be provided by a 74LS148 priority encoder.

External bus interface

If a microprocessor system is to be expanded, it must be able to communicate with external systems via a bus. In a large system with many memory components or peripherals, it is impossible to connect the 68000's pins directly to a system bus because the CPU cannot supply the current necessary to drive the distributed capacitance of the bus and all the inputs connected to it. Therefore, special-purpose circuits called bus drivers or buffers are interposed between the processor and the system bus. In addition to the bus drivers themselves, it is necessary to provide control circuitry to avoid data bus contention, which could occur when the CPU reads from memory local to the processor module.