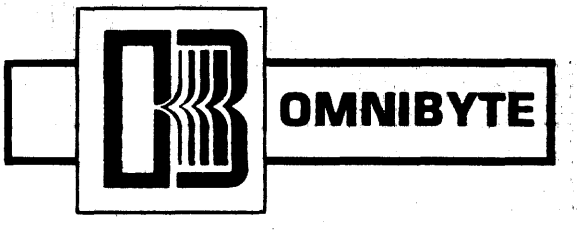
OB68K1A

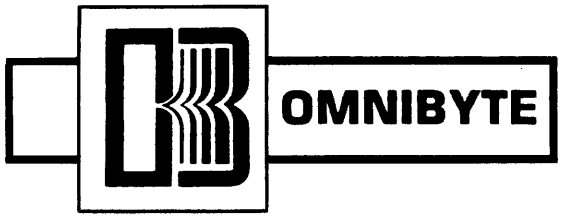
MC68000 SINGLE BOARD COMPUTER
USER'S MANUAL



OB68K1A

MC68000 SINGLE BOARD COMPUTER

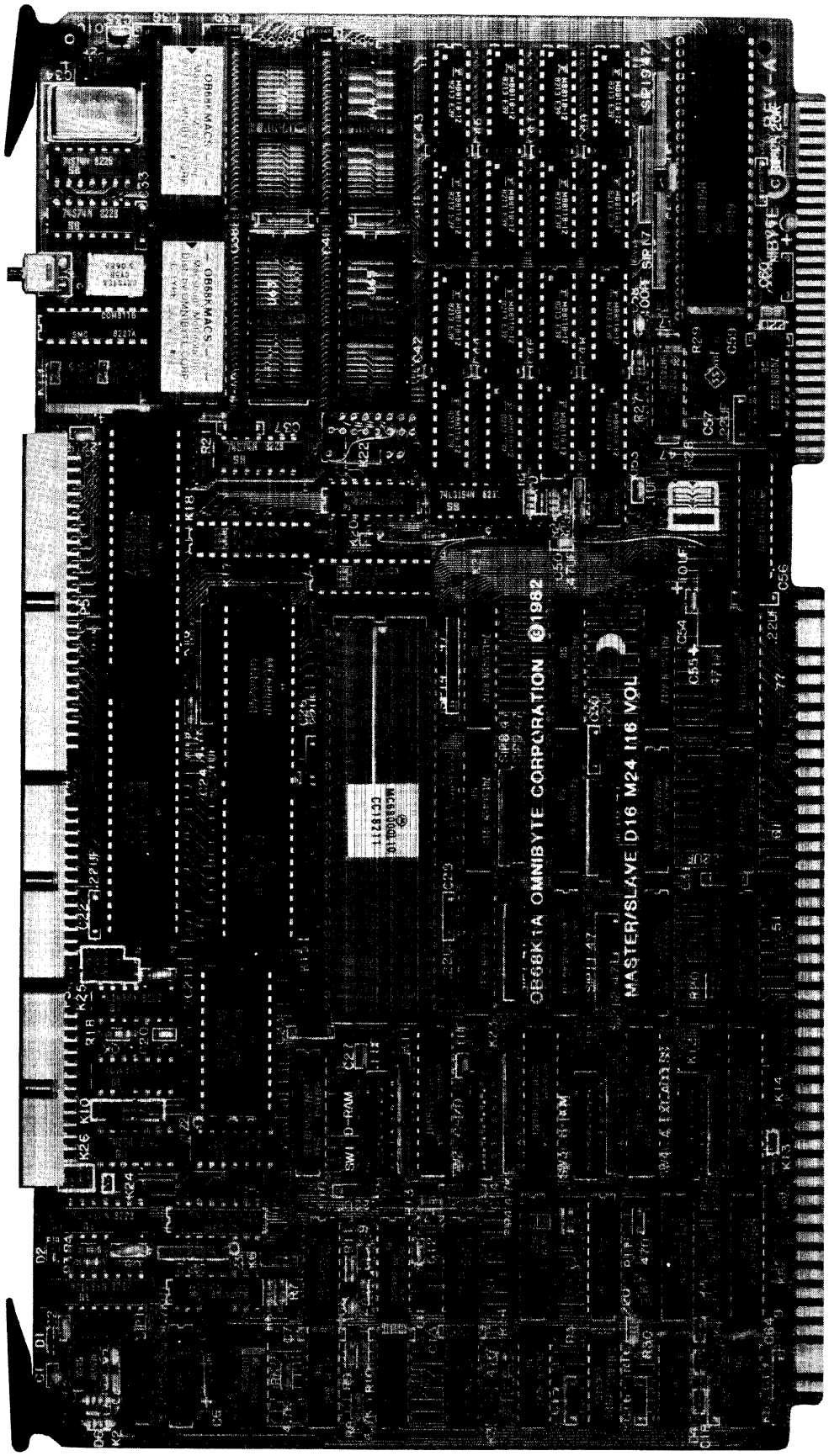
USER'S MANUAL



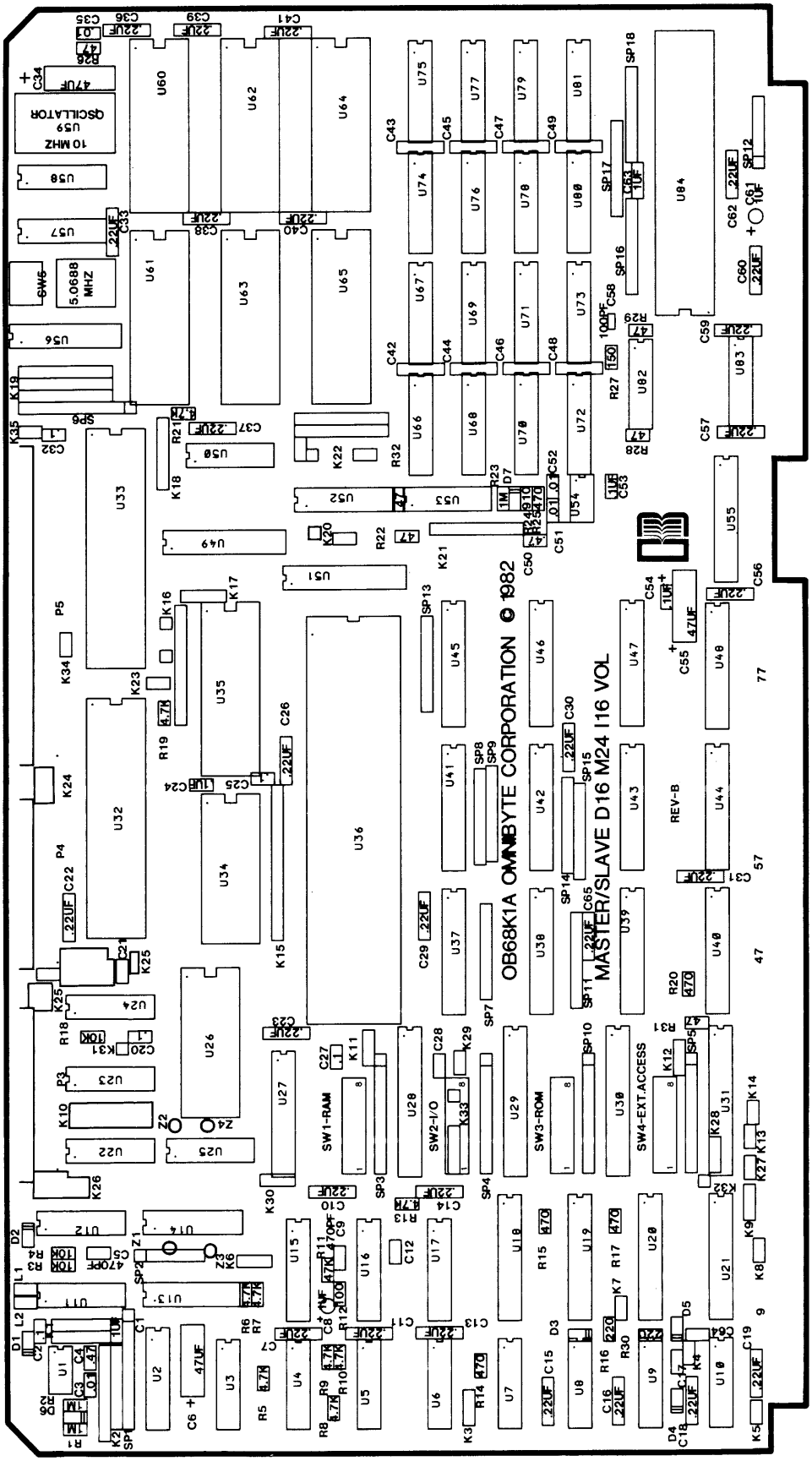
OMNIBYTE CORP • 245 West Roosevelt Road • West Chicago, Illinois 60185 • 312/231-6880

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES



PHOTOGRAPH OF THE OB68K1A
FIGURE 1.0



OB68K1A PARTS LOCATION DIAGRAM
FIGURE 1.1

1.0 INTRODUCTION / INSTALLATION

The OMNIBYTE OB68K1A 68000 Single Board Computer has been carefully designed to fulfill a variety of processing applications ranging from extremely small one board dedicated instruments to extremely large multi-processing systems utilizing several processor boards with shared memory and I/O. Figure 1.0 is a photograph of this board and Figure 1.1 is the parts location diagram.

1.1 Introduction

This chapter provides the unpacking, inspection and configuration instructions for the OB68K1A Single Board Computer.

1.2 Unpacking Instructions

> IF THE SHIPPING CARTON IS DAMAGED UPON RECEIPT, REQUEST THAT CARRIER'S AGENT BE PRESENT WHILE THE ITEMS ARE BEING UNPACKED AND INSPECTED.

Unpack the OB68K1A Single Board Computer from its shipping carton. Save the packing material for storing and reshipping the items in case this becomes necessary.

1.3 Inspection

The OB68K1A Single Board Computer should be inspected upon receipt for broken, damaged, or missing parts, and for physical damage to the printed circuit board or connectors.

1.4 Compatibility with Multibus Products

The OB68K1A Single Board Computer has been carefully designed to meet the most current IEEE 796 bus specifications. It is advised that you become familiar with these specifications and how they compare with the original and current Multibus specifications. The OB68K1A implements full address and bus arbitration for single and multi-processor systems and has been designed for compatibility with existing Multibus products. Omnibyte assumes no liability for non-compatibility of certain products which do not meet published IEEE 796 specifications.

1.5 Factory Standard Configuration

The OB68K1A Single Board Computer may be used in several configurations. Prior to inserting the OB68K1A in a system, care should be taken to install the proper jumper options where necessary for your system configuration. Refer to Figure 4.0 for physical locations of these jumpers on the OB68K1A. Included below is factory standard configuration information.

The OB68K1A is shipped in a configuration that allows it to be operated in a single master system without modification. Factory standard jumper configurations are given in Table 1.5. All cut trace options are as shown on the OB68K1A electrical schematic (See Figure 6.6). Standard jumper connections are indicated by dashed lines on the schematic.

JUMPER GROUP	CONFIGURATION	FUNCTION
K3	K3-1 TO K3-2	INIT line driven by OB68K1A
K4	Installed	BCLK driven by OB68K1A
K5	Installed	Serial Prioritization Enabled
K6	K6-2 TO K6-3	BERR enabled
K7	Installed	BPRN grounded
K8	Removed	BREQ for serial arbitration
K9	K9-2 TO K9-1	CBRQ connected to Multibus
K10	K10-5 TO K10-6	Normal RTS (Circuit Board Traces)
	K10-4 TO K10-5	Serial Port transparent mode enabled
K11(32K)	K11-1 TO K11-2	Onboard RAM begins on 32K boundaries (with K29 installed)
K11(128K)	K11-2 TO K11-3	Onboard RAM begins on 128K boundaries (with K29 removed)
K12(32K)	K12-2 TO K12-3	External access RAM address begins at 32K boundary (with K27 installed)
K12(128K)	Removed	External access RAM address begins at 128K boundary (with K27 removed)
K14	Installed	CCLK driven by OB68K1A
K18	Removed	Hardware selected Baud Rate
K19	K19-13 TO K19-21	Serial Port 0 Baud rate set to 9600
K19	K19-9 TO K19-17	Serial Port 1 Baud rate set to 9600
K20	K20-2 TO K20-3	ROM DTACK set for 4 wait states
K22	Installed	2764 PROM type
K23	Removed	External access watchdog timer enabled
K27(32K)	Installed	External Access RAM begins at 64K boundary (with K12 removed)
K27(128K)	Removed	External Access RAM begins at 128K boundary (with K12 removed)
K29(32K)	Installed	Onboard RAM begins at 64K boundary (with K11 removed)
K29(128K)	Removed	Onboard RAM begins at 128K boundary (with K11 removed)

FACTORY STANDARD JUMPER CONFIGURATION
TABLE 1.5

The OB68K1A is configured at the factory to operate in the following way;

a) BUS

The Bus Clock (BCLK), Constant Clock (CCLK) and Reset line (INIT) are driven off the board. On-board power-on reset enabled and the bus error jumper (K6) is installed so that bus error exception processing will be executed, if a bus error is encountered.

b) INTERRUPTS

No interrupts are connected.

c) RAM

On-board RAM begins at $OOOOOO (HEX). Contiguous RAM continues to $OO7FFF (HEX) in the 32K version and to $O1FFFF (HEX) in the 128K version.

d) ROM

All on-board ROM sockets are configured for 2764-type (8Kx8) 5 volt only EPROM and the memory map is configured to MAP 1 (for 2764). ROM address begins at $FEOOOO (HEX) and continues to $FEFFFF (HEX). The ROM DTACK is factory preset for 350 ns (access time) ROM chips.

e) ON-BOARD SERIAL I/O PORTS

On-board I/O begins at address $FFFEOO (HEX). See table 6.2 for specific device address assignments. Serial Port 0 is configured as a modem for direct connection to a RS232C terminal. Serial Port 1 is configured as a terminal for direct connection to a RS232C modem or another computer. The baud rates are set to 9600 BPS at the factory for testing.

f) OFF-BOARD I/O

Off-board I/O begins at address $FFOOOO (HEX) and continues to $FFFDFF.

g) TRANSPARENT MODE

Transparent mode is enabled.

Please note that the above is the configuration of the OB68K1A as shipped from the factory and it does not include setting-up the board in a different configuration if desired before power-up. A detailed list of the factory installed jumper configurations is given in Table 1.5. Factory standard configuration is compatible with Omnibyte's optional PROM based terminal monitor routines that provide the functionality of Motorola's MACSbug/VERSAbug/VMEbug/TUTOR programs.

2.0 OVERVIEW OF THE COMPUTER BOARD

This section describes the major features of the OB68K1A. A block diagram of this single board computer is shown in Figure 2.0.

2.1 Summary of Features

The OB68K1A computer board provides the following features:

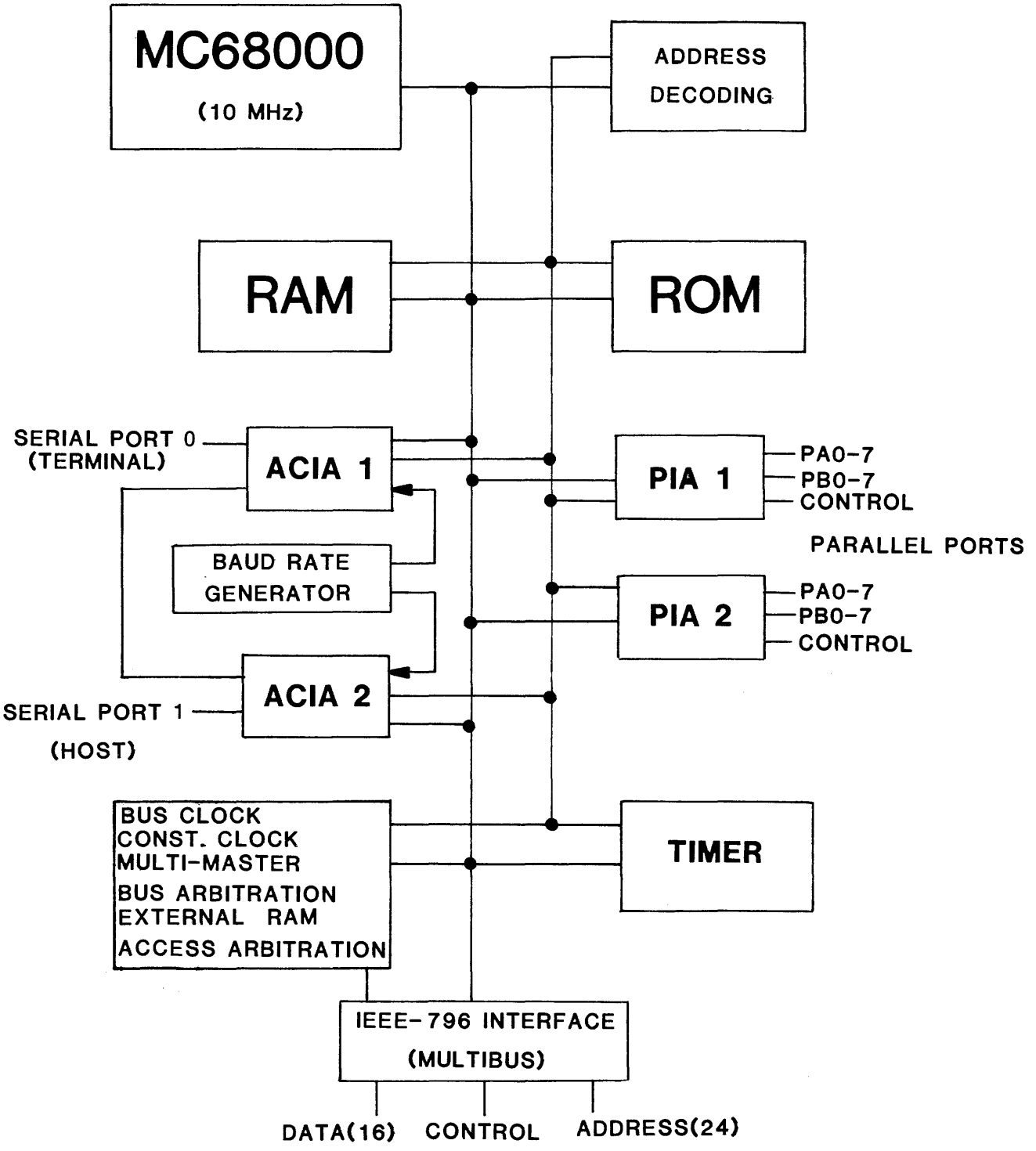
a. 10MHz processor & clock
b. IEEE 796 (Multibus) Compatible (MASTER D16 M24 I16 VOL/SLAVE M24 D16)
c. Single step circuitry
d. Dual Ported on-board RAM (32K byte or 128K byte)
e. Zero wait states for on-board RAM accesses
f. LSI Hardware memory refresh circuit
g. On-board ROM (up to 192K bytes)
h. Two asynchronous serial ports (RS232C)
i. Hardware or software programmable baud rate generator
j. Two programmable 16-bit parallel I/O ports
k. Three 16-bit programmable timers
l. 16 Megabyte (24-bit) direct memory addressing
m. Independantly Switch Selectable RAM, ROM, EXT. RAM ACCESS, and I/O base addresses
n. Multi-Master bus arbitration
o. Motorola MEX68KDM software compatibility

2.2 Power Requirements

The computer receives its power through the Multibus motherboard. Typical power requirements are as follows:

	32K VERSION	128 K VERSION
+5V — ± 5%	@ 3.0A	@ 3.25A
+12V — ± 5%	@ 0.05A	@ 0.05A
−12V — ± 5%	@ 0.05A	@ 0.05A

Note: Single 5 volt operation is possible with the OB68K1A if the RS232C ports are not used.



OB68K1A BLOCK DIAGRAM
FIGURE 2.0

3.0 GENERAL DESCRIPTION OF OB68K1A

The OB68K1A is designed to be both simple and flexible to use. Only general purpose memory and I/O are included on the board. Special purpose facilities such as disk controllers can be added as additional Multibus boards to configure larger systems.

3.1 Serial Interface

The two asynchronous serial ports are implemented using MC6850 Asynchronous Communication Interface Adapter (ACIA) chips. The baud rates are individually selectable for standard frequencies between 50 and 19,200 baud. The baud rate selection for each port may be determined either by hardware jumpers or by software setting. Both ports are factory configured to transmit and receive without handshake lines although jumper options are provided for the normal CTS, RTS, and DCD interface signals. All signals are received and transmitted through RS232C compatible buffers. The interface is made through individual, standard 26 pin header connectors.

3.2 Timer

A three channel 16-bit timer (MC6840) is available on the board. This timer is intended primarily for processor housekeeping. No connector is provided for input or output signals to or from the timer — the clock, gate and output pins for each timer are terminated on wire wrap posts near the chip. For normal applications the timer counts the processor "E" clock, a 1 MHz signal generated by the 68000.

3.3 Parallel Interface

Two MC6821 Peripheral Interface Adapters (PIA) are provided on the OB68K1A. All the interface lines are brought out on a standard 50 pin header connector. Five volt power is also brought to this connector. The two 8-bit PIA's are configured to straddle the 16-bit data bus so that 16-bit data may be transferred using the A ports or B ports of both PIA's. Byte operations are supported on both PIA's.

3.4 Bus Arbitration

The bus arbitration circuit allows this computer to share the Multibus with other processors or master controllers. RAM memory that resides on this board may be made available to other masters on the Multibus. The address of the Multibus port of the onboard RAM is independently selected.

3.5 On-Board Memory

3.5.1 On-Board Read Only Memory

The six memory sockets contained on the board are organized as three pairs of byte-wide memories. A minimum of 64K bytes of memory space is always allocated for on-board ROM. However, for ROMs of less than 64K bits each, the total ROM memory space will not be utilized. ***Only EPROM's that are pin compatible with the pinout shown in Figure 4.8.4 can be utilized on the OB68K1A.*** Some compatible memories are 2716, 2732, 2764, 27128 and 27256 5 volt only, 24/28 pin EPROM's containing 2K, 4K, 8K, 16K and 32 K bytes respectively. Each socket pair is selected with its base address and range within the allocated space arranged so that the PROM sockets form a contiguous block of memory. All three socket pairs occupy the same amount of memory. A small prewired PC plug is inserted in the location provided to configure the board for various sizes of ROM memories. Note that 28-PIN sockets are used for all ROMs to allow for a larger base of usable ROM chips. The base address of the ROM is selected as a single memory block using switch SW-3. Note that although various ROM sizes can be accommodated, all ROM sockets are configured for the same ROM size. Also see section 3.6.1.

3.5.2 On-Board Dynamic RAM

The OB68K1A has sixteen dynamic RAM chips. If configured for 16Kx1 parts, a total of 32K bytes are available. If configured for 64Kx1 parts, a total of 128K bytes are available. The board is offered with either the 16K or 64K parts. The RAM is selected as a block by the main address decoding circuit and may be addressed either in the byte or word mode.

The on-board dynamic RAM chips are accessed and refreshed using an LSI DRAM controller, the DP8409. This 48-pin bipolar circuit generates the Row Address Strobes (RAS), Column Address Strobes (CAS), drives the multiplexed address lines, and performs the necessary refresh cycles. Under normal operation the RAM controller receives the processor address strobes and becomes selected when the asserted address falls within the RAM memory space. A 555 clock generates a 15 us pulse train to provide the basic refresh timing. The controller monitors both the Address Strobe and its Chip Select inputs. When a refresh cycle is needed the controller waits for an Address Strobe to occur with no Chip Select — an indication that the processor is accessing other memory or I/O. The controller will then perform a "hidden" refresh cycle, that is, a refresh cycle that does not take time away from the processor. If, during the 15 us period of the refresh clock, the RAM is accessed on all Address Strobes, the controller will request the use of the bus, cause the processor to relinquish the bus and the RAM controller will then perform a forced refresh cycle.

The duration of forced refresh cycle is about 500 ns. Under most conditions the refresh overhead of the OB68K1A will be very low. Note that because the entire RAM refresh task is implemented in hardware no processor code-execution cycles are wasted to perform RAM refresh by software.

When the processor performs a Multibus cycle, the duration of the cycle will depend on the availability of the bus and the response time of the addressed device. During these offboard cycles, no address strobes are applied to the RAM controller and forced refreshes are then implemented when needed. The XACK signal output by the OB68K1A signals the external master that data from the onboard RAM has been placed on the Multibus or the data to be written has been stored. Shortly thereafter the external master should terminate the cycle by removing the read or write command. If the cycle is not terminated after several microseconds, data in the dynamic RAM could be lost. A timeout circuit is included on the OB68K1A that interrupts the external access so a refresh can occur. The offboard master is not required to wait for the completion of an instruction. The LOCK feature of Multibus is implemented so an offboard master may do a test-and-set operation to the onboard RAM. Also see sections 3.6.2 and 3.6.4.

3.6 Address Decoding and Memory Mapping

The OB68K1A has been designed so that the memory mapping is switch selectable by the user. No fusible link devices are needed to change the memory map. Four separate address decoders are included on the board to individually select the base address of onboard RAM, onboard ROM, I/O and the Multibus access to the onboard RAM. For each of these four blocks, the base address is selected by the setting of an 8-bit DIP switch. The dip switches have been socketed so that dip jumpers can be used to fix addressing for production purposes, these dip jumpers are available through OMNIBYTE. See section 9.0 for ordering information. The switch setting is compared with the upper address lines to determine when the various blocks are being selected (SWITCH BIT 1 = L.S.B., SWITCH BIT 0 = M.S.B.; ON = 0, OFF = 1). Much of the random logic associated with address decoding and strobe timing has been consolidated into Programmable Array Logic (PAL)* chips. ***These circuits are programmed at the factory and cannot be changed by the user.***

* PAL is a trademark of Monolithic Memories, Inc.

3.6.1 ROM Address Selection (SW-3)

The 8-bit DIP switch (SW-3) is used to select the base address of the onboard ROM. Address lines A16-23 are compared with the switch setting resulting in a minimum ROM block size of 64K bytes. The minimum block size will accommodate memories through 8K bytes per chip. For larger ROM chips, larger memory block sizes of 128K and 256K bytes may be selected by removing A16 and A17 from the comparator inputs, respectively. All the necessary connections are made by inserting pre-wired plugs in the location provided. Various jumper models are available from the factory. Note that within the ROM block, there are addresses for which no memory exists. For example, using 8K byte PROMs, the six sockets provided will occupy 48K of the minimum 64K byte block size. The remainder of this block is not accessable and cannot be assigned to other memory or devices.

3.6.2 RAM Address Selection (SW-1)

The 8-bit DIP switch (SW-1) is used to select the base address of the onboard RAM. Jumpers and cut traces at A15 and A16 are factory configured for either 32K or 128K RAM. For 32K RAM switch SW-1 selects base addresses on the lower 32K block of 64K byte boundary. For 128K RAM switch (SW-1) selects base address on 128K byte blocks. For 128K RAM, BIT 1 of SW-1 is a "don't care" bit, since pins 13 and 14 on U28 are tied together and the cut trace option at A16 is opened.

3.6.3 I/O Base Address Selection (SW-2)

Because the 68000 accesses I/O the same as memory, a 64K byte block of memory space is decoded and assigned to the I/O space defined in the IEEE 796 bus specification. No options exist on the board to change the size of the I/O address space, but the base address may be located at any 64K byte boundary within the available 16 megabyte address range. The onboard I/O devices occupy the uppermost 512 bytes of the I/O space; all other addresses within the block default to offboard I/O. Table 6.2 gives the addresses of the onboard I/O devices.

3.6.4 EXTERNAL RAM Access Address (SW-4)

A separate 8-bit DIP switch and comparator are used to determine the base address of the onboard RAM when that RAM is accessed by another Multibus master. The RAM is then accessed at independently selected addresses chosen to satisfy requirements of both the offboard master and the OB68K1A. Address line A15 can be used to gate the comparator in order to decode a 32K byte block size and jumper options allow the block size to be increased. Notice that it is possible to limit the amount of memory that can be accessed from offboard.

A 128K byte OB68K1A may allow offboard masters to access 32K, 64K or all 128K bytes of its onboard RAM. This feature allows portions of the onboard RAM to be protected from other offboard masters. Although limiting blocks of RAM from off-board access is possible, this option has not been implemented by the factory and should be selected by the user if desired. Also note that in addition to the jumper and cut trace option on A16, pins 13 and 14 on U31 have been jumpered and bit 1 of SW-4 is a "don't care" bit. (See Section 3.6.2).

3.6.5 Operational Considerations

When the user is setting the base address switches of the OB68K1A care must be exercised to avoid overlap of the onboard memory spaces of RAM ROM, and I/O. The offboard access to the dual-ported RAM may arbitrarily overlap any or all of the onboard RAM, ROM, or I/O space.

In systems that use multiple OB68K1A computers, the onboard base address selections may be the same for all boards.

The external access to the OB68K1A dual-ported RAM appears as a simple RAM board to an external Multibus Master. Therefore, the base address of the external access (SW 4) must be selected to avoid overlap of the dual-ported memory space with any other memory space on the Multibus. External access spaces of Multiple OB68K1A boards must not overlap. The OB68K1A is protected from accessing its own RAM via the Multibus by negating onboard access while performing offboard accesses.

3.6.6 Undecoded Addresses

The OB68K1A is designed so that all memory accesses default to offboard Multibus accesses unless the address that is asserted falls within the onboard RAM, ROM or onboard I/O space.

3.7 Transfer Acknowledge and Bus Errors

The 68000's data transfers are asynchronous — A Data Transfer ACKnowledge (DTACK) signal is required to complete an access. For Multibus cycles this signal is provided naturally by the Multibus Transfer ACKnowledge (XACK). For on-board ROM cycles, a DTACK generator is provided to terminate the cycle a fixed time after an on-board memory access is started. The delay time is selectable to match the access time of the on-board ROM memory chips. (See Section 4.3).

The DTACK signal for onboard RAM access cycles is also provided by the ROM DTACK generator and has also been optimzed for the RAM. The OB68K1A must generate an XACK signal when other Multibus masters access the onboard RAM. This signal serves to terminate the bus cycle for the offboard master. A tap on the ROM DTACK generator is used to generate this signal at a delayed time that has been optimized for the offboard RAM access time and no user adjustment is permitted.

When the OB68K1A accesses external Multibus memory or I/O, the cycle is terminated by the XACK signal returned by the board that was addressed.

In the event that unimplemented off-board memory is accessed, no DTACK will be generated. An on-board "watchdog" timer is included to detect a lack of response, and a pulse is generated that may be jumpered to the 68000 Bus Error input pin. A signal asserted on this pin will initiate bus error exception processing and a user-supplied routine is executed to allow the system to analyze the report or recover from this condition.

Notice that the 68000 itself will patiently wait forever, if desired, for a DTACK response to come. No restriction is placed on the speed of response of the addressed memory or device. The watchdog timer delay is user determined and its implementation is optional. It is included to keep the system from hanging up if no response is received.

Conditions that will cause a bus error are:

a) Access to off-board memory addresses that have no responding memory (not plugged in, or not working).

b) Access to off-board I/O addresses that have no responsing device (not plugged in, or not working).

An access to an on-board memory address will not cause a bus error even if a memory chip is not installed. On-board I/O uses the 68000 synchronous transfer capability and no DTACK is required.

3.8 Function Codes

The 68000 processor outputs three function code bits, FC0, FC1, and FC2 that allow external circuitry to know the internal operating mode of the processor. The state of these outputs indicates whether the processor is in the supervisor or user state, whether the present access is a program or data reference, or if the processor is responding to an interrupt. The standard configuration of the OB68K1A makes use of various function code values only to recognize interrupt acknowledge cycles.

3.9 Clocks

Four clocks are generated on-board.

3.9.1 Processor Clock

A 10 MHz crystal oscillator provides the processor clock. It connects directly to the 68000 clock input pin. The 10 MHz clock is also used as the time base for the DTACK generator.

3.9.2 Baud Rate Clock

A 5.0688 MHz crystal and a COM8116 comprise the baud rate generator. This baud rate selection can be done by setting four jumpers for each serial port, or by storing the baud rate setting in the COM8116 chip under software control. These jumper options allow the processor to dynamically control the baud rate of either one or both serial ports.

3.9.3 Bus Clock and Constant Clock

The 10 MHz processor clock may be used as the Multibus BCLK and CCLK. Because **only one card in a Multibus system can assert these signals,** jumpers are provided to remove these clocks from the bus. With these jumpers removed, the card uses the BCLK and CCLK generated by another master in the Multibus bin.

3.9.4 The E Clock

The 68000 outputs an E clock that is one-tenth the processor clock frequency. Synchronous transfers to the on-board Motorola peripherals are made using this 1MHz clock. Accordingly, it is connected to the enable input of the PIA, ACIA and timer chips. This frequency is used by the timer chip when it is configured to count the E clock.

3.10 Interrupts

The 68000 provides for seven levels of prioritized auto-vectored interrupts. A 74148 priority encoder is included for inputting low active interrupts. The output of the 74148 directly connects to the IPL0, IPL1, IPL2 inputs of the processor. Interrupts outputs from the on-board peripherals and the Multibus interrupt lines must be connected to the priority encoder by the user. Wire wrap pins are provided for each interrupt source.

The 68000 feature of reading an interrupt vector number from the interrupting device has not been implemented on this board.

3.11 Status Indicators

Two LED indicator lights are included to show when the processor is in the reset mode (YELLOW) and when the processor is halted (RED).

312 Single-Step Mode

Circuitry is included on the OB68K1A to implement single step and halt operations. These functions are activated by an external momentary contact switch (for single stepping the processor) and an SPST switch to select the Single Step or Run mode. A 10-pin header on the board may be used to connect these switches. Additional signals included on this header are a Reset input, a Non-Maskable Interrupt input and connections to LED drivers for Halt and Reset indicators. This group of signals may be connected to a front panel if desired.

3.13 Restart Vector Accessing

When a power-on or manual reset of the procesor occurs, the processor begins operation by accessing memory location zero in Supervisory Program space to load the restart vector and the Supervisor stack pointer. These two vectors must be stored in PROM because the contents of RAM are unknown at restart time. The OB68K1A causes the first four memory accesses following a restart to be unconditionally directed to the location 0 through 7 of the first ROM socket pair (IC60 and IC61). The access to these PROM locations is independent of the switch-selected location of ROM in the address map of the OB68K1A. For proper operation, after restart some memory must exist at location $OOOOOO in order to have memory at the addresses where the processor expects to find exception vectors. This may be either RAM or PROM and, can be in offboard memory. In any case, ROM chips must be present in the first two ROM sockets, to supply the initial program counter and stack pointer values at restart.

3.14 Front Panel Connector

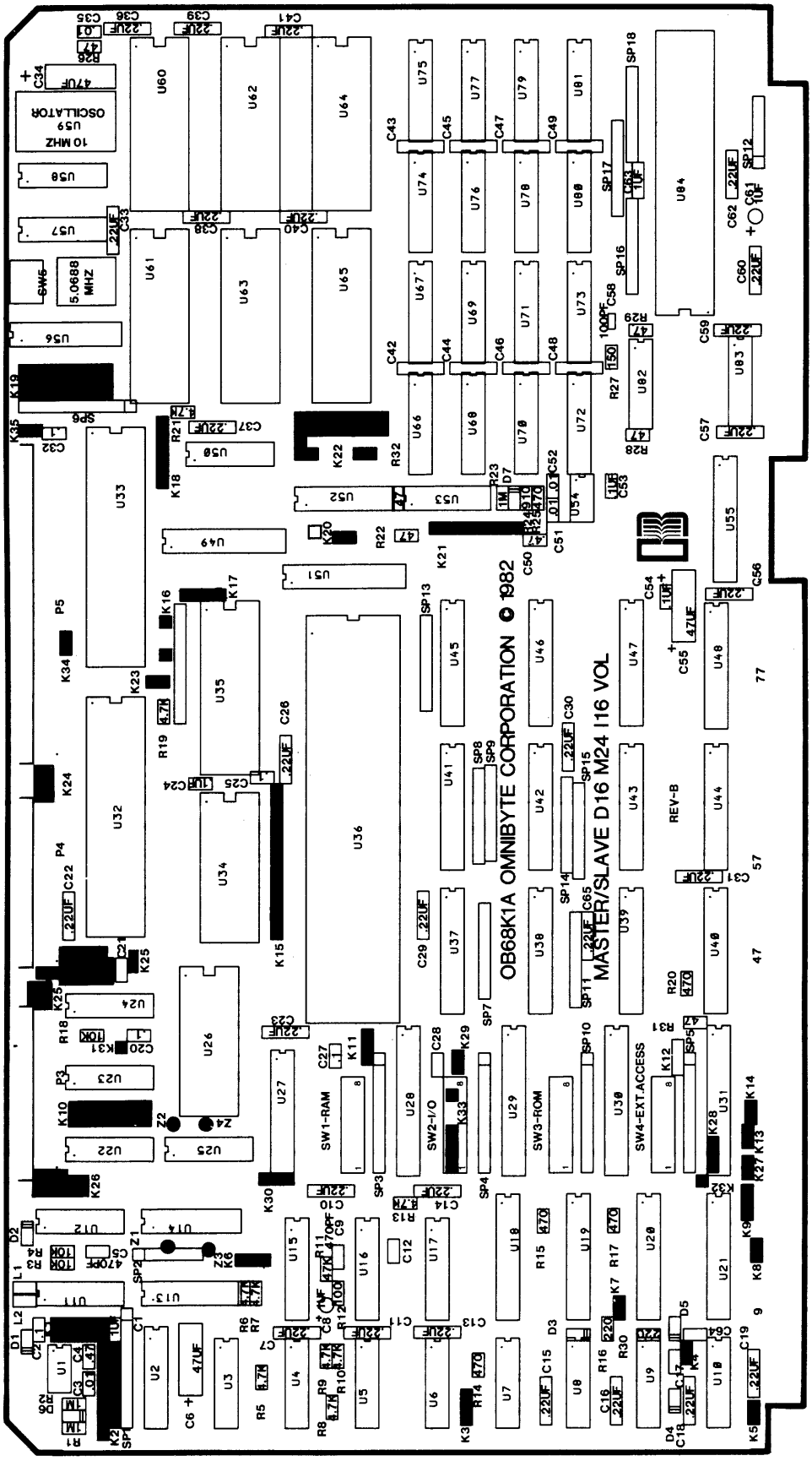
The OB68K1A has a provision for connecting a front panel board that includes RESET, SINGLE-STEP, and RUN-STOP switches, HALT and RESET LEDS and a Software Abort button that asserts NMI, the level 7 interrupt. An optional front panel box and interconnecting cable is available. See Section 9.0 for ordering information.

4.0 USER DEFINABLE OPTIONS

This section describes the jumpers and options included on the board. Table 4.0 is a summary of all the user definable jumpers. The location of the jumpers are shown in Figure 4.0.

JUMPER NO.	NO. OF PINS	FUNCTION
K1	10	Front Panel Connector
K2	15	Interrupt Jumpers
K3	3	INIT Jumpers
K4	2	BCLK to Multibus
K5	2	BPRO to Multibus
K6	3	BERR Enable
K7	2	BPRN Enable
K8	2	BREQ for parallel Arbitration
K9	3	CBRQ Enable/Disable
K10	7	Serial Port Transparent/Independent Mode Enable
K11	3	Onboard RAM Size Select
K12	3	Ext. RAM access, Upper/Lower 32K select
K13	2	Future RAM Enhancement
K14	2	CCLK to Multibus
K15	13	ACIA IRQ, Handshake Configuration; Test points
K16	10	Timer 2 & 3, CLK, Gate, Output; Timer IRQ, PIA 0 IRQ's
K17	4	Timer 1; CLK, gate, output
K18	6	SW Baud Rate strobe; PIA 1, IRQ
K19	24	Hardware/Software Baud Rate Jumpers
K20	3	ROM Delay (Jumper to K21)
K21	8	ROM DTACK Delay Select 100ns/tap
K22	21	Rom Socket Sizing
K23	2	External access watchdog timer enable
K24	4	± 12V TO Port 0 (For External Circuitry)
K25	20	Port 0 ACIA Configuration
K26	10	Port 1 ACIA Configuration
K27	2	A16 64/128K EXT. RAM Access Limit
K28	2	Future RAM Enhancement
K29	2	A16 Onboard RAM Size
K30	3	DTR Normal/Invert (Port 1)
K31	1	Spare RS232 Receiver Buffer Output
K33	4	Future RAM Enhancement
K34	2	5V for PIA 0
K35	2	5V for PIA 1

JUMPER OPTIONS
TABLE 4.0



OB68K1A LOCATION OF JUMPER OPTIONS
FIGURE 4.0

4.1 Serial Port Configuration (K25, K26)

The serial I/O is implemented using (2) MC6850 ACIA chips. When these are connected using ribbon cable and compatible 26 pin Berg type headers and 25 pin "D" connectors, the pinout of the "D" connector for port 0 is the same as an RS232C modem. The pinout for port 1 is the same as an RS232C terminal. That is, port 0 will plug directly onto an RS232C terminal and port 1 will plug directly onto an RS232C modem or host computer. The pinouts of these connectors are shown in Section 5.

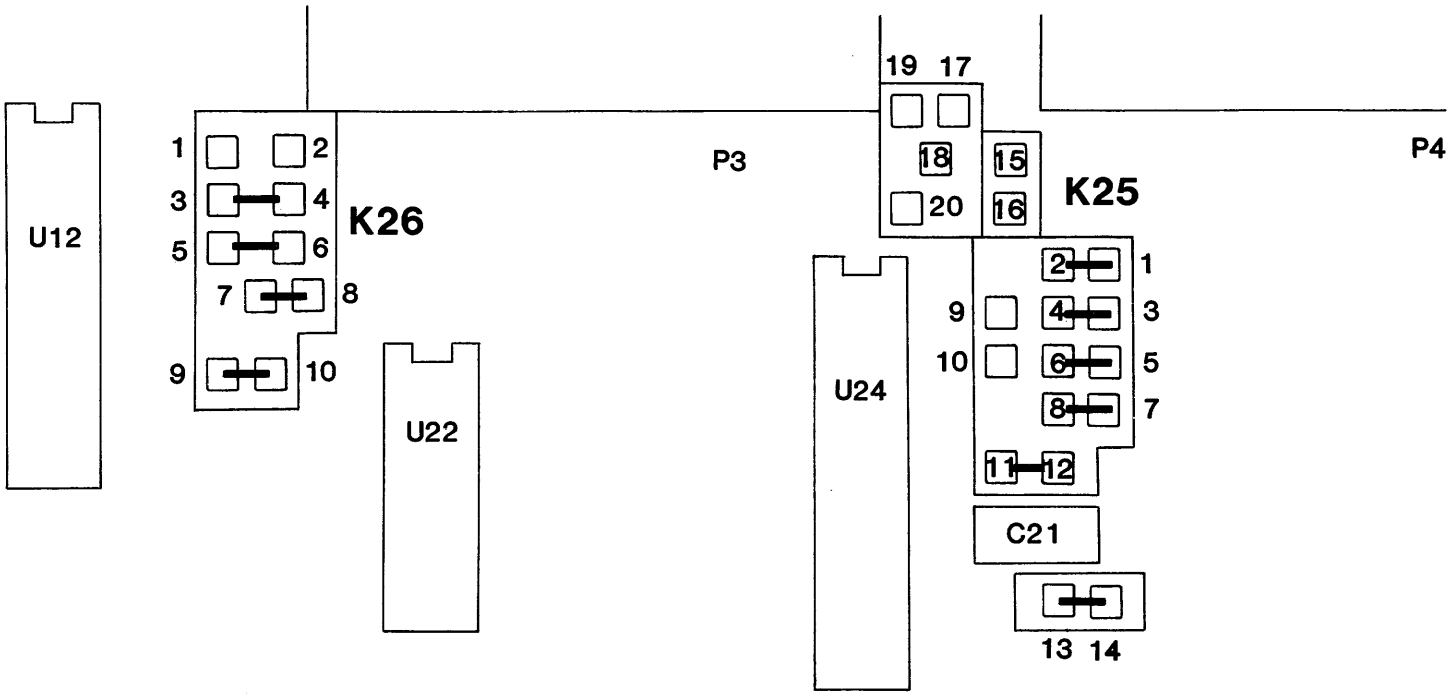
Port 0 ACIA has both CTS and DCD tied together and connected via an RS232C buffer to DTR on the connector. A pull up resistor is provided to automatically enable the ACIA if DTR is not driven by the terminal used. Cut trace options are provided to allow the user to reconfigure the port and implement handshaking if desired.

Port 1 ACIA has handshaking implemented with CTS and DCD. Both of these signals are provided with pull up resistors to enable the ACIA if CTS and/or DCD are not driven by the device being used. Cut trace options are provided to allow the user to reconfigure the port. See Figure 4.1 for the location of these points.

Both ports have jumper options to provide +5V, +12V and −12V power to the user's device. This could be used to provide power for an in-line RS232C to 20MA current loop converter. The factory configuration of Ports 0 and 1 are identical to the Motorola MEX68KDM design module's serial ports including the implementation of the "Transparent" mode of operation. TX and RX clock are connected together on each ACIA. Each port has independent baud rate selection.

4.1.1 Transparent Mode (K10)

The transparent mode enables communication between the devices connected to the two serial ports, such that the board appears "transparent" and for all practical purposes is not in the circuit. The transparent mode is enabled by a command in the terminal monitor program, MACbug or VERSAbug via the RTS output of the Port 0 ACIA. For successful operation of this mode, the user should take note of several points. The device connected to Port 0 must be data terminal equipment (DTE), factory configured for an RS232 terminal; the device connected to Port 1 must be data communication equipment (DCE), factory configured for an RS232 modem or host. The devices connected to the ports must have the same baud rate. Jumpers at jumper group K10 enables/disables the transparent mode, the factory configuration is K10-4 to K10-5 transparent mode enabled. If the user should want to reconfigure the serial ports, the transparent mode may be an undesirable mode in which to run, therefore, jumper option K10-3 to K10-4 has been provided to disable the transparent mode and configures the serial ports for independent operation.



K26
PORT 1 JUMPERS

1. +5V
2. P3-16
3. U12-1
4. P3-5(RXD)
5. U11-3
6. P3-14(DTR)
7. U11-8
8. P3-3 (TXD)
9. U12-13
10. P3-9 (CTS)

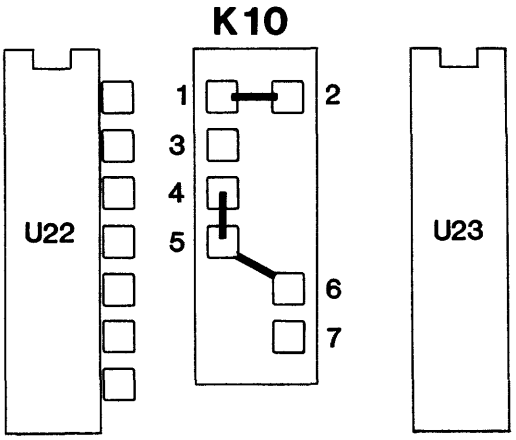
Note: P3 is PORT 1 Connector

K25
PORT 0 JUMPERS

1. P4-3(RXD)
2. U23-1
3. P4-7 (RTS)
4. U24-6
5. P4-9 (CTS)
6. U24-3
7. P4-15 (DCD)
8. U11-6
9. U23-13
10. P4-14 (DTR)
11. U24-11
12. P4-5 (TXD)
13. U24-8
14. P4-11 (DSR)
15. P4-16
16. +5V
17. −12V
18. P3-18
19. +12V
20. P3-20

Note: P4 is PORT 0 Connector

SERIAL PORT JUMPER OPTIONS
FIGURE 4.1



K10

1. U12-4
2. P3-15 (DCD)
3. GND
4. U22-13
5. U27-6
6. U24-5
7. U27-5

OPTIONS

4-5 TRANSPARENT MODE ENABLED
3-4 TRANSPARENT MODE DISABLED
5-6 NORMAL RTS
6-7 INVERTED RTS
1-2 DCD TO PORT 1

NOTE: CHIPS U12 AND U23 ARE RS232 RECEIVERS
CHIPS U11 AND U24 ARE RS232 DRIVERS

TRANSPARENT / INDEP. MODE JUMPER LOCATION
FIGURE 4.1.1

4.1.2 Baud Rate Selection (K18, K19)

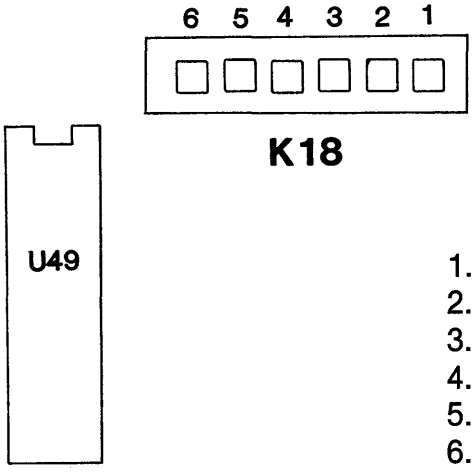
Jumper group K19 selects the serial baud rate for both Port 0 and Port 1. K19 is a group of 24 pins arranged in three columns as shown in Figure 4.1.2. The center row of pins connect to the baud-rate-setting pins of the COM8116 baud rate generator. The baud rate may be set manually or by software.

4.1.2.1 Manual Baud Rate Setting

The center row of pins in jumper group K19 are pulled up so that if a particular pin is not jumpered to ground, it will be pulled to a logic 1. To select a logic zero, a pin must be jumpered to ground by connecting to the corresponding pin in the right hand column. The baud rates corresponding to the possible values of the D, C, B, and A signals are given in table 4.1.2.1.

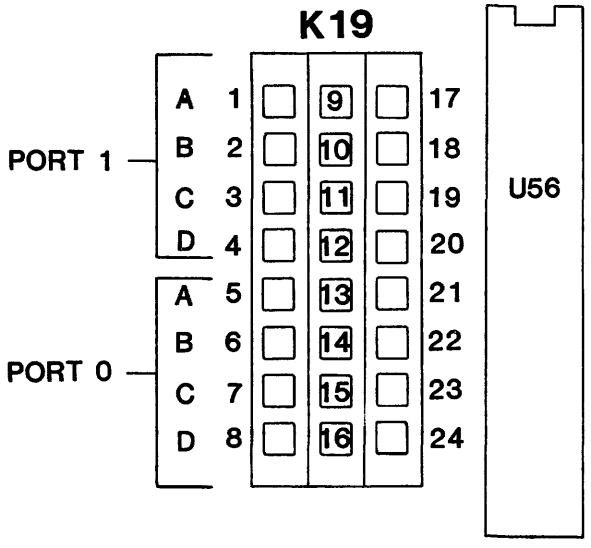
4.1.2.2 Software Baud Rate Setting

If the four pins corresponding to the DCBA inputs to the baud rate generator are jumpered to the pins in the left hand column of K19, the baud rate of the serial port may be written by the processor. To enable the Store Strobe for Port 1 and Port 0 install jumpers K18-3 to K18-4 and K18-1 to K18-2 respectively. Note that the baud rate selection for the two ports are completely independent so that any combination of manual or software baud rate selection is allowed. To set the baud rate under program control, the desired value is simply written to the baud rate generator address, see Table 6.2 for this address. For example, storing a value of $E5 will set Port 0 for 9600 baud (DCBA = 1110) and Port 1 for 300 baud (DCBA = 0101). If one Port is selected for manual baudrate, the corresponding bits of the byte written to the baudrate address become "don't cares".



K18

1. U56-8 PORT 0 SOFTWARE BAUD RATE ENABLE
2. U10-10 SOFTWARE BAUD RATE STROBE
3. U56-12 PORT 1 SOFTWARE BAUD RATE ENABLE
4. U10-10 SOFTWARE BAUD RATE STROBE
5. U33-38 PIA 1 IRQ A
6. U33-37 PIA 1 IRQ B

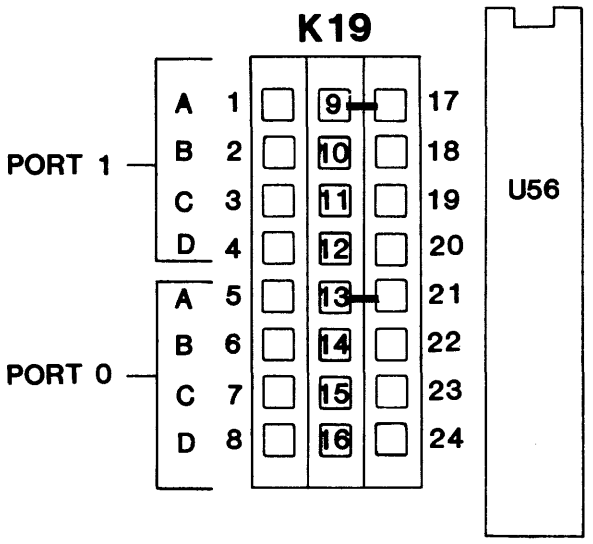


K19

1. D8	9. U56-16 PORT 1 BIT A
2. D9	10. U56-15 PORT 1 BIT B
3. D10	11. U56-14 PORT 1 BIT C
4. D11	12. U56-13 PORT 1 BIT D
5. D12	13. U56-4 PORT 0 BIT A
6. D13	14. U56-5 PORT 0 BIT B
7. D14	15. U56-6 PORT 0 BIT C
8. D15	16. U56-7 PORT 0 BIT D
	17.-24. GND

SERIAL PORT BAUD RATE JUMPERS
FIGURE 4.1.2

D C B A	BAUD RATE
0000	50
0001	75
0010	110
0011	134.5
0100	150
0101	300
0110	600
0111	1200
1000	1800
1001	2000
1010	2400
1011	3600
1100	4800
1101	7200
1110	9600
1111	19200



EXAMPLE ABOVE ILLUSTRATES HARDWARE BAUD RATE OF 9600 FOR BOTH PORTS (FACTORY STANDARD).

INSTALLED JUMPERS = 0
REMOVED JUMPERS = 1

BAUD RATE SELECTION
TABLE 4.1.2.1

4.2 Bus Error Jumper (K6)

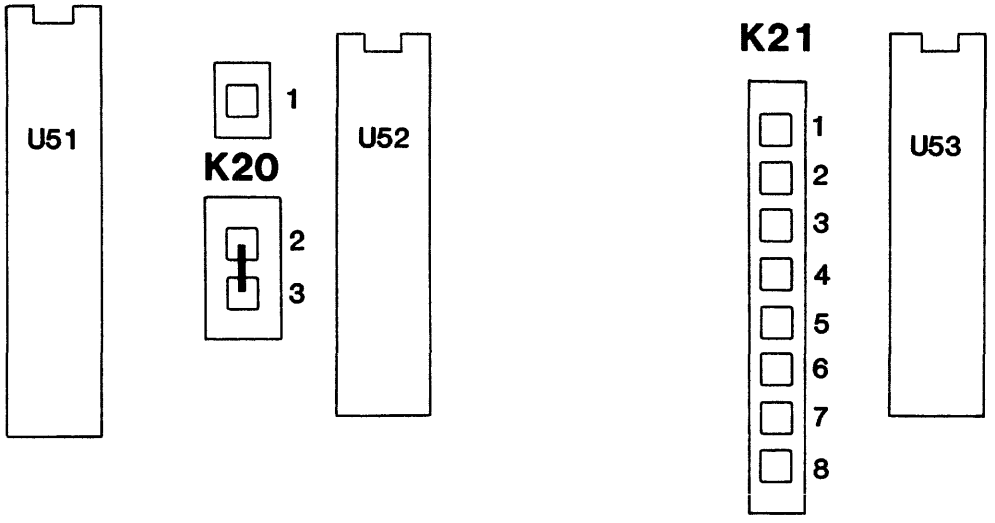
Jumper K6 connects the output of the watchdog timer to an encoder that operates the BERR and HALT lines of the 68000. When K6-2 to K6-3 is installed, BERR enabled, and DTACK is not received from an addressed device, a bus error is generated and the processor will begin bus error exception processing. With no jumper installed, the address and data lines will be static and can be examined at leisure. K6-1 and K6-2 will cause the processor to repeatedly rerun the cycle until DTACK is received. This is a useful mode for debugging off board system problems. Note that this feature is possible because the hardware refresh circuitry continues to operate even though the processor is stopped. See Figures 4.0 and 4.13 for location of jumper K6.

4.3 DTACK Select (K20,K21)

The DTACK generator for the ROM consists of a 74164 shift register that is held in its cleared state until either LDS or UDS is asserted. Logic ones are clocked through the shift register by the 10MHz (processor) oscillator, and DTACK timing is selected by choosing the stage of the shift register connected to the DTACK input pin. ***DTACK is factory set for 350ns ROM chips. If faster ROM is used, the DTACK jumper can be moved to provide a shorter DTACK delay.*** Each jumper represents an increment of 100ns., K21-1 being the shortest and K21-5 being the longest DTACK delay. (See Figure 4.3) Notice that the DTACK timer is active only for on-board ROM access. For off-board accesses DTACK is derived from the Multibus XACK signal. For on-board RAM accesses DTACK is derived from the RAM controller circuitry. Normal onboard RAM access will incur no wait states. See Table 4.3 below. Jumper K20-3 is the ROM delay input pin. This pin may be connected to various delayed outputs of the DTACK generator via the K21 jumper group. For convenience, K21-2 is routed to K20-2 to become the factory standard ROM DTACK delay. A jumper connecting K20-2 and K20-3 provides a ROM DTACK to operate with 350ns ROMS. For other ROM speeds K20-3 may be connected to other DTACK generator outputs at K21 according to the delay time given in Table 4.3. If 150ns or faster ROMS are used, no wait states are required. For zero wait state operation for ROM memory accesses, connect K20-3 to K20-1.

Jumper K20-3 Connect to	Maximum ROM ACCESS TIME	No. of Wait States
K20-1	155	0
K21-1	255	2
K20-2/K21-2	355	4
K21-3	455	6
K21-4	555	8
K21-8	655	10
K21-7	755	12
K21-6	855	14
K21-5	955	16

ROM DTACK DELAYS
TABLE 4.3



K20

1. U51-19 0 ROM WAIT STATES TAP
2. U53-4 4 ROM WAIT STATES TAP
3. U51-1 ROM DTACK INPUT

2-3 FACTORY STANDARD

K21

1. U53-3 2 ROM WAIT STATES TAP
2. U53-4 4 ROM WAIT STATES TAP
3. U53-5 6 ROM WAIT STATES TAP
4. U53-6 8 ROM WAIT STATES TAP
5. U53-13 16 ROM WAIT STATES TAP
6. U53-12 14 ROM WAIT STATES TAP
7. U53-11 12 ROM WAIT STATES TAP
8. U53-10 10 ROM WAIT STATES TAP

ROM DTACK DELAY JUMPERS
FIGURE 4.3

4.4 Interrupt Priority (K2)

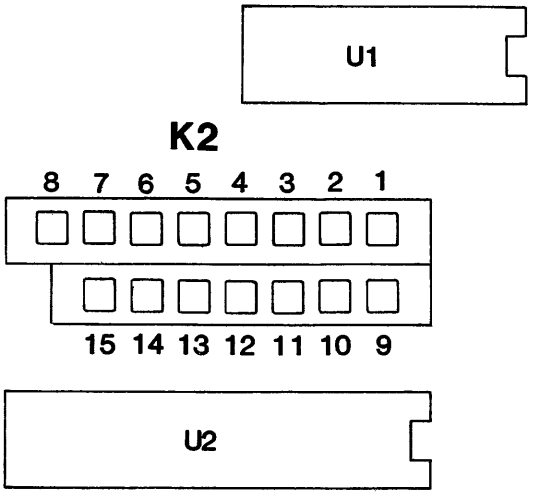
Interrupts for on-board peripheral chips are terminated on wire wrap posts so that the user may select the interrupts to be used and the priority level of each interrupt. The following interrupts from the onboard peripheral chips are available:

INTERRUPT SOURCE	JUMPER PIN NO.
Port 0 ACIA	K15-11
Port 1 ACIA	K15-12
Timer IRQ	K16-8
PIA 0 IRQ A	K16-9
PIA 0 IRQ B	K16-10
PIA 1 IRQ A	K18-5
PIA 1 IRQ B	K18-6

Because Motorola IRQ lines are open drain connections, the interrupt requests from several chips may be wire 0Red to occupy a single priority level.

The Multibus interrupt lines (8) are brought to on-board wire wrap posts. These may also be wire 0Red and connected to the priority encoder inputs. Both the Multibus interrupts and the inputs to the priority encoder are located in jumper group K2. See Figure 4.4.

Priority encoder input seven is the highest priority; one is the lowest. Interrupt requests are not latched on-board. These interrupt requests must be held active until serviced when they are reset by the processor during execution of the interrupt service routine as indicated by the IEEE-796 specification.



K2

P1 CONNECTOR (BACKPLANE)		PROCESSOR (INT PRIORITY CHIP)	
1. P1-37	INT 4	9. U2-1	LEVEL 4 INPUT
2. P1-38	INT 5	10. U2-2	LEVEL 5 INPUT
3. P1-35	INT 6	11. U2-3	LEVEL 6 INPUT
4. P1-36	INT 7	12. U2-4	LEVEL 7 INPUT
5. P1-40	INT 3		(NMI)
6. P1-39	INT 2	13. U2-13	LEVEL 3 INPUT
7. P1-42	INT 1	14. U2-12	LEVEL 2 INPUT
8. P1-41	INT 0	15. U2-11	LEVEL 1 INPUT

INTERRUPT JUMPERS
FIGURE 4.4

4.5 CCLK and BCLK (K14, K4)

Jumpers K14 and K4 connect the on-board 10 MHz oscillator to the Multibus CCLK and BCLK lines, respectively. For a single processor system these jumpers should be left in place. For multi-processor systems, only one master should drive the Multibus clock lines. Removing these jumpers will remove the CCLK and BCLK signals from the bus. See Figures 4.0 and 4.13 for location of K14 and K4.

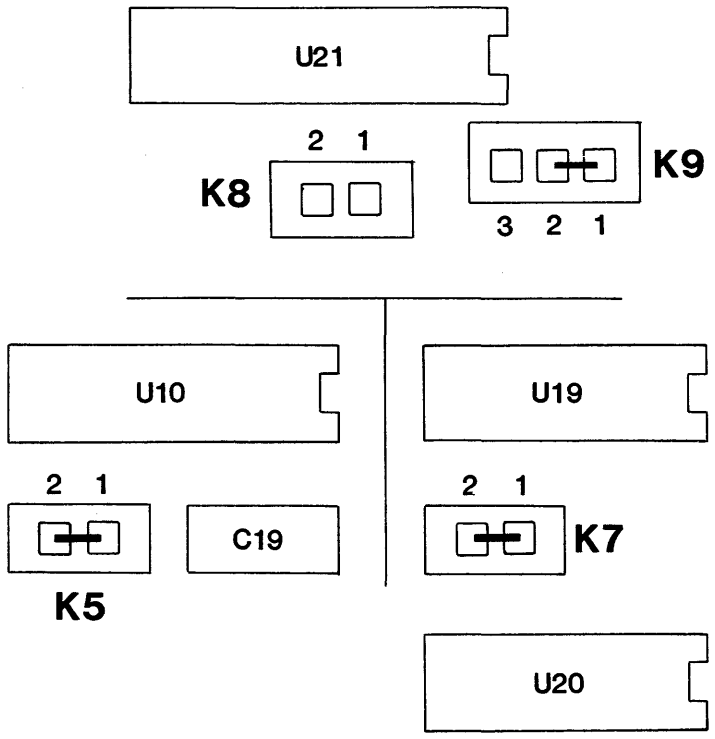
4.6 Bus Arbitration (K5, K7, K8, K9)

Four jumpers are associated with the bus priority logic. K5 connects the output of the bus arbitration logic to the Multibus BPRO* signal. The trace at K7 connects the BPRN* line to ground. In a system with a single bus master, K7 should be in place. In a Multibus system using serial priority, BPRN* is driven by the BPRO* signal from the master with the next highest priority. For serial operation, K5 should be installed and K7 should be installed for single master or highest priority board in a multimaster configuration.

Common Bus Request (CBRQ) is a signal that alerts a Multibus Master that another Master needs to use the bus. As shown in Figure 4.6, if K9 is jumpered to connect K9-1 to K9-2, CBRQ will be implemented by the bus arbitration circuitry. In this mode, the OB68K1A will retain ownership of the bus until it is forced to arbitrate with a higher priority user or until a lower priority user asserts CBRQ. If another Multibus is not capable of asserting CBRQ the OB68K1A, will relinquish the bus upon the negation of BPRN and completion of the present bus cycle. This allows the OB68K1A to maintain increased offboard throughput using its CBRQ feature. If K9-2 and K9-3 are connected, the OB68K1A will relinquish the bus between each cycle.

When parallel arbitration is used the OB68K1A requests the use of the bus by asserting BREQ. For this mode of arbitration, K8 should be installed. K8 should be removed for serial arbitration.

* indicates low active



K5

1. P1-16 (BPRO)
2. U3-8

JUMPER IN: SERIAL MULTIMASTER PRIORITIZATION

JUMPER OUT: PARALLEL MULTIMASTER PRIORITIZATION

K8

1. P1-1 (BREQ)
2. U21-15

JUMPER IN: PARALLEL ARBITRATION

JUMPER OUT: SERIAL ARBITRATION

K7

1. GND
2. P1-15 (BPRN)

JUMPER IN: SINGLE MASTER OR HIGHEST PRIORITY BOARD IN MULTIMASTER SYSTEM WITH SERIAL ARBITRATION

JUMPER OUT: ALL OTHER CONFIGURATIONS

K9

1. P1-29 (CBRQ)
2. U21-3
3. GND

1-2: BOARD MAINTAINS BUS CONTROL UNTIL ANOTHER MASTER REQUEST BUS

2-3: BOARD RELEASES BUS CONTROL AFTER EACH TRANSFER

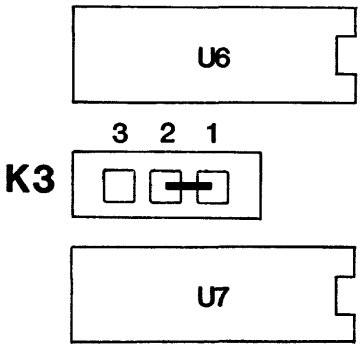
BUS ARBITRATION JUMPER CONFIGURATION
FIGURE 4.6

4.7 Initialize (K3)

The master in a single master system should drive the Multibus INIT line. For this type of operation K3-1 should be connected to K3-2 (See Figure 4.7). Both the power-on reset and software generated resets will drive the Multibus INIT line (Factory Standard).

For a Multi-Master system, it may be desirable for INIT to be an input to the OB68K1A board. This is done by connecting K3-2 to K3-3 so that the INIT signal will trigger the onboard reset generator.

For some applications it may be necessary for the OB68K1A to be reset directly by the INIT signal in order to synchronize the restart sequence of multiple processors. To use INIT directly, cut the two traces Z1-Z3 and Z2-Z4. Connect Pin K3-2 to both Z3 and Z4. (See Figure 4.13). This modification disables the onboard reset generator and the RESET push button will not be operative. The Master driving INIT must meet the power-on timing requirements of the 68000 processor in order to insure a valid restart sequence.



K3

1: U4-8 RESET OUTPUT
2. P1-14 INIT
3. U1-2 RESET GENERATOR INPUT

1-2: PROCESSOR AND ONBOARD INIT CIRCUITRY DRIVE MULTIBUS INIT.

2-3: MULTIBUS INIT TRIGGERS ONBOARD INIT CIRCUITRY.

RESET JUMPER CONFIGURATION
FIGURE 4.7

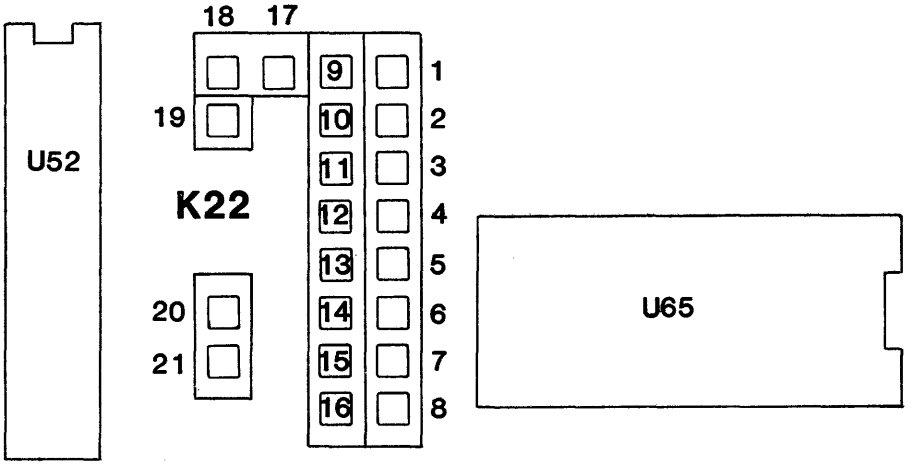
4.8 ROM Socket Configuration (K22)

Jumper area K22 configures the ROM sockets for the ROM size that is chosen. Rather than wire wrapping this group of pins, a small configurator board is plugged in to make all the connections simultaneously. The OB68K1A is shipped with a configurator for 8K (2764's) byte PROMs, but other configuration boards are available from the factory.

4.8.1 ROM Size Jumper Configuration

Figure 4.8.1 shows the jumper group K22 and identifies each of the pins. The user may choose to wire wrap K22 to field modify the size or type of ROM used on the OB68K1A.

Normally the user will use the factory supplied configuration plug to interconnect the K22 jumper pins. If a different PROM type is to be used, the user may reconnect K22 as shown in Figure 4.8.2. This figure shows the printed circuit artwork for the factory supplied plugs. The user may choose to fabricate plugs or purchase them from Omnibyte. The ROM sockets used on the OB68K1A have 28 pins. This is to provide for operation with certain types of 32K and 64K ROM chips. For operating with the 24 pin devices, ROM's MUST be installed with the unused pins on the right side of the socket (left justified). See Figure 4.8.3 for ROM socket configuration and Figure 4.8.4 for compatible chip pinout configuration. See section 9.0 for ordering information.

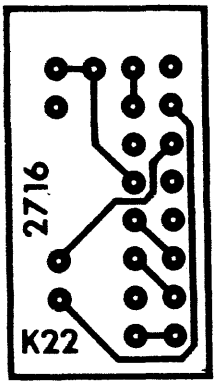


K22

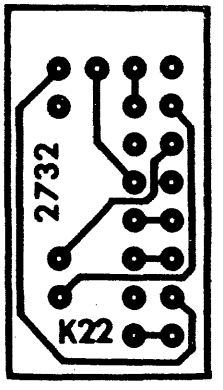
1. A 18
2. A 17
3. A 16
4. A 15
5. A 14
6. A 13
7. A 12
8. A 11
9. GND
10. ROM PIN 22
11. ROM PIN 27
12. +5V
13. U52-2
14. U52-1
15. ROM PIN 2
16. ROM PIN 21
17. ROM PIN 26
18. ROM PIN 23
19. GND
20. U30-3
21. U30-13

NOTE: SEE FIGURE 4.8.2 FOR FACTORY CONFIGURATION

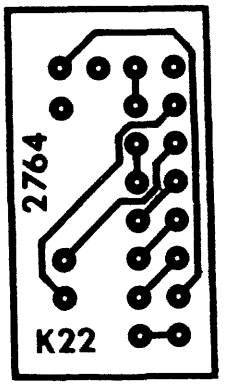
ROM SIZE JUMPER CONFIGURATION AND LOCATION
FIGURE 4.8.1



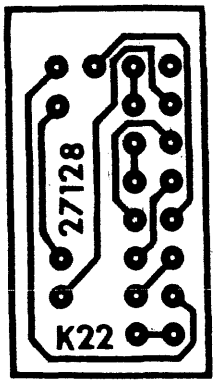
OBK1A/K22-2716



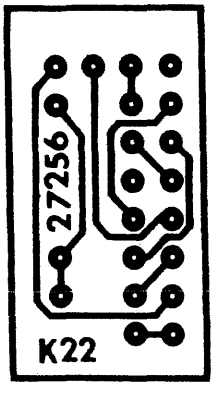
OBK1A/K22-2732



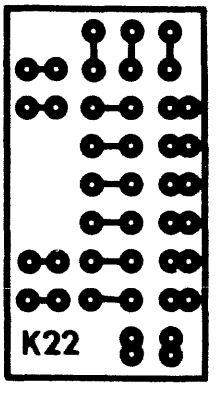
OBK1A/K22-2764
(FACTORY STANDARD)



OBK1A/K22-27128

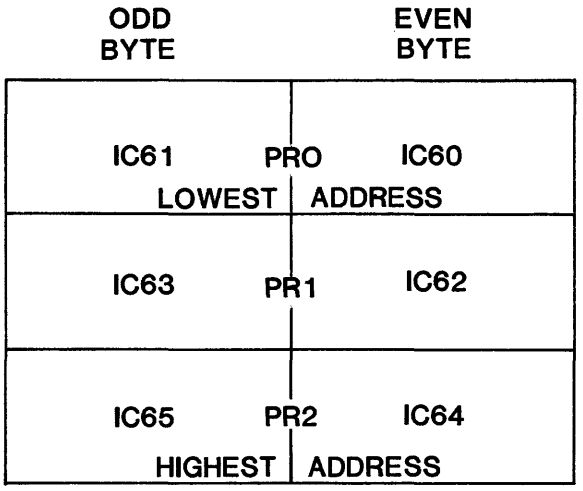


OBK1A/K22-27256



OBK1A/K22-UD
(USER DEFINABLE)

ROM CONFIGURATION PLUGS LAYOUTS
FIGURE 4.8.2



ROM SOCKET CONFIGURATION
FIGURE 4.8.3

27256	27128	2764	2732	2716	PIN	PIN	27256	27128	2764	2732	2716
Vpp	Vpp	Vpp			1	28	Vcc	Vcc	Vcc		
A12	A12	A12			2	27	A14	$\overline{PGM}$	$\overline{PGM}$		
A7	A7	A7	A7	A7	3	26	A13	A13	N/C	Vcc	Vcc
A6	A6	A6	A6	A6	4	25	A8	A8	A8	A8	A8
A5	A5	A5	A5	A5	5	24	A9	A9	A9	A9	A9
A4	A4	A4	A4	A4	6	23	A11	A11	A11	A11	Vpp
A3	A3	A3	A3	A3	7	22	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$/Vpp	$\overline{OE}$
A2	A2	A2	A2	A2	8	21	A10	A10	A10	A10	A10
A1	A1	A1	A1	A1	9	20	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
A0	A0	A0	A0	A0	10	19	O7	O7	O7	O7	O7
O0	O0	O0	O0	O0	11	18	O6	O6	O6	O6	O6
O1	O1	O1	O1	O1	12	17	O5	O5	O5	O5	O5
O2	O2	O2	O2	O2	13	16	O4	O4	O4	O4	O4
Gnd	Gnd	Gnd	Gnd	Gnd	14	15	O3	O3	O3	O3	O3

PIN NAMES	
A0 - A12	ADDRESS LINES
O0-O7	DATE LINES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
$\overline{PGM}$	PROGRAM

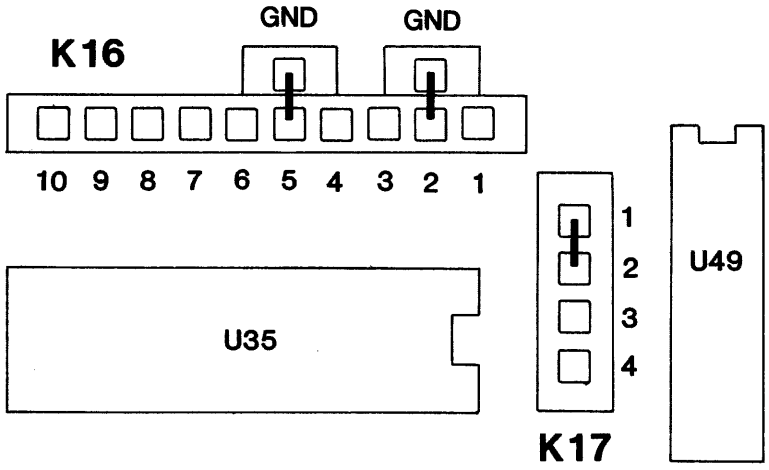
ROM CHIP PINOUT CONFIGURATION
FIGURE 4.8.4

4.9 Timer (K16, K17)

The timer (MC6840) has a clock, a gate and an output for each of its three timing channels. All nine of these signals are terminated on wire wrap pins in jumper groups K16 and K17. To operate the MC6840 timer channels, the gate signals must be low.

A trace on the board connects each of the three gate signals to ground. A trace must be cut for the appropriate channel if the gate is to be operated from an active source.

To use the timer as a programmable interrupt generator for the CPU, no external clock or gate connections are needed. The MC6840 may use the 1 MHz "E" clock as a time base. The timer IRQ is usually connected to a selected priority interrupt level. See Figure 4.9 for the PTM jumper pad locations.



K16

1. GND
2. U35-2 GATE 2
3. U35-3 OUTPUT 2
4. U35-4 CLOCK 2
5. U35-5 GATE 3
6. U35-6 OUTPUT 3
7. U35-7 CLOCK 3
8. U35-9 TIMER IRQ
9. U32-38 PIA 0 IRQ A
10. U32-37 PIA 1 IRQ B

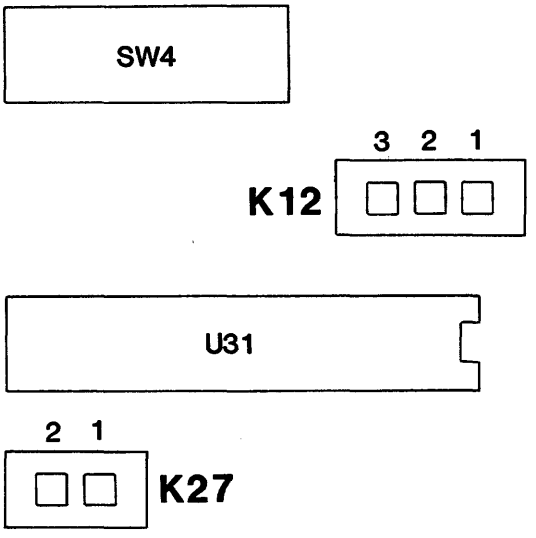
K17

1. GND
2. U35-26 GATE 1
3. U35-27 OUTPUT 1
4. U35-28 CLOCK 1

TIMER OPTION PIN IDENTIFICATION AND LOCATION
FIGURE 4.9

4.10 External RAM Access (K12, K27)

Switch SW4 is used to set the base address of the dual-port RAM for offboard access. Each OB68K1A is set to allow offboard access to the full onboard RAM. For the 128K version, the offboard access can be limited to 64K bytes by installing jumper K27. The external access can be further restricted to a 32K block of memory using jumper K12. The upper or lower half of the switch selected 64K base address is chosen by connecting K12-1 to K12-2 or K12-2 to K12-3 respectively. Jumper K12 provides the same function for 32K versions of the OB68K1A so that the external access can be made to appear at any 32K byte boundary. Address lines A19-A23 have been pulled to logic 0 to accommodate systems that do not implement the full 16M-byte address. Please note that the 128K RAM starts on 128K byte boundaries even if, a lesser amount is made accessible to offboard access, ie. upper or lower 64K byte blocks.



K12

1. P1-44 A15 INVERTED = 1
2. COMPARATOR ENABLE
3. P1-44 A15 NORMAL = 0

2-3: 32K RAM (FACTORY STANDARD FOR OB68K1A-32K)

NO JUMPER: 128K RAM (FACTORY STANDARD FOR OB68K1A-128K)

K27

1. U31-13
2. P1-28 A16

NO JUMPER: 32K/128K RAM BOARDS (FACTORY STANDARD)

JUMPER IN W/128K: 64K EXTERNAL RAM ACCESS LIMIT

JUMPER IN W/32K: NO EFFECT

NOTE: WITH THE 128K BOARD K27 IS NOT JUMPERED AND PINS 13 AND 14 ON U31 (ADDRESS COMPARATOR) ARE TIED TOGETHER. THIS MAKES POSITION 1 ON SWITCH 4 A DON'T CARE BIT AND REDUCES POSSIBLE ERRORS.

EXTERNAL RAM ACCESS SIZE JUMPERS
FIGURE 4.10

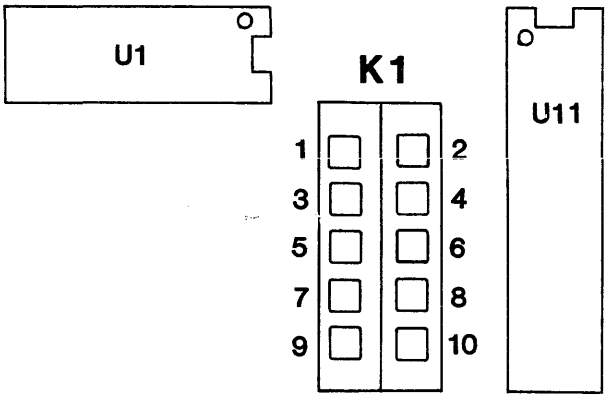
4.11 Watchdog Timer for External RAM Access (K23)

In the case of a malfunction of an external Master, an external RAM access could fail to terminate a command even after the XACK was generated. If this condition would persist for many microseconds, the contents of the onboard RAM could be destroyed. Logic is included in the dual port arbitration circuitry to negate the external request to the onboard RAM after 2.5 micro seconds. Once the access is terminated by the arbitration circuitry the board is allowed to resume normal processing; ie a processor cycle or RAM refresh. The external RAM access, if still present, is treated as a new request by the arbitration circuit and this cycle is repeated. The board is shipped with the watchdog timer enabled, but this feature may be defeated by installing jumper K23. See Figure 4.13.

4.12 Optional Front Panel (K1)

Jumper group K1 is provided as a means of connecting a front panel to the OB68K1A. Using the K1 signals, a front panel can be implemented that includes a RUN/HALT switch, RESET, SINGLE STEP and NMI pushbuttons. The required external circuitry is shown in Figure 4.12 (B). The NMI (non-maskable interrupt) may be used as a software abort button. This front panel requires only one quad 2-input NAND gate. Power for the front panel is included in the K1 jumper group.

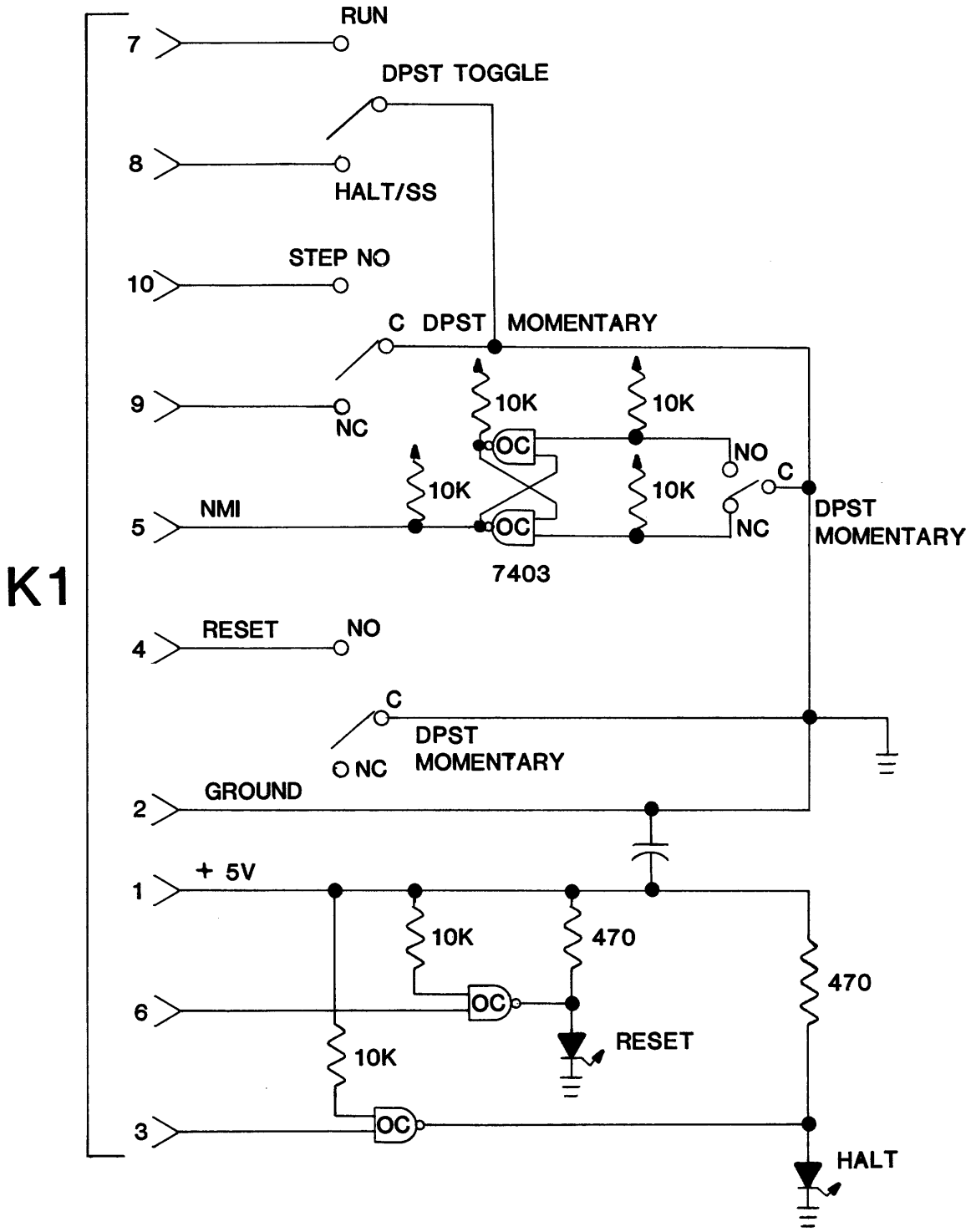
The front panel circuit of Figure 4.12 (B) is available as an option from Omnibyte. This is a small box with a 5-foot interconnecting cable terminated in a connector that plugs into jumper group K1. The ordering information is given in Section 9.0.



K1

1. +5V
2. GND
3. HALT LED
4. MANUAL RESET
5. NMI INPUT
6. RESET LED
7. RUN
8. HALT/SS MODE
9. SINGLE STEP PUSH BUTTON N.C.
10. SINGLE STEP PUSH BUTTON N.O.

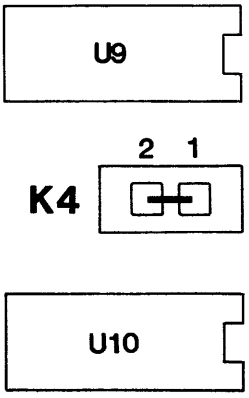
OPTIONAL FRONT PANEL - CONNECTOR
FIGURE 4.12 (A)



OPTIONAL FRONT PANEL - CIRCUIT
FIGURE 4.12 (B)

4.13 Miscellaneous Jumper Identification

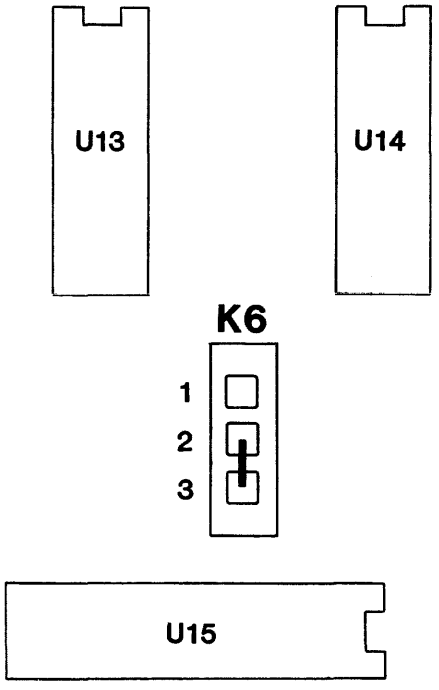
Several jumper options have been included on the OB68K1A for flexibility and future enhancement. Although these jumpers are not normally changed by the user, they are documented in Figure 4.13 for completeness. The Figure serves to define the physical location of specific jumper pins that appear on the OB68K1A electrical schematic diagram. Those jumper groups not documented elsewhere in the text are included in Figure 4.13.



K4

BUS CLOCK (BCLK) SOURCE TO MULTIBUS (P1-13)

NOTE: IN MULTIMASTER SYSTEMS, ONLY ONE BCLK SOURCE PER SYSTEM.



K6

BUS ERROR (BERR)

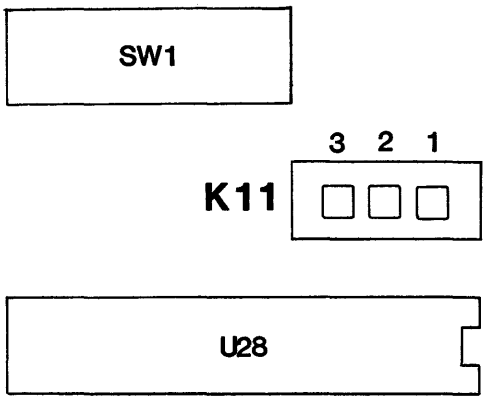
JUMPER 2-3: BERR WATCHDOG TIMER CIRCUITRY ENABLED.

NO JUMPER: BERR WATCHDOG TIMER CIRCUITRY DISABLED.

JUMPER 1-2: CYCLE RETRY (USED FOR FACTORY TESTING.)

NOTE: BERR DISABLED IS GENERALLY USED FOR TROUBLESHOOTING AND IS NOT A DESIRABLE MODE TO RUN AS ONE COULD GET INTO A STATE THAT WOULD REQUIRE A RESET.

MISCELLANEOUS JUMPERS LOCATIONS
FIGURE 4.13

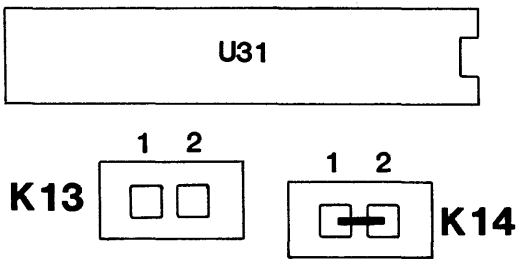


K11

ONBOARD RAM SIZE

1-2 32K RAM
2-3 128K RAM

NOTE: FACTORY CONFIGURED, GENERALLY NOT CHANGED BY USER, JUMPER MUST BE IN PLACE TO ACCESS RAM.



K13

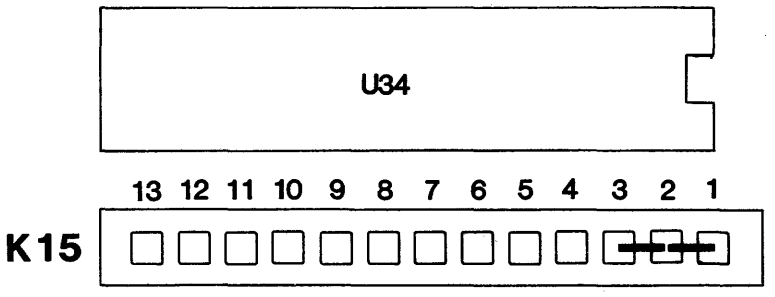
JUMPER IN: CONNECTS A18 TO ADDRESS COMPARATOR FOR ONBOARD RAM.

NOTE: K13 IS SUPPLIED FOR FUTURE ONBOARD RAM ENHANCEMENT.

K14

CONSTANT CLOCK (CCLK) TO MULTIBUS (P1-31)

NOTE: IN MULTIMASTER SYSTEMS,ONLY ONE CCLK SOURCE IS PERMITEED.



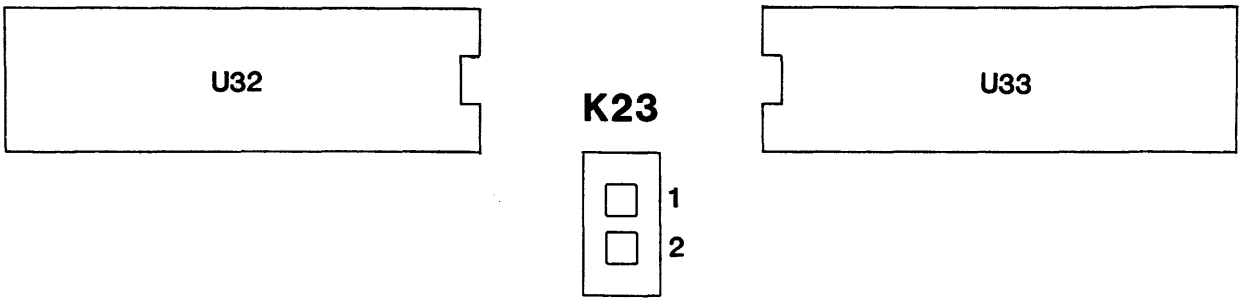
U36

K15

1. U23-11 (DTR)
2. U34-24 (CTS)
3. U34-23 (DCD)
4. GND
5. TEST POINT 2 (TP2) HALT
6. TEST POINT 1 (TP1) RESET
7. TEST POINT 3 (TP3) BERR
8. TEST POINT 4 (TP4) IPL 2
9. TEST POINT 5 (TP5) IPL 1
10. TEST POINT 6 (TP6) IPL 0
11. U34-7 PORT 0 IRQ
12. U26-7 PORT 1 IRQ
13. SPARE RS232 RECEIVER BUFFER INPUT (OUTPUT ON K31; TTL)

NOTE: K-15 - 1,2,3 CONNECTION MADE VIA CIRCUIT BOARD TRACES.

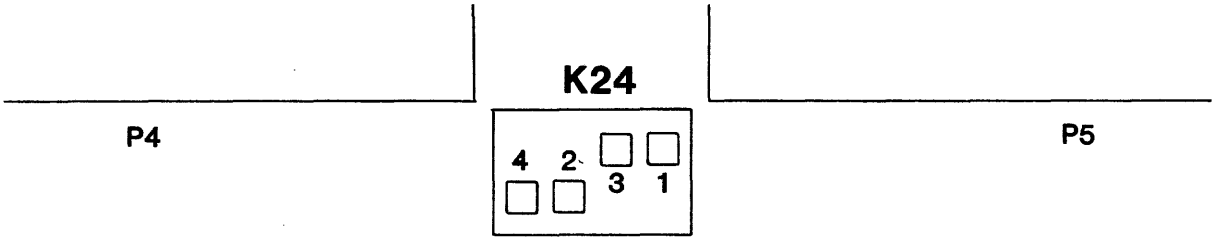
MISCELLANEOUS JUMPERS LOCATIONS - CONT.
FIGURE 4.13



K23

JUMPER OUT: ENABLES EXTERNAL RAM ACCESS WATCHDOG TIMER CIRCUITRY.

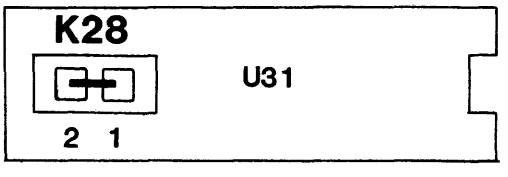
JUMPER IN : DISABLES EXTERNAL RAM ACCESS WATCHDOG TIMER CIRCUITRY.



K24

± 12V TO PORT 0
(FOR EXTERNAL USE)

1. −12V
2. P4-18
3. +12V
4. P4-20

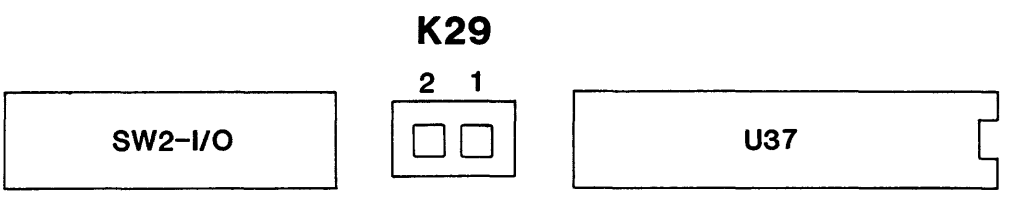


K28

CONNECTS A17 TO ONBOARD RAM ACCESS COMPARATOR.

NOTE: K28 IS SUPPLIED FOR FUTURE RAM ENHANCEMENT.

MISCELLANEOUS JUMPERS LOCATIONS - CONT.
FIGURE 4.13

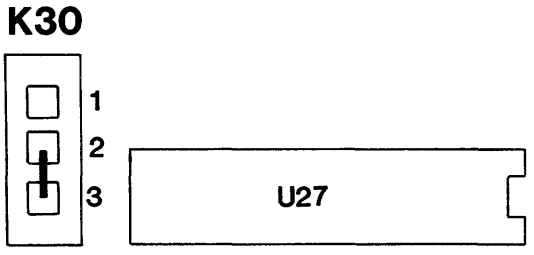


K29

CONNECTS A16 TO ONBOARD RAM ADDRESS COMPARATOR

JUMPER IN: 32K RAM
JUMPER OUT: 128K RAM

NOTE: ON 128K RAM BOARDS PINS 3 AND 4 ON U28 ARE TIED TOGETHER, THUS MAKING POSITION 1 ON SWITCH 1 A DON'T CARE BIT AND REDUCES POSSIBLE ERRORS.

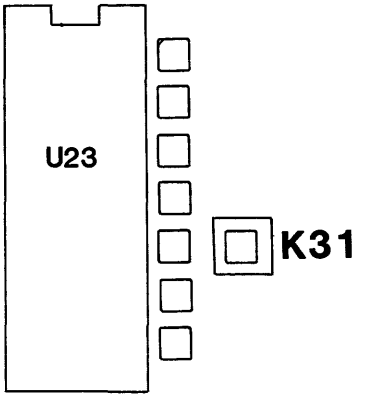


K30

INVERTS SIGN OF DTR ON PORT 1

2-3 NORMAL (FACTORY STANDARD)

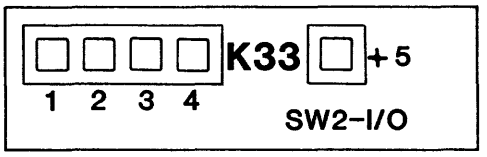
1-3 INVERTED



K31

SPARE RS232 RECEIVER BUFFER OUTPUT (INPUT ON K15-13; TTL)

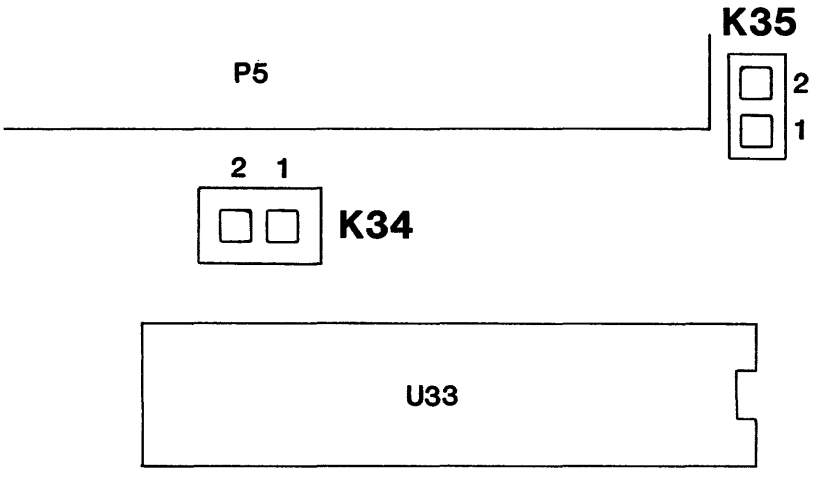
MISCELLANEOUS JUMPERS LOCATIONS - CONT.
FIGURE 4.13



K33

CONNECTS A17 (1-2) AND A18 (3-4) TO ONBOARD RAM ADDRESS COMPARATOR.

NOTE: K33 IS SUPPLIED FOR FUTURE ONBOARD RAM ENHANCEMENT.



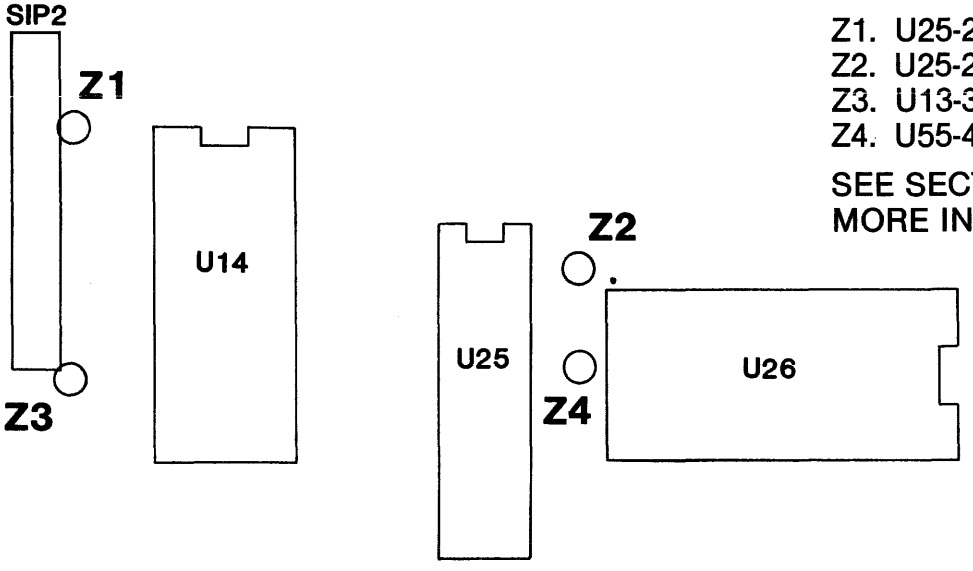
K34

(FOR EXTERNAL USE)
+5V FOR PIA 0

1. P5-22, 23
2. +5V.

K35
(FOR EXTERNAL USE)
+5V FOR PIA 1

1. +5V
2. P5-47, 48



CUT TRACE OPTIONS

Z1. U25-2 (555 OUTPUT)
Z2. U25-2 (555 OUTPUT)
Z3. U13-3 (CLR)
Z4. U55-4

SEE SECTION 4.7 FOR MORE INFORMATION

MISCELLANEOUS JUMPERS OPTIONS - CONT.
FIGURE 4.13

4.14 System Configuration

OB68K1A boards that have been ordered in conjunction with, the OB68K/SYS Development System or, the three board Functional Board Pack, have been modified from the standard factory configuration stated earlier in this manual.

Port 1 has been reconfigured as Data Communications Equipment (DCE), so that a terminal can be connected to Port 1. + 12V has also been brought out to the Port 1 connector and a TI810* printer can be connected to the Port 1. Several interrupts have also been connected. Listed below are details of the modifications:

Port 1 Modifications

Jumper Group K26

cut	3-4	connect	3-8
cut	5-6	connect	4-7
cut	7-8	connect	6-9
cut	9-10		

Jumper Group K10

cut	1-2	connect	K10-2 to K25-19

P3 connector

Pin 11 to Pin 15		connect

INTERRUPTS

K2-5 to K2-13	wire-wrap	Multibus INT 3 to Level 3 INT input
K2-11 to K15-11	wire-wrap	Port 0 IRQ to Level 16 INT input
K2-10 to K15-12	wire-wrap	Port 1 IRQ to Level 5 INT input
K2-9 to K16-8	wire-wrap	Timer IRQ to Level 14 INT input
Anode Side of D6 to K2-8	jumper with IN916-anode toward D6	Isolates INT 0 from circuitry of 555 during power-on RESET

5.0 CONNECTOR PINOUTS

5.1 Multibus P1 and P2 Connectors

P1 and P2 are the Multibus connectors and are pinned according to the IEEE 796 specification. See Tables 5.1.1 & 5.1.2 respectively for the pin assignments.

*TI810 is a trademark of Texas Instruments

	(Component Side)			(Circuit Side)		
	Pin	Mnemonic	Description	Pin	Mnemonic	Description
Power Supplies	1	GND	Signal GND	2	GND	Signal GND
	3	+ 5V	+ 5Vdc	4	+ 5V	+ 5Vdc
	5	+ 5V	+ 5Vdc	6	+ 5V	+ 5Vdc
	7	+ 12V	+ 12Vdc	8	+ 12V	+ 12Vdc
	9		Reserved, bussed	10		Reserved, bussed
	11	GND	Signal GND	12	GND	Signal GND
Bus Controls	13	BCLK*	Bus Clock	14	INIT*	Initialize
	15	BPRN*	Bus Pri. In	16	BPRO*	Bus Pri. Out
	17	BUSY*	Bus Busy	18	BREQ*	Bus Request
	19	MRDC*	Mem Read Cmd	20	MWTC*	Mem Write Cmd
	21	IORC*	I/O Read Cmd	22	IOWC*	I/O Write Cmd
	23	XACK*	XFER Acknowledge	24	INH1*	Inhibit 1 (disable RAM)
Bus Controls and Address	25	LOCK*	Lock	26	INH2*	Inhibit 2 (disable PROM or ROM)
	27	BHEN*	Byte High Enable	28	ADR16*	
	29	CBRQ*	Common Bus Request	30	ADR17*	Address
	31	CCLK*	Constant Clk	32	ADR18*	Bus
	33	INTA*	Intr Acknowledge	34	ADR19*	
Interrupts	35	INT6*	Parallel	36	INT7*	Parallel
	37	INT4*	Interrupt	38	INT5*	Interrupt
	39	INT2*	Requests	40	INT3*	Requests
	41	INT0*		42	INT1*	
Address	43	ADR14*		44	ADR15*	
	45	ADR12*		46	ADR13*	
	47	ADR10*	Address	48	ADR11*	Address
	49	ADR8*	Bus	50	ADR9*	Bus
	51	ADR6*		52	ADR7*	
	53	ADR4*		54	ADR5*	
	55	ADR2*		56	ADR3*	
	57	ADR0*		58	ADR1*	
Data	59	DATE*		60	DATF*	
	61	DATC*		62	DATD*	
	63	DATA*	Data	64	DATB*	Data
	65	DAT8*	Bus	66	DAT9*	Bus
	67	DAT6*		68	DAT7*	
	69	DAT4*		70	DAT5*	
	71	DAT2*		72	DAT3*	
	73	DAT0*		74	DAT1*	
Power Supplies	75	GND	Signal GND	76	GND	Signal GND
	77		Reserved, bussed	78		Reserved, bussed
	79	– 12V	– 12Vdc	80	– 12V	– 12Vdc
	81	+ 5V	+ 5Vdc	82	+ 5V	+ 5Vdc
	83	+ 5V	+ 5Vdc	84	+ 5V	+ 5Vdc
	85	GND	Signal GND	86	GND	Signal GND

All Reserved pins are reserved for future use and should not be used if upwards compatibility is desired.

IEEE 796 P1 CONNECTOR PINOUT
TABLE 5.1.1

	(Component Side)			(Circuit Side)		
	Pin	Mnemonic	Description	Pin	Mnemonic	Description
	1		Reserved, Not Bussed	2		Reserved, Not Bussed
	3		Reserved, Not Bussed	4		Reserved, Not Bussed
	5		Reserved, Not Bussed	6		Reserved, Not Bussed
	7		Reserved, Not Bussed	8		Reserved, Not Bussed
	9		Reserved, Not Bussed	10		Reserved, Not Bussed
	11		Reserved, Not Bussed	12		Reserved, Not Bussed
	13		Reserved, Not Bussed	14		Reserved, Not Bussed
	15		Reserved, Not Bussed	16		Reserved, Not Bussed
	17		Reserved, Not Bussed	18		Reserved, Not Bussed
	19		Reserved, Not Bussed	20		Reserved, Not Bussed
	21		Reserved, Not Bussed	22		Reserved, Not Bussed
	23		Reserved, Not Bussed	24		Reserved, Not Bussed
	25		Reserved, Not Bussed	26		Reserved, Not Bussed
	27		Reserved, Not Bussed	28		Reserved, Not Bussed
	29		Reserved, Not Bussed	30		Reserved, Not Bussed
	31		Reserved, Not Bussed	32		Reserved, Not Bussed
	33		Reserved, Not Bussed	34		Reserved, Not Bussed
	35		Reserved, Not Bussed	36		Reserved, Not Bussed
	37		Reserved, Not Bussed	38		Reserved, Not Bussed
	39		Reserved, Not Bussed	40		Reserved, Not Bussed
	41		Reserved, Bussed	42		Reserved, Bussed
	43		Reserved, Bussed	44		Reserved, Bussed
	45		Reserved, Bussed	46		Reserved, Bussed
	47		Reserved, Bussed	48		Reserved, Bussed
	49		Reserved, Bussed	50		Reserved, Bussed
	51		Reserved, Bussed	52		Reserved, Bussed
	53		Reserved, Bussed	54		Reserved, Bussed
Address	55	ADR22*	Address Bus	56	ADR23*	Address Bus
	57	ADR20*		58	ADR21*	
	59		Reserved, Bussed	60		Reserved, Bussed

All Reserved Pins are reserved for future use and should not be used if upwards compatability is desired.

IEEE 796 P2 CONNECTOR PINOUT
TABLE 5.1.2

5.2 PIA and ACIA Connectors

The header connector pin for each signal of the PIA,s are shown in Table 5.2.1 below. The "D" Columns indicate the pin numbers of a 50 conductor ribbon cable is split into two 25 conductor sections for termination at the user end of the cable.

SIGNAL	PIN NUMBERS HEADER	*	*	HEADER	SIGNAL
PIA0 PA7	1	1	14	2	PIA0 CB2
PA6	3	2	15	4	CB1
PA5	5	3	16	6	PB7
PA4	7	4	17	8	PB6
PA3	9	5	18	10	PB5
PA2	11	6	19	12	PB4
PA1	13	7	20	14	PB3
PA0	15	8	21	16	PB2
CA2	17	9	22	18	PB1
CA1	19	10	23	20	PB0
NO CONN.	21	11	24	22	+ 5 VOLTS
+ 5 VOLTS	23	12	25	24	GROUND
GROUND	25	13	1	26	PIA1 PA7
PIA1 CB2	27	14	2	28	PA6
CB1	29	15	3	30	PA5
PB7	31	16	4	32	PA4
PB6	33	17	5	34	PA3
PB5	35	18	6	36	PA2
PB4	37	19	7	38	PA1
PB3	39	20	8	40	PA0
PB2	41	21	9	42	CA2
PB1	43	22	10	44	CA1
PB0	45	23	11	46	NO CONN.
+ 5 VOLTS	47	24	12	48	+ 5 VOLTS
GROUND	49	25	13	50	GROUND

PIA CONNECTOR PINOUT
TABLE 5.2.1

In Table 5.2.2 and 5.2.3 below, both the header connector pin and the "D" pin are shown for each signal on the ACIA connectors, because a 25 pin "D" connector is normally installed on the user end of the ribbon cable.

	PIN NUMBERS				
SIGNAL	HEADER	"D"	"D"	HEADER	SIGNAL
GROUND	1	1	14	2	
RX DATA	3	2	15	4	
TX DATA	5	3	16	6	
RTS	7	4	17	8	
CTS	9	5	18	10	
DSR	11	6	19	12	
GROUND	13	7	20	14	DTR
DCD	15	8	21	16	+ 5 VDC
GROUND	17	9	22	18	- 12 VDC
GROUND	19	10	23	20	+ 12 VDC
	21	11	24	22	
	23	12	25	24	
	25	13		26	

ACIA CONNECTOR (PORT 0) PINOUT (DCE)
TABLE 5.2.2

	PIN NUMBERS				
SIGNAL	HEADER	"D"	"D"	HEADER	SIGNAL
GROUND	1	1	14	2	
TX DATA	3	2	15	4	
RX DATA	5	3	16	6	
	7	4	17	8	
CTS	9	5	18	10	
	11	6	19	12	
GROUND	13	7	20	14	DTR
DCD	15	8	21	16	+ 5 VDC
GROUND	17	9	22	18	- 12 VDC
GROUND	19	10	23	20	+ 12 VDC
	21	11	24	22	
	23	12	25	24	
	25	13		26	

ACIA CONNECTOR (PORT 1) PINOUT (DTE)
TABLE 5.2.3

5.3 Compatible Cable End Connectors

Connector numbers that are compatible with the serial and parallel I/O connectors of the OB68K1A are given below. For completeness the numbers of the 25 pin "D" connectors are also listed.

Serial I / O

Cable connector for Computer end; 26 pin –
Berg # 65484–009
Ansley # 609–2631 & # 609–2630

Cable connector for Terminal/Modem end; 25 pin "D" –
Berg # 66167–025
Ansley # 609–25P

Parallel I / O

Cable Connector at Computer end; 50 pin –
Berg # 65484–021
Ansley # 609–5030 & # 609–5031

Cable Connector at User End, 25 pin "D" –
Berg # 66167–025
Ansley # 609–25P

Note that these are common ribbon cable connectors, and although the part numbers above are only for the Berg and T & B/Ansley parts, equivalent connectors are available from several other vendors.

For user convenience, Omnibyte can furnish assembled parallel and serial cables. Order numbers for these cables are given in Section 9.0. The parallel I/O cable is a 50 conductor cable terminated in a 50 contact connector at the computer end, split in half and terminated in two 25 socket "D" connectors on the user end. This arrangement results in two user connectors with the same pinout that contain all PIA 0 signals in one connector and all PIA 1 signals in the other connector.

6.0 MEMORY DECODING

The IEEE 796 bus specifications define 24 address lines on the backplane and the 68000 can support all 24 bits of address. The upper four bits are brought out to the P2 connector. This makes an available address space of 16 Megabytes. Memory allocation is programmable through the use of switches SW-4, SW-3, SW-2 and SW-1.

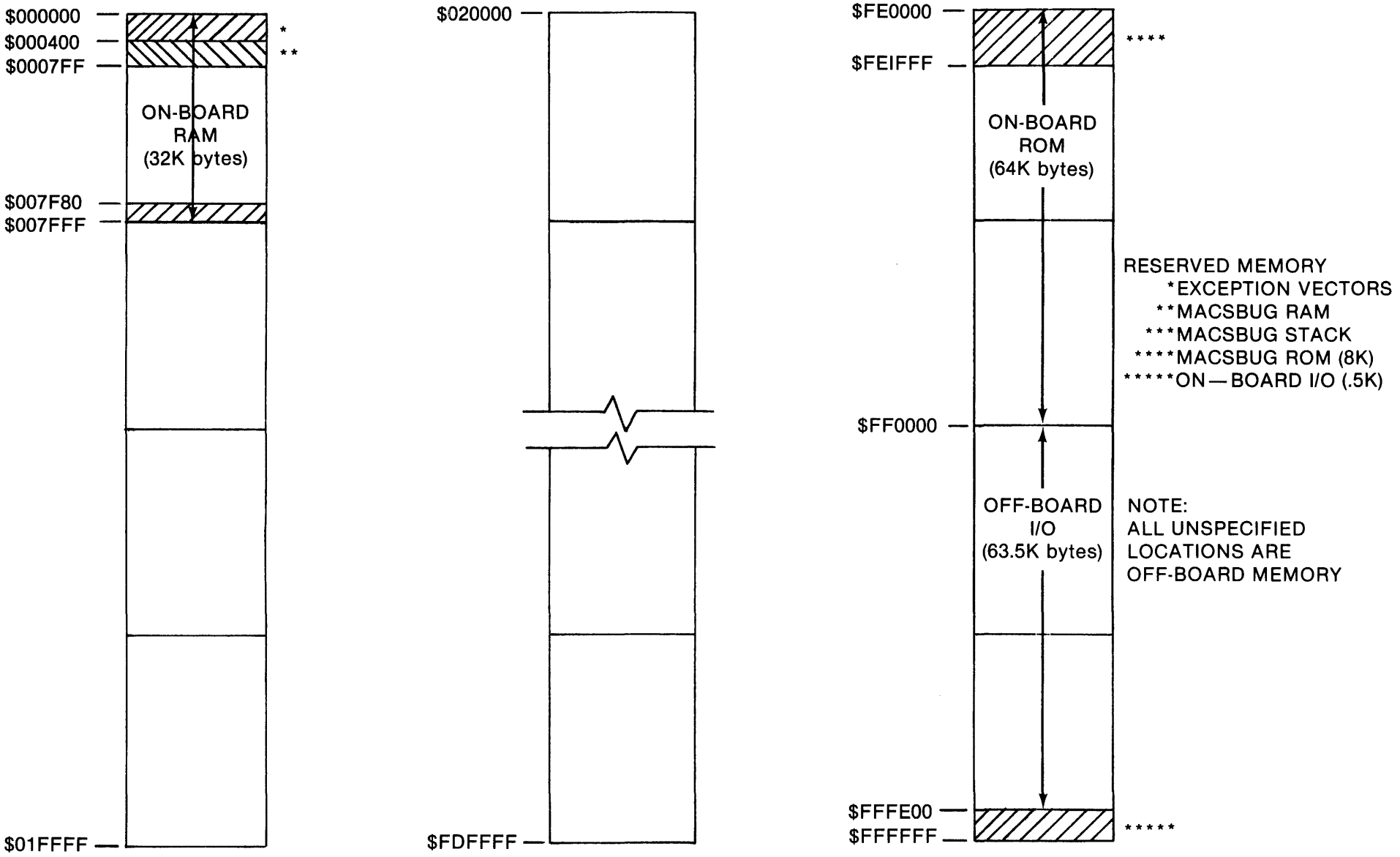
6.1 Memory Maps

The factory standard memory configuration (MAP 1) is shown in the diagrams of Figure 6.1.1 & 6.1.2. Switch settings for this map are:

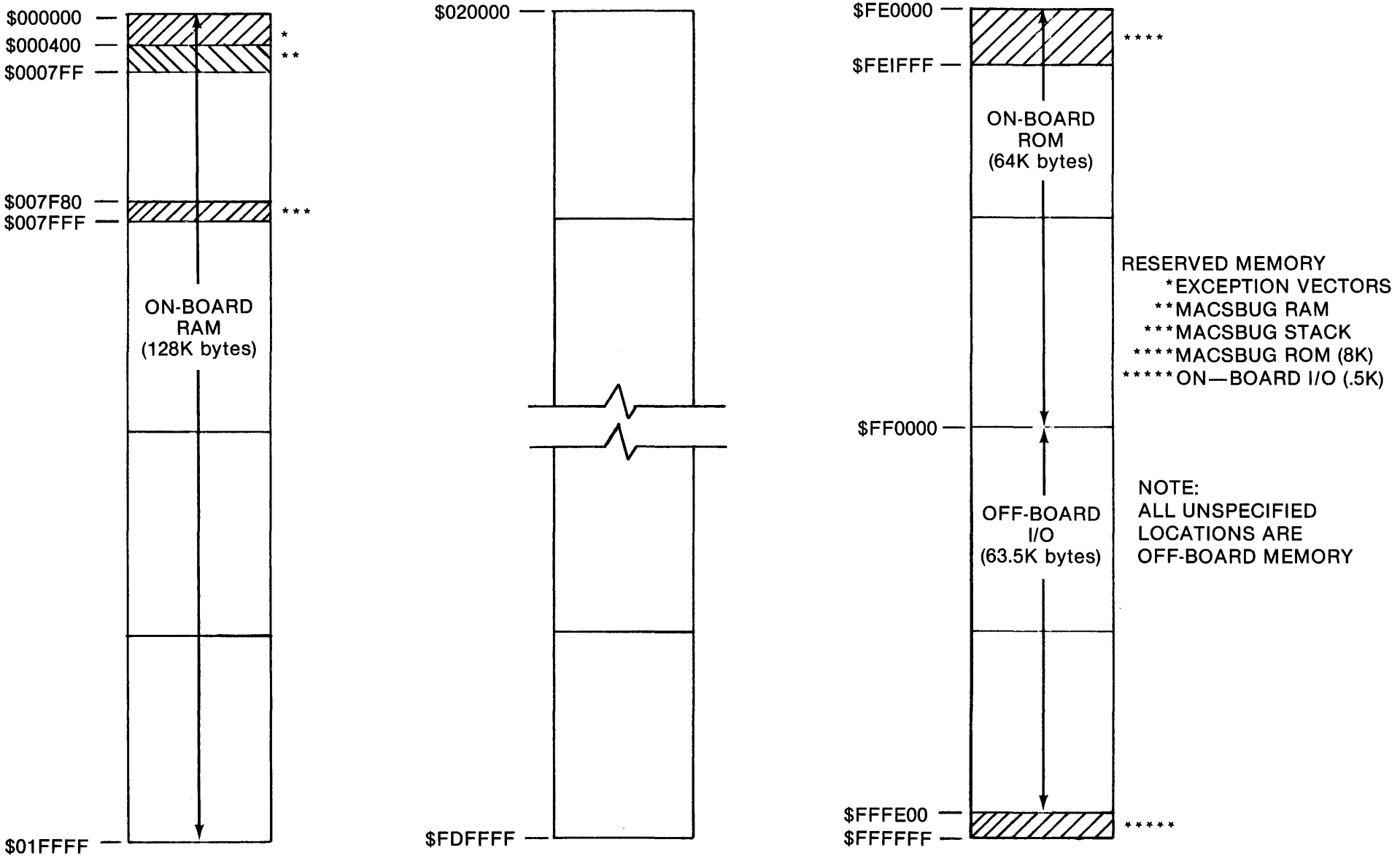
			SWITCH SETTING (ON = 0, OFF = 1)
SW-1	RAM BASE ADDRESS	$000000	0000 0000
SW-2	I/O BASE ADDRESS	$FF0000	1111 1111
SW-3	ROM BASE ADDRESS	$FE0000	1111 1110

The MOTOROLA MEX68KDM Design Module Memory Configuration (MAP 0) is shown in the diagrams of Figure 6.1.3 and 6.1.4. Switch settings for this map are:

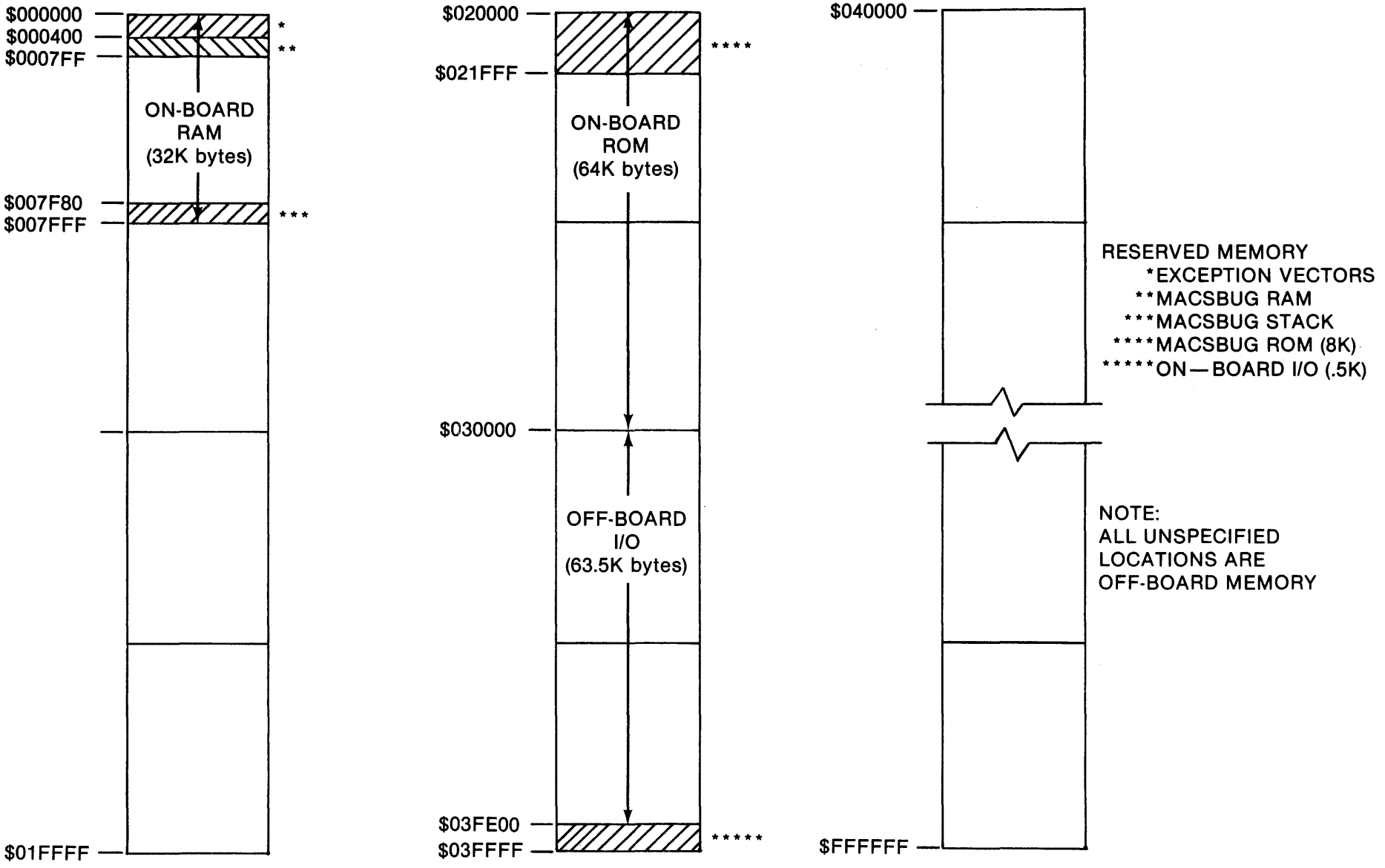
			SWITCH SETTING (ON = 0, OFF = 1)
SW-1	RAM BASE ADDRESS	$000000	0000 0000
SW-2	I/O BASE ADDRESS	$030000	0000 0011
SW-3	ROM BASE ADDRESS	$020000	0000 0010



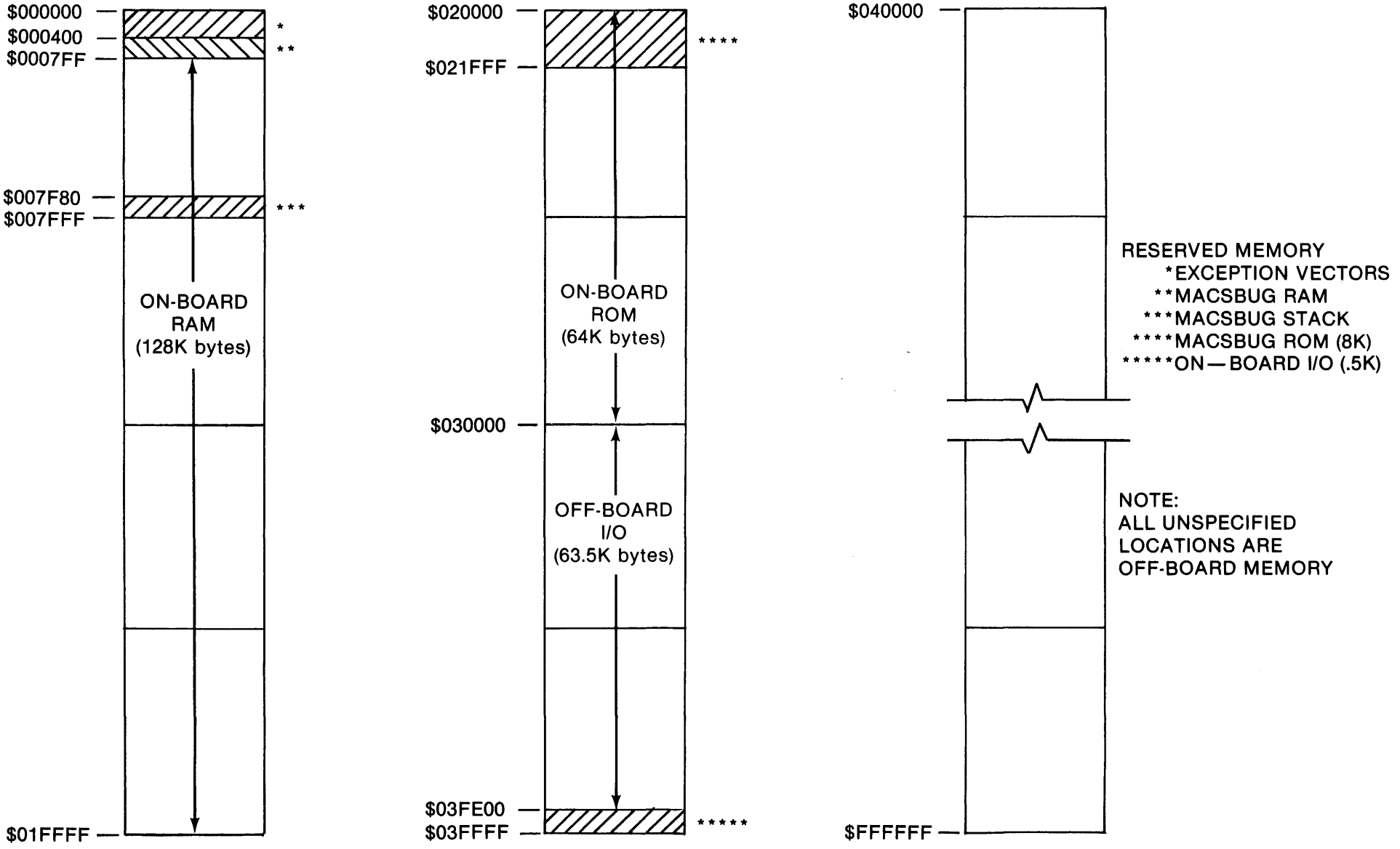
MAP 1 — 32K bytes RAM (FACTORY STANDARD)
FIGURE 6.1.1



MAP 1 — 128K byte RAM (FACTORY STANDARD)
FIGURE 6.1.2



MAP 0 — 32K byte RAM (OPTIONAL)
FIGURE 6.1.3



MAP 0 — 128 byte RAM (OPTIONAL)
FIGURE 6.1.4

6.2 I / O Address Assignments

Table 6.2 shows the standard assignment of the address space for the onboard peripherals. The base address is selected by the setting of SW-2; an 8-bit dip switch that determines the upper 8-bits of the I/O address. In Table 6.2, XX represents the upper two hexidecimal digits determined by SW-2. These switches are normally set at the factory to $FF to place the 64K byte I/O block at the upper end of the 24-bit address space. The least significant bits of the individual peripheral registers were chosen to match the addresses of the Motorola KDM board. These addresses are hardwired and cannot be changed by the user.

DEVICE	ADDRESS MAIN MEMORY	MODE	DESCRIPTION
ACIA 0 (U34) (MC 6850)	$XXFF01	Read	Status Register
	$XXFF01	Write	Control Register
	$XXFF03	Read	Receive Data Register
	$XXFF03	Write	Transmit Data Register
BAUD RATE SELECT (U56) (COM 8116)	$XXFF10	Write	Baud Rate Register
ACIA 1 (U26) (MC 6850)	$XXFF21	Read	Status Register
	$XXFF21	Write	Control Register
	$XXFF23	Read	Receive Data Register
	$XXFF23	Write	Transmit Data Register
PIA 0 (U32) (MC 6821)	$XXFF41	R/W	Data Direction Reg. A (CRA – 2 = 0)
	$XXFF41	R/W	Peripheral Reg. A (CRA – 2 = 1)
	$XXFF43	R/W	Data Direction Reg. B (CRB – 2 = 0)
	$XXFF43	R/W	Peripheral Reg. B (CRB – 2 = 1)
	$XXFF45	R/W	Control Register A
	$XXFF47	R/W	Control Register B
PIA 1 (U33) (MC 6821)	$XXFF40	R/W	Data Direction Reg. A (CRA – 2 = 0)
	$XXFF40	R/W	Peripheral Reg. A (CRA – 2 = 1)
	$XXFF42	R/W	Data Direction Reg. B (CRB – 2 = 0)
	$XXFF42	R/W	Peripheral Reg. B (CRB – 2 = 1)
	$XXFF44	R/W	Control Register A
	$XXFF46	R/W	Control Register B
PTM (U35) (MC 6840)	$XXFF61	Read	Unused
	$XXFF61	Write	Control Register #3 (CR2 – 0 = 0)
	$XXFF61	Write	Control Register #1 (CR2 – 0 = 1)
	$XXFF63	Read	Status Register
	$XXFF63	Write	Control Register #2
	$XXFF65	Read	Timer #1 Counter
	$XXFF65	Write	MSB Buffer Register
	$XXFF67	Read	LSB Buffer Register
	$XXFF67	Write	Timer #1 Latch
	$XXFF69	Read	Timer #1 Counter
	$XXFF69	Write	MSB Buffer Register
	$XXFF6B	Read	LSB Buffer Register
	$XXFF6B	Write	Timer #2 Latch
	$XXFF6D	Read	Timer #3 Counter
	$XXFF6D	Write	MSB Buffer Register
	$XXFF6F	Read	LSB Buffer Register
	$XXFF6F	Write	Timer #3 Latch

NOTE: XX DETERMINED BY SW-2 (FACTORY SET FF.)

ON-BOARD I / O ADDRESS ASSIGNMENT
TABLE 6.2

6.3 Motorola MEX68KDM Compatibility

The OB68K1A can be made object code compatible with the Motorola 68000 Design Module. For compatibility, the user should select the following base addresses:

			Switch Settings (ON = 0, OFF = 1)
SW-1	RAM base address	$000000	00000000
SW-2	I/O base address	$030000	00000011
SW-3	ROM base address	$020000	00000010

These settings will allow the user to execute the MACSbug (MAP 0) terminal monitor program and any other MEX68KDM compatible software. (See Section 6.1 for Memory Maps).

6.4 OB68K1 / OB68K1A Compatibility/Enhancements

The **NEW** Omnibyte 68000-based OB68K1A Single Board Computer is an upgrade of the original Omnibyte OB68K1. Although the OB68K1A is an extensive redesign, careful attention has been given to keeping the "A" version compatible with the earlier OB68K1. We believe that complete compatibility has been maintained in the following important areas:

1) I/O Circuitry.
   The same peripheral chips are included—Two 6850 ACIA's, two 6821 PIA's, one 6840 three channel timer.
2) Connector Pinout.
   The pinout of both serial I/O connectors and the parallel I/O connector are the same as the OB68K1.
3) Address Decoding.
   The OB68K1 address configuration is available on the OB68K1A.
4) Software Compatibility.
   The OB68K1A maintains software compatibility at the "object code" level. All programs developed for the OB68K1 can be run on the OB68K1A without modification.

In spite of the above compatibility, the following important enhancements have been made:

1) The on-board RAM has been dual-ported to make it available to other Multibus Masters.
2) An LSI dynamic RAM controller is used for faster access and increased reliability.
3) Ten MHz operation with no wait-states for on-board RAM access is possible.
4) The base address of on-board RAM, ROM, I/O, and external access to on-board RAM are all switch selectable. No fusible link PROM's are needed to modify the address decoding.

5) ROM Socket sizing may be determined by a configuration plug. No wire-wrapping is necessary.

6) Baud rates may be software or manually selected.

7) Several cut-trace options have been replaced with jumper plugs.

8) Provision for an external Front Panel is included.

9) On-board single step circuitry is implemented.

10) Better ground and power distribution have been achieved through the use of a multi-layer printed circuit board design.

Because of the increased speed, users should be aware that slight difference could arise in routines that operate in real time. The "E-clock" frequency has been increased from 800 KHz to 1MHz and time intervals measured by the on-board MC6840 timer chip will be shortened accordingly. Similarly, because of the increased processor clock frequency and the faster RAM access time, software timing loops will execute faster.

6.5 68000 Memory Organization

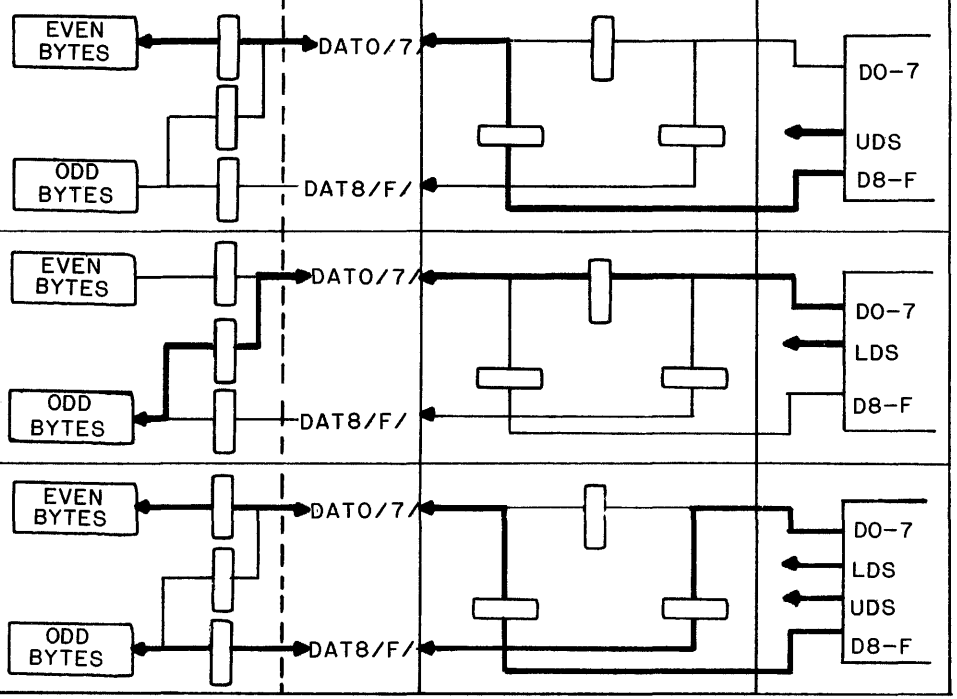
Motorola and Intel processors store 16-bit words with the high and low bytes in the opposite order. Intel words are stored with the least significant byte in the next higher (odd) location. This organization was induced on Multibus memory and peripheral cards by defining a 16-bit transfer to have the even byte asserted on data lines D0-D7; odd bytes on D8-D15.

The 68000 memory organization places the most significant byte in even addresses; least significant bytes at the next higher (odd) addresses.

The design criterion placed on this computer board was that bytes placed in byte addresses will be stored at the proper location in memory, and that data, read as 16-bit words, will appear at the 68000 in the correct order. To achieve this goal a swap byte buffer arrangement is needed on the CPU card. This design is shown in Figure 6.5.

Notice that for byte transfers, no problem exists. The bytes are transferred on D0-D7 as expected. Only the 16-bit transfers are affected, in which case Multibus lines D0-D7 carry the data that will arrive at the 68000 as the most significant byte D8-D15; a similar swap is made for the least significant byte.

BHEN/	ADRO/	MULTIBUS TRANSFER DATA PATH	DEVICE BYTE TRANSFERRED	16 BIT DEVICE	MULTIBUS	68000 CPU BOARD	MC 68000
H	H	8 BIT DATO/-DAT7/	EVEN				
H	L	8 BIT DATO/-DAT7/	ODD				
L	H	16 BIT DATO/-DATF/	EVEN & ODD				

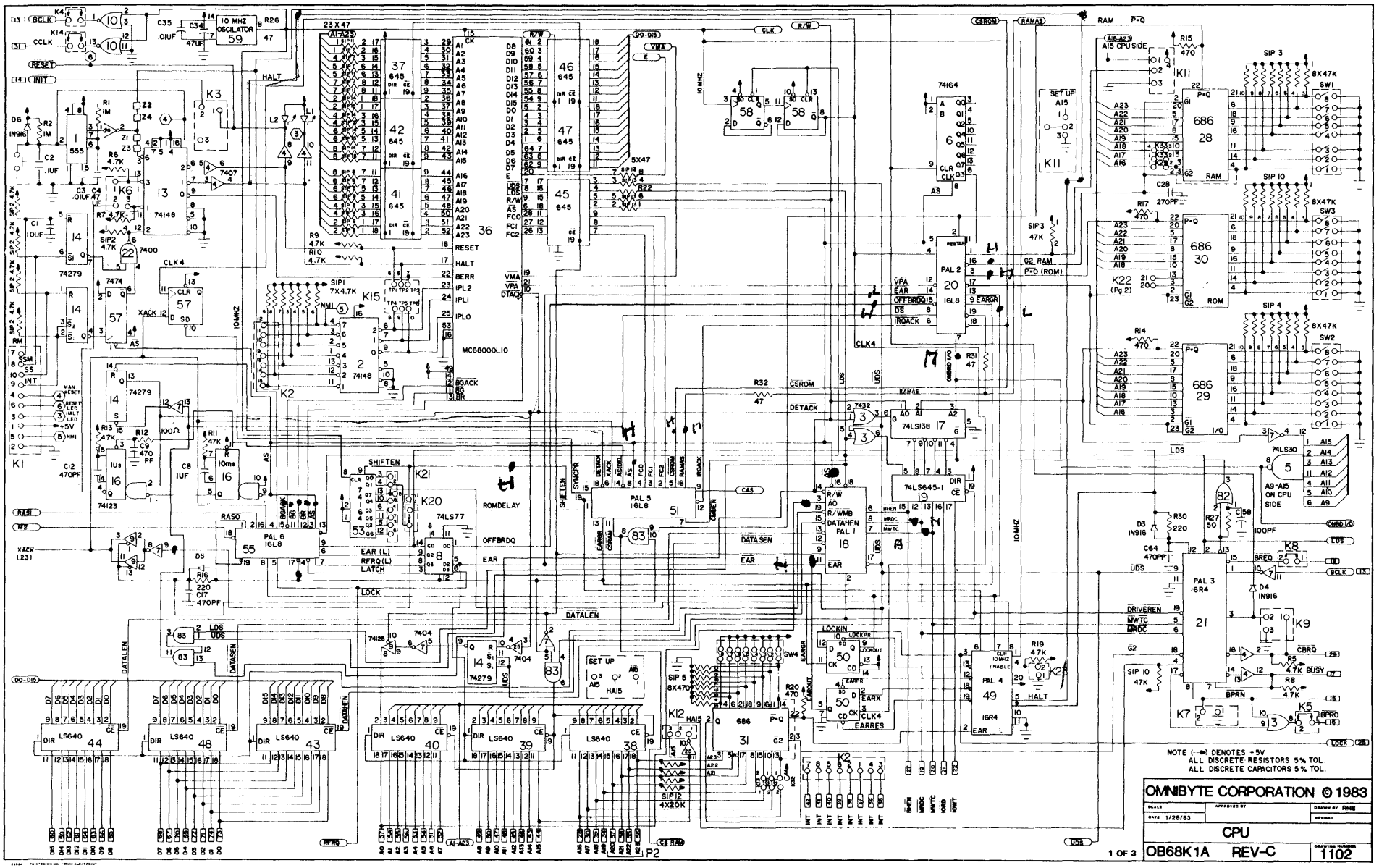


SWAP BYTE BUFFER DIAGRAM
FIGURE 6.5

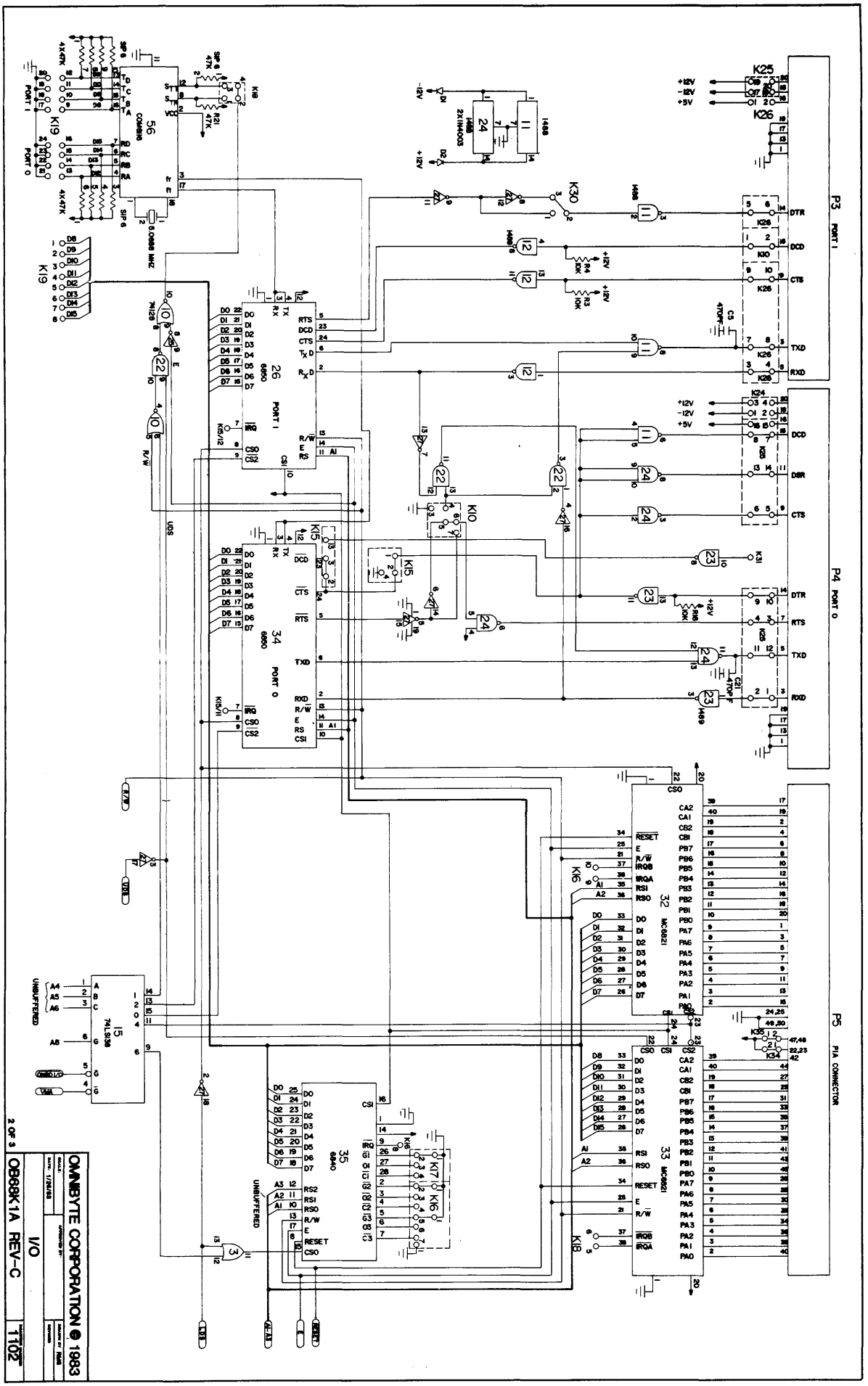
6.6 OB68K1A Schematic Diagrams

The electrical schematic diagrams are shown in Figures 6.6-A, 6.6-B, and 6.6-C. Factory standard jumper options are shown as dotted lines in these figures.

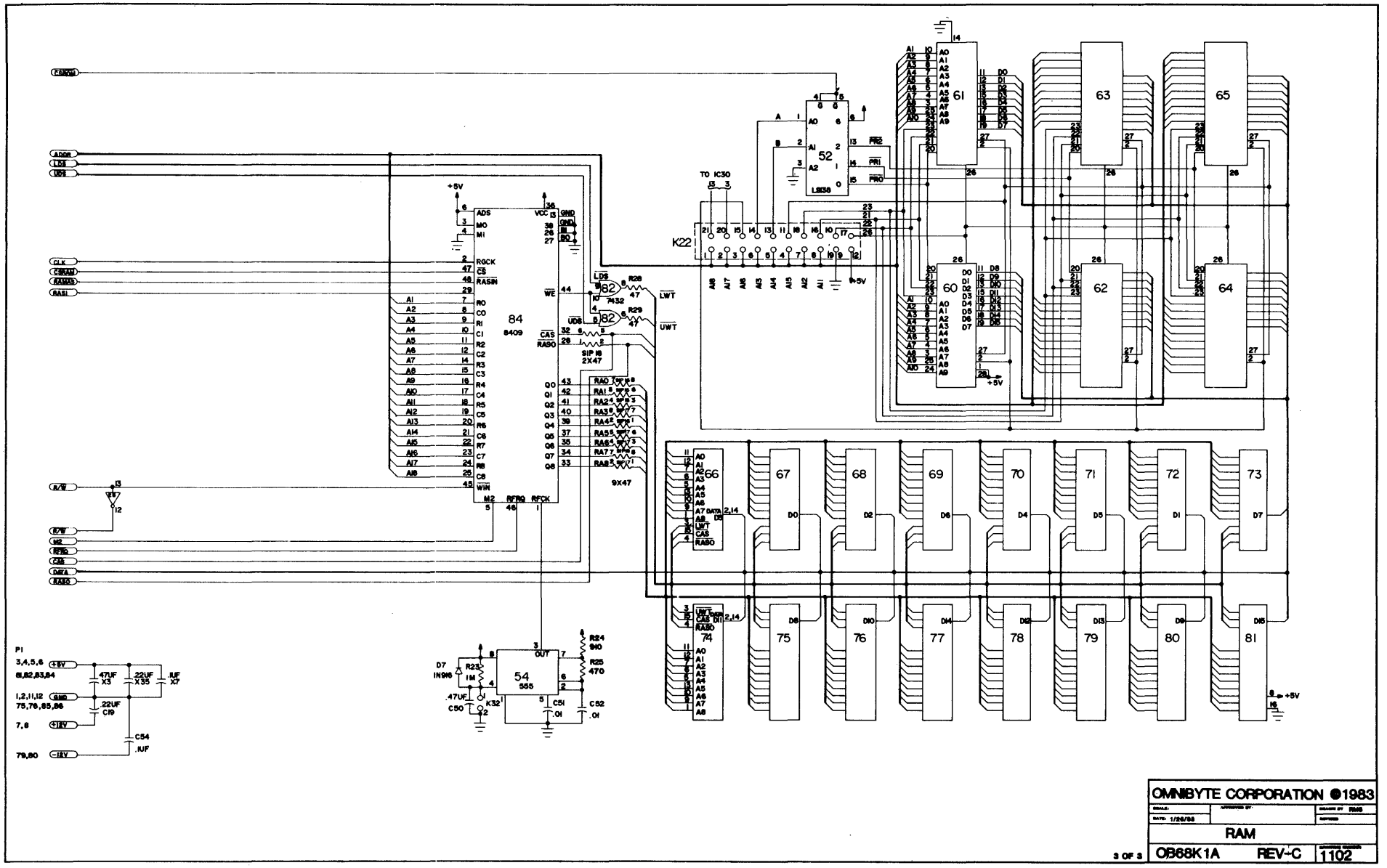
"NOTE"



OB68K1A SCHEMATIC - CPU, BUFFERING AND DECODING
FIGURE 6.6 (A)



OB68K1A SCHEMATIC - I/O
FIGURE 6.6 (B)



OB68K1A SCHEMATIC - MEMORY
FIGURE 6.6 (C)

PART NUMBER	DESCRIPTION	# OF PCS.	LOCATION
OB00406CN	26 PIN RIGHT ENC. HEADER	2	P3, 4
OB00409CN	50 PIN RIGHT ENC. HEADER	1	P5
OB00093CP	270 pF CAP	3	C28, C58, C64
OB00916CP	470 pF CAP	5	C5, 9, 12, 17, 21,
OB00103CP	.01 uF CAP	4	C3, 35, 51, 52,
OB00104CP	.1 uF CAP	8	C2, 20, 24, 25, 27, 32 53, 54
OB00105CP	.47 uF CAP	2	C4, 50
OB00106CP	1.0 uF CAP	2	C8, 61
OB00110CP	47 uF CAP	3	C6, 34, 55
OB00881CP	.22 uF CAP	36	C7, 10, 11, 13, 14, 15, 16, 18, 19, 22, 23, 26 29, 30, 31, 33, 36, 37 38, 39, 40, 41, 42, 43 44, 45, 46, 47, 48, 49 56, 57, 59, 60, 62, 65
OB00108CP	10 uF CAP	1	C1
OB00912CP	1 uF CAP (CERAMIC)	1	C63
OB00134DI	IN4003	2	D1, 2
OB00137DI	IN916	5	D3, 4, 5, 6, 7
OB00153LE	LED (RED)	1	L1
OB00835LE	LED (YELLOW)	1	L2
OB00008RE	47 OHM 1/4W RESISTOR	5	R22, 26, 28, 29, 31, 32
OB00011RE	100 OHM 1/4W	1	R12
OB00013RE	150 OHM 1/4W	1	R27
OB00015RE	220 OHM 1/4W	2	R16, 30
OB00018RE	470 OHM 1/4W	5	R14, 15, 17, 20, 25
OB00035RE	4.7K OHM 1/4W	9	R5, 6, 7, 8, 9, 10, 13 19, 21
OB00042RE	10K OHM 1/4W	3	R3, 4, 18
OB00049RE	47K OHM 1/4W	1	R11
OB00053RE	1M OHM 1/4W	3	R1, 2, 23
OB00919RE	910 OHM 1/4W	1	R24
OB00529SA	2 PIN STRIP	7	K4, 5, 7, 8, 14, 20, 23
OB00586SA	8 PIN STRIP	5	K2, 19, 19, 19, 21
OB00587SA	10 PIN STRIP	1	K16
OB00649SA	7 PIN STRIP	1	K2
OB00681SA	13 PIN STRIP	1	K15
OB00759SA	3 PIN STRIP	6	K3, 6, 9, 10, 11, 12
OB00760SA	4 PIN STRIP	1	K17
OB00808SA	5 PIN STRIP	2	K1, 1
OB00914SA	6 PIN STRIP	1	K18
OB00921SA	1 PIN STRIP	2	K20,32
OB00931SA	6 PIN STRIP	2	K22, K22

OB68K1A PARTS LIST
TABLE 6.6

PART NUMBER	DESCRIPTION	# OF PCS.	LOCATION
OB00932SA	2 PIN STRIP	1	K22,
OB00933SA	4 PIN STRIP	2	K22, K22
OB00377SK	64 PIN SOCKET	1	U36
OB00379SK	14 PIN SOCKET	1	U57
OB00380SK	16 PIN SOCKET	20	U66-U81, SW1-SW4
OB00381SK	18 PIN SOCKET	1	U56
OB00382SK	20 PIN SOCKET	6	U18, 20, 21, 49, 51, 55
OB00384SK	24 PIN SOCKET	2	U26, 34
OB00385SK	28 PIN SOCKET	7	U35, U60-65
OB00386SK	40 PIN SOCKET	2	U32, 33
OB00574SK	48 PIN SOCKET	1	U84
OB00812SK	INDIVIDUAL SOCKET PIN	4	U59, 59, 59, 59
OB00063SP	4.7K OHM 6 PIN SIP	1	SP2
OB00064SP	4.7K OHM 10 PIN SIP	1	SP1
OB00065SP	47K OHM 8 PIN SIP	10	SP7, 8, 9, 11, 13, 14 15, 16, 17, 18
OB00848SP	22K OHM 6 PIN SIP	1	SP12
OB00882SP	470 OHM 10 PIN SIP	1	SP5
OB00913SP	47K OHM 10 PIN SIP	4	SP3, 4, 6, 10
OB00163SW	PUSH BUTTON SWITCH (N.O.)	1	SW5
OB00834SW	8 POSITION SWITCH	4	SW1, 2, 3, 4
OB00149XT	5.0688 MHz CRYSTAL	1	
OB00813IC	7400	1	U22
OB00814IC	7404	2	U7, 25
OB00815IC	7407	1	U4
OB00997IC	74F08	1	U83
OB00817IC	74LS30	1	U5
OB00818IC	7432	1	U3
OB00920IC	74S32	1	U82
OB00819IC	74S74	3	U50, 57, 58
OB00820IC	74LS77	1	U8
OB00216IC	74123	1	U16
OB00821IC	74LS125	1	U9
OB00822IC	74128	1	U10
OB00823IC	74LS138	3	U15, 17, 52
OB00824IC	74LS148	2	U2, 13
OB00825IC	74LS164	2	U6, 53
OB00826IC	74LS279	1	U14
OB00827IC	74LS640	7	U27, 38, 39, 40, 43, 44, 48
OB00828IC	74LS645	6	U37, 41, 42, 45, 46, 47

OB68K1A PARTS LIST - CONT.
TABLE 6.6

PART NUMBER	DESCRIPTION	# OF PCS.	LOCATION
OB00829IC	74LS645-1	1	U19
OB00830IC	NE 555	2	U1, 54
OB00831IC	74LS686	4	U28, 29, 30, 31
OB00832IC	MC1488	2	U11, 24
OB00833IC	MC1489	2	U12, 23
OB00364IC	PAL 16L8	4	U18, 20, 51, 55
OB00585IC	PAL 16R4	2	U21, U49
OB00338IC	MC6821	2	U32, U33
OB00342IC	MC6840	1	U35
OB00347IC	MC6850	2	U26, U34
OB00357IC	MC68000L10	1	U36
OB00255IC	COM8116	1	U56
OB00263IC	DP8409(-2)	1	U84
OB00690IC	64K X 1D RAM (128K)	16	U66-81
OB00599IC	16K X 1D RAM (32K)	16	U66-81
OB00761XT	10 MHZ OSCILLATOR	1	U59
OB00809JP	JUMPER (128K)	13	STD. CONFIGURATION
	(32K)	14	STD. CONFIGURATION
OB00923SA	JUMPER KIT (17 JUMPERS)		USER JUMPER OPTIONS
OB00501CE	BLUE CARD EJECTOR	2	

OB68K1A PARTS LIST - CONT.
TABLE 6.6

7.0 TERMINAL MONITOR PROGRAMS

Two terminal monitor programs are available from Omnibyte for use with your OB68K1A. These programs are licensed for distribution from Motorola Inc. by Omnibyte and are provided as object code in PROM. Both programs include the source listings of initialization and I/O routines.

OB68KMACS is an 8K byte firmware monitor which provides for memory examination, program loading and controlled program execution. It provides the user with the capability to connect a terminal to the OB68K1A single board computer and communicate with the board. It provides a vehicle to initialize and configure the on-board I/O devices, examine the MC68000 registers, and perform various other functions of this powerful monitor as listed in the command summary.

OB68KMACS is essentially Motorola's MACSbug firmware monitor program designed for the MEX68KDM design module.

OB68KVERSA is a 16K byte firmware monitor/debugger which provides the user with all the functions of MACSbug, described above, and has been extended to include a single line assembler/disassembler; block move, block test, block initialize capabilities and a Help menu.

OB68KMACS and OB68KVERSA are available in the OMNIBYTE factory standard memory configuration (MAP 1) or the Motorola MEX68KDM DESIGN MODULE memory configuration (MAP 0).

8.0 WARRANTY INFORMATION

All boards manufactured and sold by Omnibyte Corporation are warranteed against defects in materials or workmanship and are guaranteed to meet specifications in effect at the time of manufacture for a period of (2) years from the date of delivery.

Omnibyte's responsibility under this warranty is limited to repair or replacement of any item (at our option) returned to the factory during the warranty period, **postage prepaid.** Omnibyte shall either repair or replace the item, provided that the failure of the item, in our opinion, was not due to abuse, modification or acts of God. EXCEPT AS OTHERWISE INDICATED, THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Omnibyte's liability shall be limited to the purchase price of the item or items, and Omnibyte shall not be responsible or liable for any lost profits or consequential damages, or for any claim against the purchaser by any party.

"NOTE"

ALL BOARDS BEING RETURNED TO OMNIBYTE CORPORATION FOR REPAIR MUST BE ACCOMPANIED WITH A RMA NUMBER (RETURN MATERIAL AUTHORIZATION). THIS NUMBER WILL BE ISSUED TO YOU BY OMNIBYTE WHEN REQUESTING AUTHORIZATION TO RETURN YOUR BOARD. ALSO, AN EXPLANATION OF THE PROBLEM AND NAME AND PHONE NUMBER OF THE PERSON USING THE BOARD WHEN PROBLEM OCCURED SHOULD BE ENCLOSED WITH THE RETURNED BOARD. THIS PROCEDURE WILL HELP US TO SPEED UP THE REPAIR AND RETURN OF YOUR BOARD.

ANY BOARDS RECEIVED BY OMNIBYTE CORPORATION WITHOUT A RMA NUMBER WILL BE RETURNED TO SENDER, AT THEIR EXPENSE, UNTIL A RMA NUMBER HAS BEEN OBTAINED FROM OMNIBYTE.

9.0 ORDERING INFORMATION

Order numbers for the Omnibyte OB68K1A computer, Monitor programs and ROM size configuration plugs and other accessories are as follows:

32K RAM Computer	OB68K1A—32K
128K RAM Computer	OB68K1A—128K
MACSBug in 8K X 8 PROMS	
MAP 0 (See Section 7.0)	OB68KMACS—64—0
MAP 1 (See Section 7.0)	OB68KMACS—64—1
VERSABug in 8K X 8 PROMS	
MAP 1 (See Section 7.0)	OB68KVERSA—64—1
MAP 0 (See Section 7.0)	OB68KVERSA—64—0
ROM Configuration Plug for 2716	OBK1A/K22—2716
2732	OBK1A/K22—2732
2764	OBK1A/K22—2764
27128	OBK1A/K22—27128
27256	OBK1A/K22—27256
User-definable ROM Configuration Plug	OBK1A/K22—UD
Optional Front Panel Box w/5′ cable	OB68K1AFPM
Serial I/O Interface Cable Assy. - 10′ long	OB68K1S1C
Parallel I/O Interface Cable Assy. - 5′ long	OB68K1P1C
Dual In Line Shunt	OB01049CN

Omnibyte's terms are: NET 30 DAYS with approved credit. The F.O.B. point is: West Chicago, Illinois. Items will be shipped: United Parcel Service surface unless otherwise instructed.

10.0 APPENDIX (DATA SHEETS)

The following pages contain excerpts from the data sheets of the OB68K1A on-board devices. Familiarity with the operating characteristics of these devices will be necessary for proper operation and are included for that purpose. They have been reprinted courtesy of Motorola, Inc.

The reprints contained herein are the most current available at the time of printing and may be updated by Motorola, Inc. without notice at any time. Omnibyte assumes no liability for notifying purchasers of this manual of these updates.

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC68000L4
(4 MHz)
MC68000L6
(6 MHz)
MC68000L8
(8 MHz)
MC68000L10
(10 MHz)

Advance Information

HMOS
(HIGH-DENSITY, N-CHANNEL, SILICON-GATE DEPLETION LOAD)

16-BIT MICROPROCESSOR

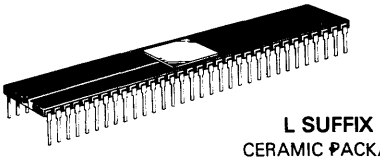
16-BIT MICROPROCESSING UNIT

Advances in semiconductor technology have provided the capability to place on a single silicon chip a microprocessor at least an order of magnitude higher in performance and circuit complexity than has been previously available. The MC68000 is the first of a family of such VLSI microprocessors from Motorola. It combines state-of-the-art technology and advanced circuit design techniques with computer sciences to achieve an architecturally advanced 16-bit microprocessor.

The resources available to the MC68000 user consist of the following:

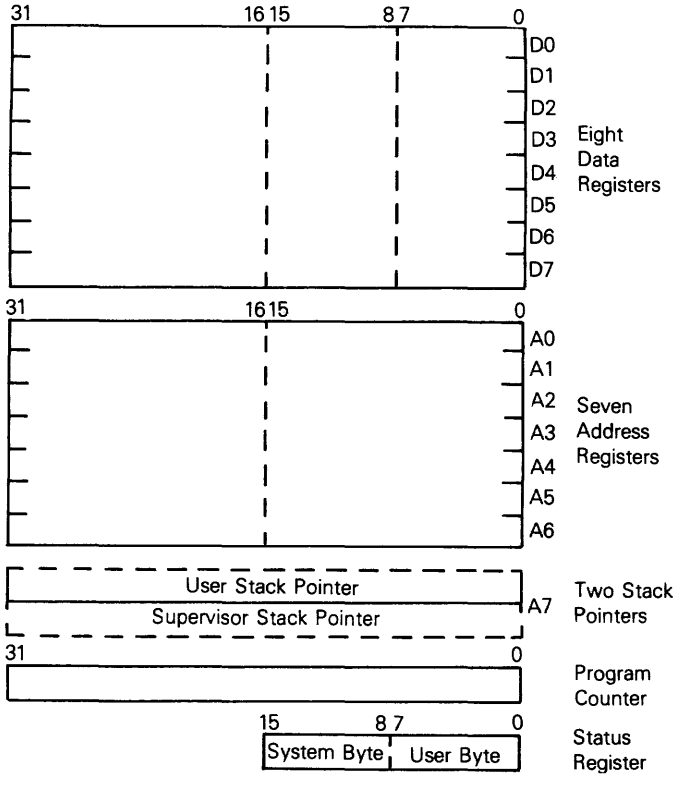
- 32-Bit Data and Address Registers
- 16 Megabyte Direct Addressing Range
- 56 Powerful Instruction Types
- Operations on Five Main Data Types
- Memory Mapped I/O
- 14 Addressing Modes

As shown in the programming model, the MC68000 offers seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. The first eight registers (D0-D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0-A6) and the system stack pointer may be used as software stack pointers and base address registers. In addition, these registers may be used for word and long word address operations. All seventeen registers may be used as index registers.

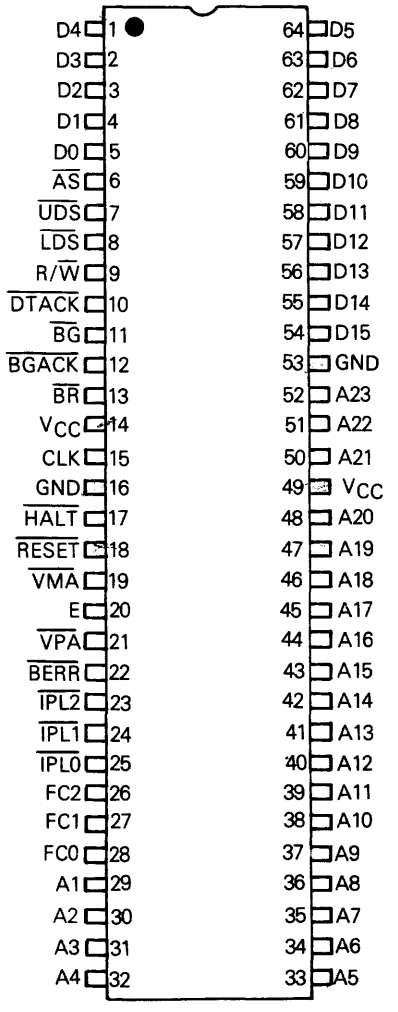


L SUFFIX
CERAMIC PACKAGE
CASE 746

PROGRAMMING MODEL



PIN ASSIGNMENT



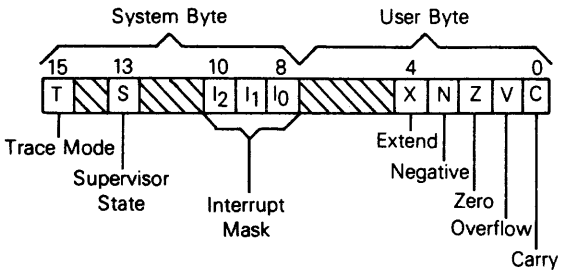
This document contains information on a new product. Specifications and information herein are subject to change without notice.

ADI-814R2

A 23-bit address bus provides a memory addressing range of greater than 16 megabytes. This large range of addressing capability, coupled with a memory management unit, allows large, modular programs to be developed and operated without resorting to cumbersome and time consuming software bookkeeping and paging techniques.

The status register contains the interrupt mask (eight levels available) as well as the condition codes; extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace (T) mode and/or in a supervisor (S) state.

STATUS REGISTER



Five basic data types are supported. These data types are:

- Bits
- BCD Digits (4-bits)
- Bytes (8-bits)
- Word (16-bits)
- Long Words (32-bits)

In addition, operations on other data types such as memory addresses, status word data, etc., are provided for in the instruction set.

The 14 addressing modes, shown in Table 1, include six basic types:

- Register Direct
- Register Indirect
- Absolute
- Immediate
- Program Counter Relative
- Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting and indexing. Program counter relative mode can also be modified via indexing and offsetting.

TABLE 1 - ADDRESSING MODES

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An), An ← An + N
Predecrement Register Indirect	An ← An − N, EA = (An)
Register Indirect with Offset	EA = (An) + d_{16}
Indexed Register Indirect with Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:

EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
( ) = Contents of
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words
← = Replaces

INSTRUCTION SET OVERVIEW

The MC68000 instruction set is shown in Table 2. Some additional instructions are variations, or subsets, of these and they appear in Table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

TABLE 2 — INSTRUCTION SET

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

TABLE 3 — VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

The following paragraphs describe the data organization and addressing capabilities of the MC68000.

OPERAND SIZE

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. All explicit instructions support byte, word or long word operands. Implicit instructions support some subset of all three sizes.

DATA ORGANIZATION IN REGISTERS

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers together with the active stack pointer support address operands of 32 bits.

DATA REGISTERS. Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

ADDRESS REGISTERS. Each address register and the stack pointer is 32 bits wide and holds a full 32 bit address. Address registers do not support byte sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

DATA ORGANIZATION IN MEMORY

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in Figure 1. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the second word of that datum is located at address n+2.

The data types supported by the MC68000 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in Figure 2.

ADDRESSING

Instructions for the MC68000 contain two kinds of information: the type of function to be performed, and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

- Register Specification – the number of the register is given in the register field of the instruction.
- Effective Address – use of the different effective address modes.
- Implicit Reference – the definition of certain instructions implies the use of specific registers.

FIGURE 1 – WORD ORGANIZATION IN MEMORY

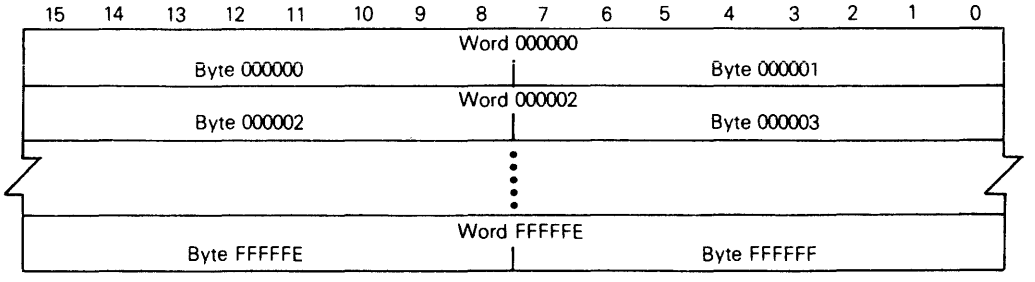
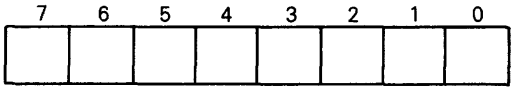
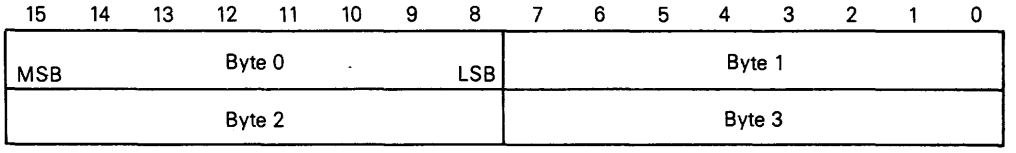


FIGURE 2 — DATA ORGANIZATION IN MEMORY

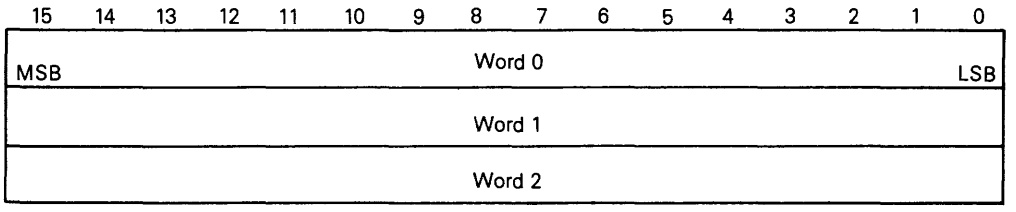
Bit Data
1 Byte = 8 Bits



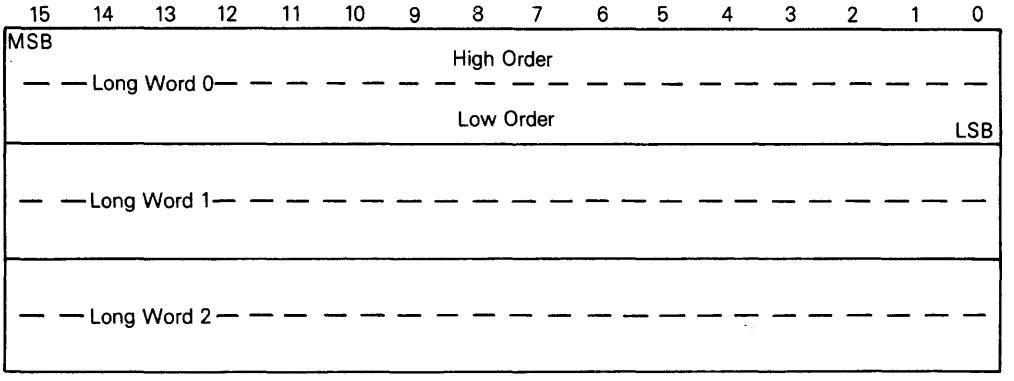
Integer Data
1 Byte = 8 Bits



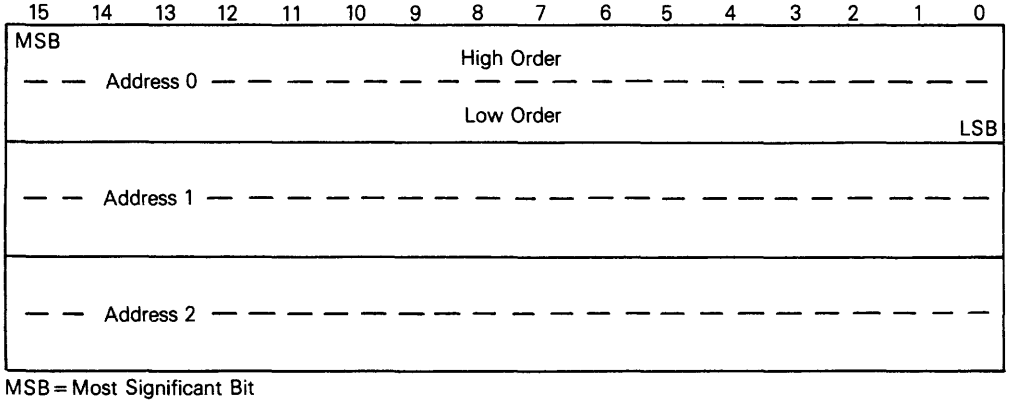
1 Word = 16 Bits



1 Long Word = 32 Bits

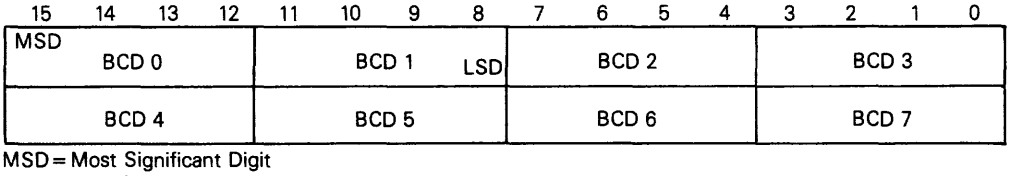


Addresses
1 Address = 32 Bits



MSB = Most Significant Bit
LSB = Least Significant Bit

Decimal Data
2 Binary Coded Decimal Digits = 1 Byte



MSD = Most Significant Digit
LSD = Least Significant Digit

INSTRUCTION FORMAT

Instructions are from one to five words in length, as shown in Figure 3. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

PROGRAM/DATA REFERENCES

The MC68000 separates memory references into two classes: program references, and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Generally, operand reads are from the data space. All operand writes are to the data space.

REGISTER SPECIFICATION

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, Figure 4 shows the general format of the single effective address instruction operation word. The effective address is composed of two 3-bit fields: the mode field, and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in Figure 3. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

REGISTER DIRECT MODES. These effective addressing modes specify that the operand is in one of the 16 multifunction registers.

Data Register Direct. The operand is in the data register specified by the effective address register field.

Address Register Direct. The operand is in the address register specified by the effective address register field.

MEMORY ADDRESS MODES. These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect. The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Address Register Indirect With Postincrement. The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect With Predecrement. The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect With Displacement. This address mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Address Register Indirect With Index. This address mode requires one word of extension. The address of the operand

FIGURE 3 — INSTRUCTION FORMAT

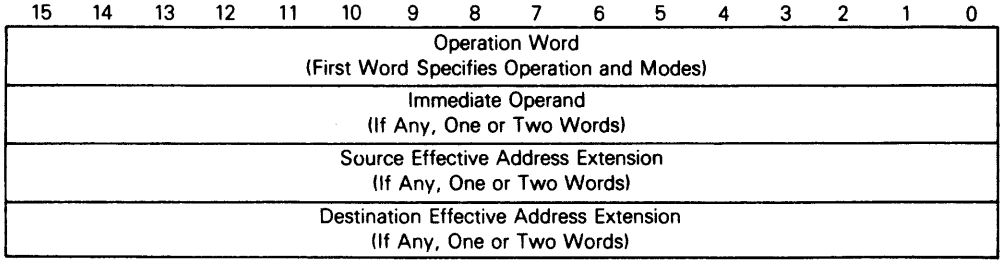
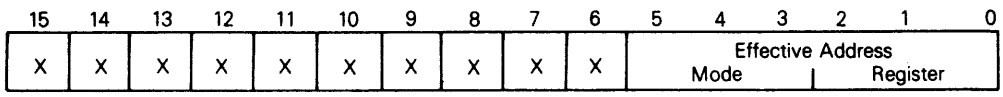


FIGURE 4 — SINGLE-EFFECTIVE-ADDRESS INSTRUCTION OPERATION WORD GENERAL FORMAT



is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

SPECIAL ADDRESS MODES. The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address. This address mode requires one word of extension. The address of the operand is the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Absolute Long Address. This address mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high-order part of the address is the first extension word; the low-order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Program Counter With Displacement. This address mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Program Counter With Index. This address mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classified as a program reference.

Immediate Data. This address mode requires either one or two words of extension depending on the size of the operation.

Byte operation — operand is low order byte of extension word

Word operation — operand is extension word

Long word operation — operand is in the two extension words, high-order 16 bits are in the first extension word, low-order 16 bits are in the second extension word.

Condition Codes or Status Register. A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR
ANDI to SR
EORI to CCR
EORI to SR
ORI to CCR
ORI to SR

EFFECTIVE ADDRESS ENCODING SUMMARY

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

IMPLICIT REFERENCE

Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR).

SYSTEM STACK. The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S-bit in the status register. If the S-bit indicates supervisor state, SSP is the active system stack pointer, and the USP cannot be referenced as an address register. If the S-bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

TABLE 4 — EFFECTIVE ADDRESS ENCODING SUMMARY

Addressing Mode	Mode	Register
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement	101	register number
Address Register Indirect with Index	110	register number
Absolute Short	111	000
Absolute Long	111	001
Program Counter with Displacement	111	010
Program Counter with Index	111	011
Immediate	111	100

INSTRUCTION SET SUMMARY

The following paragraphs contain an overview of the form and structure of the MC68000 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data Movement
Integer Arithmetic
Logical
Shift and Rotate
Bit Manipulation
Binary Coded Decimal
Program Control
System Control

The complete range of instruction capabilities combined with the flexible addressing modes described previously provide a very flexible base for program development.

DATA MOVEMENT OPERATIONS

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address move instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 6 is a summary of the data movement operations.

TABLE 6 — DATA MOVEMENT OPERATIONS

Instruction	Operand Size	Operation
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	An → SP@ – SP → An SP + d → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVEP	16, 32	(EA) → Dn Dn → EA
MOVEQ	8	#xxx → Dn
PEA	32	EA → SP@ –
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp SP@ + → An

NOTES:
s = source
d = destination
[ ] = bit numbers
@ – = indirect with predecrement
@ + = indirect with postdecrement

INTEGER ARITHMETIC OPERATIONS

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 7 is a summary of the integer arithmetic operations.

TABLE 7 — INTEGER ARITHMETIC OPERATIONS

Instruction	Operand Size	Operation
ADD	8, 16, 32 16, 32	Dn + (EA) → Dn (EA) + Dn → EA (EA) + #xxx → EA An + (EA) → An
ADDX	8, 16, 32 16, 32	Dx + Dy + X → Dx Ax@ – Ay@ – + X → Ax@
CLR	8, 16, 32	0 → EA
CMP	8, 16, 32 16, 32	Dn – (EA) (EA) – #xxx Ax@ + – Ay@ + An – (EA)
DIVS	32 ÷ 16	Dn/(EA) → Dn
DIVU	32 ÷ 16	Dn/(EA) → Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 * 16 → 32	Dn * (EA) → Dn
MULU	16 * 16 → 32	Dn * (EA) → Dn
NEG	8, 16, 32	0 – (EA) → EA
NEGX	8, 16, 32	0 – (EA) – X → EA
SUB	8, 16, 32 16, 32	Dn – (EA) → Dn (EA) – Dn → EA (EA) – #xxx → EA An – (EA) → An
SUBX	8, 16, 32	Dx – Dy – X → Dx Ax@ – – Ay@ – – X → Ax@
TAS	8	(EA) – 0, 1 → EA[7]
TST	8, 16, 32	(EA) – 0

NOTE: [ ] = bit number

LOGICAL OPERATIONS

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 8 is a summary of the logical operations.

TABLE 8 — LOGICAL OPERATIONS

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn∧(EA) → Dn (EA)∧Dn → EA (EA)∧#xxx → EA
OR	8, 16, 32	Dn v (EA) → Dn (EA) v Dn → EA (EA) v #xxx → EA
EOR	8, 16, 32	(EA) ⊕ Dy → EA (EA) ⊕ #xxx → EA
NOT	8, 16, 32	~(EA) → EA

NOTE: ~ = invert

SHIFT AND ROTATE OPERATIONS

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in the instruction of one to eight bits, or 0 to 63 specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates.

Table 9 is a summary of the shift and rotate operations.

TABLE 9 — SHIFT AND ROTATE OPERATIONS

Instruction	Operand Size	Operation
ASL	8, 16, 32	X/C ← ← ← 0
ASR	8, 16, 32	→ → → X/C
LSL	8, 16, 32	X/C ← ← ← 0
LSR	8, 16, 32	0 → → → X/C
ROL	8, 16, 32	C ← ← ←
ROR	8, 16, 32	→ → → C
ROXL	8, 16, 32	C ← ← ← X
ROXR	8, 16, 32	X → → → C

BIT MANIPULATION OPERATIONS

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 10 is a summary of the bit manipulation operations. (Bit 2 of the status register is Z.)

TABLE 10 — BIT MANIPULATION OPERATIONS

Instruction	Operand Size	Operation
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z ~bit of (EA) → bit of EA

BINARY CODED DECIMAL OPERATIONS

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 11 is a summary of the binary coded decimal operations.

TABLE 11 — BINARY CODED DECIMAL OPERATIONS

Instruction	Operand Size	Operation
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $Ax@-_{10} + Ay@-_{10} + X \rightarrow Ax@$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $Ax@-_{10} - Ay@-_{10} - X \rightarrow Ax@$
NBCD	8	$0 - (EA)_{10} - X \rightarrow EA$

PROGRAM CONTROL OPERATIONS

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in Table 12.

The conditional instructions provide setting and branching for the following conditions:

CC	— carry clear	LS	— low or same
CS	— carry set	LT	— less than
EQ	— equal	MI	— minus
F	— never true	NE	— not equal
GE	— greater or equal	PL	— plus
GT	— greater than	T	— always true
HI	— high	VC	— no overflow
LE	— less or equal	VS	— overflow

TABLE 12 — PROGRAM CONTROL OPERATIONS

Instruction	Operation
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

SYSTEM CONTROL OPERATIONS

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in Table 13.

BUS ERROR AND HALT OPERATION. In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error signal is received, the processor has two options: initiate a bus error exception sequence or try running the bus cycle again.

Exception Sequence. The bus error exception sequence is entered when the processor receives a bus error signal and the halt pin is inactive. The sequence is composed of the following elements:

1. Stacking the program counter and status register
2. Stacking the error information
3. Reading the bus error vector table entry
4. Executing the bus error handler routine

The stacking of the program counter and the status register is the same as if an interrupt had occurred. Several additional items are stacked when a bus error occurs. These items are used to determine the nature of the error and correct it, if possible. The bus error vector is vector number two located at address $000008. The processor loads the new program counter from this location. A software bus error handler routine is then executed by the processor. Refer to **EXCEPTION PROCESSING** for additional information.

TABLE 13 — SYSTEM CONTROL OPERATIONS

Instruction	Operation
Privileged	
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
ORI to SR	Logical OR to status register
MOVE USP	Move user stack pointer
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
Trap Generating	
TRAP	Trap
TRAPV	Trap on overflow
CHK	Check register against bounds
Status Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
ORI to CCR	Logical OR to condition codes
MOVE SR to EA	Store status register

Re-Running the Bus Cycle. When the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence.

The processor completes the bus cycle, then puts the address, data and function code output lines in the high-impedance state. The processor remains "halted," and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed before the halt signal is removed.

NOTE

The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a Test-and-Set operation is performed without ever releasing $\overline{AS}$.

Halt Operation with No Bus Error. The halt input signal to the MC68000 performs a Halt/Run/Single-Step function in a similar fashion to the M6800 halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

The single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the "run" mode until the processor starts a bus cycle then changing to the "halt" mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state. These include:

1. address lines
2. data lines

This is required for correct performance of the re-run bus cycle operation.

Note that when the processor honors a request to halt, the function codes are put in the high-impedance state (their buffer characteristics are the same as the address buffers). While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

Double Bus Faults. When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception, and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

RESET OPERATION. The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a RESET sequence is executed, the processor drives the reset pin for 124 clock pulses. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a RESET instruction. All external devices connected to the reset line should be reset at the completion of the RESET instruction.

PROCESSING STATES

The following paragraphs describe the actions of the MC68000 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions is detailed.

The MC68000 is always one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed. In this state, no further memory references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

PRIVILEGE STATES

The processor operates in one of two states of privilege: the "user" state or the "supervisor" state. The privilege state determines which operations are legal, is used by the external memory management device to control and translate accesses, and is used to choose between the supervisor stack pointer and the user stack pointer in instruction references.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

SUPERVISOR STATE. The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the status register; if the S-bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

USER STATE. The user state is the lower state of privilege. For instruction execution, the user state is determined by the S-bit of the status register; if the S-bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

The bus cycles generated by an instruction executed in user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly, or address register seven explicitly, access the user stack pointer.

PRIVILEGE STATE CHANGES. Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S-bit of the status register is saved and the S-bit is asserted, putting the processing in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

REFERENCE CLASSIFICATION. When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor states, such as interrupt acknowledge. Table 17 lists the classification of references.

TABLE 17 — REFERENCE CLASSIFICATION

Function Code Output			Reference Class
FC2	FC1	FC0	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

EXCEPTION PROCESSING

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made, and the status register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

EXCEPTION VECTORS. Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (Figure 21), except for the reset

FIGURE 21 — EXCEPTION VECTOR FORMAT

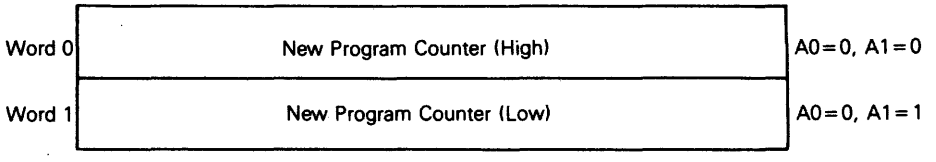
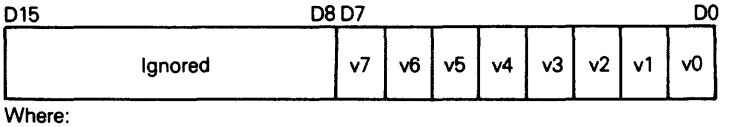


FIGURE 22 — PERIPHERAL VECTOR NUMBER FORMAT



Where:
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an eight-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (Figure 22) to the processor on data bus lines D0 through D7. The processor translates the vector number into a full 24-bit address, as shown in Figure 23. The memory layout for exception vectors is given in Table 18.

As shown in Table 18, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are reserved for TRAPS and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 64 entries, so user interrupt vectors may overlap at the discretion of the systems designer.

KINDS OF EXCEPTIONS. Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from ad-

FIGURE 23 — ADDRESS TRANSLATED FROM 8-BIT VECTOR NUMBER

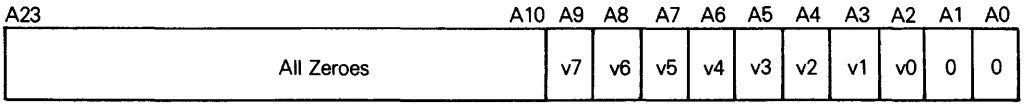


TABLE 18 — EXCEPTION VECTOR ASSIGNMENT

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
–	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Unitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		–
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		–
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		–
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		–

*Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

dress errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution.

EXCEPTION PROCESSING SEQUENCE. Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S-bit is asserted, putting the processor into the supervisor privilege state. Also, the T-bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch, classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer. The program counter value stacked usually points to the next unexecuted instruction, however for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

MULTIPLE EXCEPTIONS. These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The Group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The Group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while Group 2 exceptions have lowest priority. Within Group 0, reset has highest priority, followed by bus error and then address error. Within Group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within Group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in Table 19.

TABLE 19 — EXCEPTION GROUPING AND PRIORITY

Group	Exception	Processing
0	Reset Bus Error Address Error	Exception processing begins
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, Zero Divide	Exception processing is started by normal instruction execution

EXCEPTION PROCESSING DETAILED DISCUSSION

Exceptions have a number of sources, and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

RESET. The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

INTERRUPTS. Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels

are numbered from one to seven, level seven being the highest priority. The status register contains a three-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in a following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in Figure 24.

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

UNINITIALIZED INTERRUPT. An interrupting device asserts VPA or provides an interrupt vector during an interrupt acknowledge cycle to the MC68000. If the vector register has not been initialized, the responding M68000 Family peripheral will provide vector 15, the unitialized interrupt vector. This provides a uniform way to recover from a programming error.

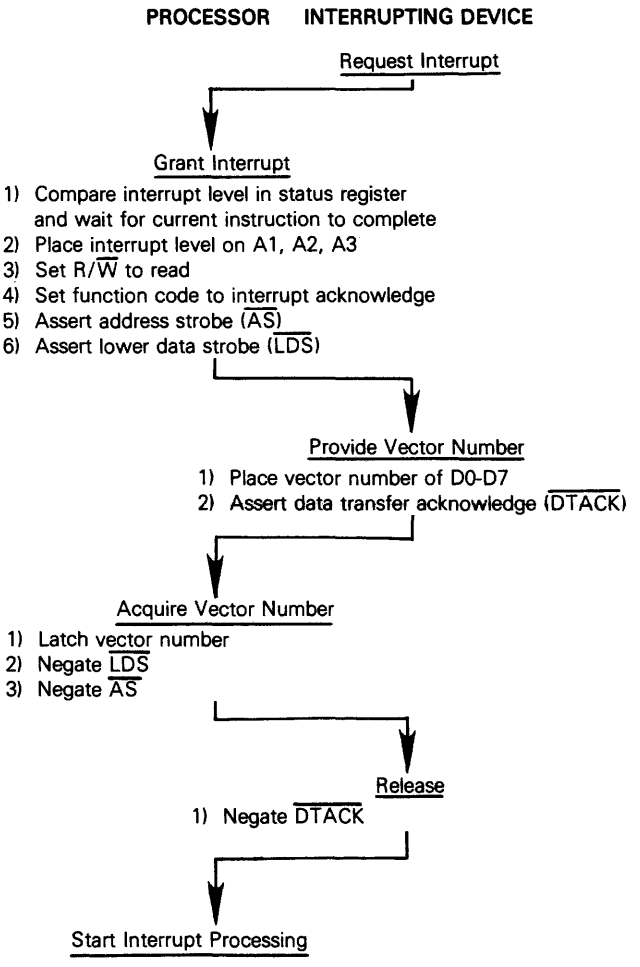
SPURIOUS INTERRUPT. If during the interrupt acknowledge cycle no device responds by asserting DTACK or VPA, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

INSTRUCTION TRAPS. Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception, and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a run-time error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

FIGURE 24 — INTERRUPT ACKNOWLEDGE SEQUENCE FLOW CHART



ILLEGAL AND UNIMPLEMENTED INSTRUCTIONS. Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

PRIVILEGE VIOLATIONS. In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

STOP	AND (word) Immediate to SR
RESET	EOR (word) Immediate to SR
RTE	OR (word) Immediate to SR
MOVE to SR	MOVE USP

TRACING. To aid in program development, the MC68000 includes a facility to allow instruction by instruction tracing. In the trace state, after each instruction is executed an exception is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T-bit in the supervisor portion of the status register. If the T-bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T-bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction, an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

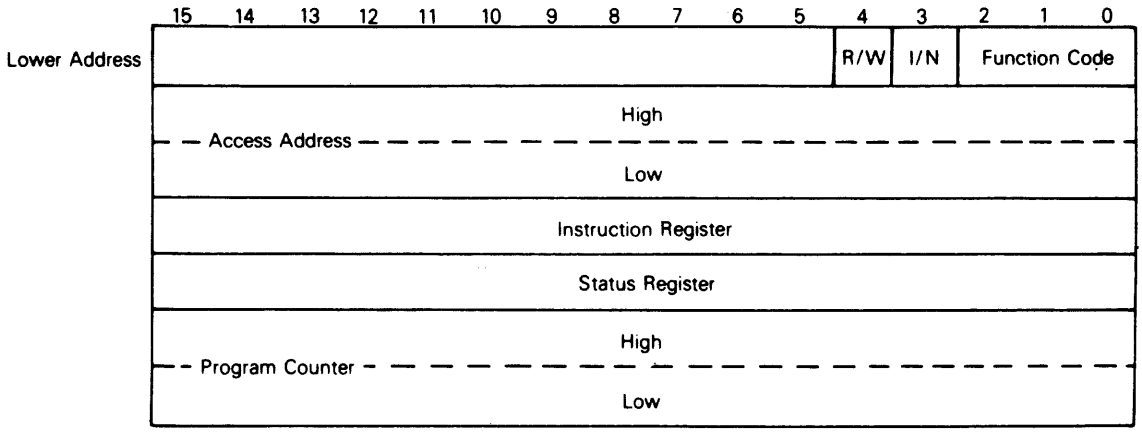
BUS ERROR. Bus error exceptions occur when the external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value saved for the program counter is advanced by some amount, two to ten bytes beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed, and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether the processor was processing an instruction or not, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a Group 2 exception; the processor is not processing an instruction if it is processing a Group 0 or a Group 1 exception. Figure 26 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted, and all processing ceases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroy all memory contents. Only the RESET pin can restart a halted processor.

ADDRESS ERROR. Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and begins exception processing. After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted.

FIGURE 26 — SUPERVISOR STACK ORDER



R/W (read/write): write = 0, read = 1. I/N (instruction/not): instruction = 0, not = 1

INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the MC68000.

ADDRESSING CATEGORIES

Effective address modes may be categorized by the ways in which they may be used. The following classifications will be used in the instruction definitions.

Data	If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.
Memory	If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.
Alterable	If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.
Control	If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 20 shows the various categories to which each of the effective address modes belong. Table 21 is the instruction set summary.

The status register addressing mode is not permitted unless it is explicitly mentioned as a legal addressing mode.

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable.

INSTRUCTION PRE-FETCH

The MC68000 uses a 2-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1) When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2) In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3) The last fetch from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4) If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch. In the case of an interrupt or trace exception, both words are not used.
5) The program counter usually points to the last word fetched from the instruction stream.

TABLE 20 — EFFECTIVE ADDRESSING MODE CATEGORIES

Effective Address Modes	Mode	Register	Data	Addressing Categories		
				Memory	Control	Alterable
Dn	000	register number	X	–	–	X
An	001	register number	–	–	–	X
An@	010	register number	X	X	X	X
An@+	011	register number	X	X	–	X
An@–	100	register number	X	X	–	X
An@(d)	101	register number	X	X	X	X
An@(d, ix)	110	register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
PC@(d)	111	010	X	X	X	–
PC@(d, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

TABLE 21 — INSTRUCTION SET

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
ABCD	Add Decimal with Extend	(Destination)$_{10}$ + (Source)$_{10}$ → Destination	*	U	*	U	*
ADD	Add Binary	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Add Address	(Destination) + (Source) → Destination	–	–	–	–	–
ADDI	Add Immediate	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDQ	Add Quick	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDX	Add Extended	(Destination) + (Source) + X → Destination	*	*	*	*	*
AND	AND Logical	(Destination) Λ (Source) → Destination	–	*	*	0	0
ANDI	AND Immediate	(Destination) Λ Immediate Data → Destination	–	*	*	0	0
ASL, ASR	Arithmetic Shift	(Destination) Shifted by <count> → Destination	*	*	*	*	*
B$_{CC}$	Branch Conditionally	If $_{CC}$ then PC + d → PC	–	–	–	–	–
BCHG	Test a Bit and Change	~(<bit number>) OF Destination → Z ~(<bit number>) OF Destination → <bit number> OF Destination	–	–	*	–	–
BCLR	Test a Bit and Clear	~(<bit number>) OF Destination → Z 0 → <bit number> → OF Destination	–	–	*	–	–
BRA	Branch Always	PC + d → PC	–	–	–	–	–
BSET	Test a Bit and Set	~(<bit number>) OF Destination → Z 1 → <bit number> OF Destination	–	–	*	–	–
BSR	Branch to Subroutine	PC → SP@ – ; PC + d → PC	–	–	–	–	–
BTST	Test a Bit	~(<bit number>) OF Destination → Z	–	–	*	–	–
CHK	Check Register against Bounds	If Dn <0 or Dn> (<ea>) then TRAP	–	*	U	U	U
CLR	Clear an Operand	0 → Destination	–	0	1	0	0
CMP	Compare	(Destination) – (Source)	–	*	*	*	*
CMPA	Compare Address	(Destination) – (Source)	–	*	*	*	*
CMPI	Compare Immediate	(Destination) – Immediate Data	–	*	*	*	*
CMPM	Compare Memory	(Destination) – (Source)	–	*	*	*	*
DB$_{CC}$	Test Condition, Decrement and Branch	If ~$_{CC}$ then Dn – 1 → Dn; if Dn ≠ – 1 then PC + d → PC	–	–	–	–	–
DIVS	Signed Divide	(Destination)/(Source) → Destination	–	*	*	*	0
DIVU	Unsigned Divide	(Destination)/(Source) → Destination	–	*	*	*	0
EOR	Exclusive OR Logical	(Destination) ⊕ (Source) → Destination	–	*	*	0	0
EORI	Exclusive OR Immediate	(Destination) ⊕ Immediate Data → Destination	–	*	*	0	0
EXG	Exchange Register	Rx ↔ Ry	–	–	–	–	–
EXT	Sign Extend	(Destination) Sign-extended → Destination	–	*	*	0	0
JMP	Jump	Destination → PC	–	–	–	–	–
JSR	Jump to Subroutine	PC → SP@ – ; Destination → PC	–	–	–	–	–
LEA	Load Effective Address	Destination → An	–	–	–	–	–
LINK	Link and Allocate	An → SP@ – ; SP → An; SP + d → SP	–	–	–	–	–
LSL, LSR	Logical Shift	(Destination) Shifted by <count> → Destination	*	*	*	0	*
MOVE	Move Data from Source to Destination	(Source) → Destination	–	*	*	0	0
MOVE to CCR	Move to Condition Code	(Source) → CCR	*	*	*	*	*
MOVE to SR	Move to the Status Register	(Source) → SR	*	*	*	*	*

* affected 0 cleared U defined
– unaffected 1 set

TABLE 21 — INSTRUCTION SET (CONTINUED)

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
MOVE from SR	Move from the Status Register	SR → Destination	–	–	–	–	–
MOVE USP	Move User Stack Pointer	USP → An; An → USP	–	–	–	–	–
MOVEA	Move Address	(Source) → Destination	–	–	–	–	–
MOVEM	Move Multiple Registers	Registers → Destination (Source) → Registers	–	–	–	–	–
MOVEP	Move Peripheral Data	(Source) → Destination	–	–	–	–	–
MOVEQ	Move Quick	Immediate Data → Destination	–	*	*	0	0
MULS	Signed Multiply	(Destination) * (Source) → Destination	–	*	*	0	0
MULU	Unsigned Multiply	(Destination) * (Source) → Destination	–	*	*	0	0
NBCD	Negate Decimal with Extend	$0-(\text{Destination})_{10}-X \rightarrow$ Destination	*	U	*	U	*
NEG	Negate	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Negate with Extend	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	No Operation	–	–	–	–	–	–
NOT	Logical Complement	~(Destination) → Destination	–	*	*	0	0
OR	Inclusive OR Logical	(Destination) v (Source) → Destination	–	*	*	0	0
ORI	Inclusive OR Immediate	(Destination) v Immediate Data → Destination	–	*	*	0	0
PEA	Push Effective Address	Destination → SP@ –	–	–	–	–	–
RESET	Reset External Devices	–	–	–	–	–	–
ROL, ROR	Rotate (Without Extend)	(Destination) Rotated by <count> → Destination	–	*	*	0	*
ROXL, ROXR	Rotate with Extend	(Destination) Rotated by <count> → Destination	*	*	*	0	*
RTE	Return from Exception	SP@ – → SR; SP@ + → PC	*	*	*	*	*
RTR	Return and Restore Condition Codes	SP@ + → CC; SP@ + → PC	*	*	*	*	*
RTS	Return from Subroutine	SP@ + → PC	–	–	–	–	–
SBCD	Subtract Decimal with Extend	$(\text{Destination})_{10}-(\text{Source})_{10}-X \rightarrow$ Destination	*	U	*	U	*
S_{CC}	Set According to Condition	If CC then 1's → Destination else 0's → Destination	–	–	–	–	–
STOP	Load Status Register and Stop	Immediate Data → SR; STOP	*	*	*	*	*
SUB	Subtract Binary	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Subtract Address	(Destination) – (Source) → Destination	–	–	–	–	–
SUBI	Subtract Immediate	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBQ	Subtract Quick	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBX	Subtract with Extend	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Swap Register Halves	Register [31:16] ↔ Register [15:0]	–	*	*	0	0
TAS	Test and Set an Operand	(Destination) Tested → CC; 1 → [7] OF Destination	–	*	*	0	0
TRAP	Trap	PC → SSP@ – ; SR → SSP@ – ; (Vector) → PC	–	–	–	–	–
TRAPV	Trap on Overflow	If V then TRAP	–	–	–	–	–
TST	Test an Operand	(Destination) Tested → CC	–	*	*	0	0
UNLK	Unlink	An → SP; SP@ + → An	–	–	–	–	–

[ ] = bit number

INSTRUCTION EXECUTION TIMES

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as: (r/w) where r is the number of read cycles and w is the number of write cycles.

NOTE

The number of periods includes instruction fetch and all applicable operand fetches and stores.

EFFECTIVE ADDRESS OPERAND CALCULATION TIMING

Table 23 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand. The number of bus read and write cycles is shown in parenthesis as (r/w). Note there are no write cycles involved in processing the effective address.

MOVE INSTRUCTION CLOCK PERIODS

Tables 24 and 25 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as: (r/w).

STANDARD INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 26 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 26 the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

IMMEDIATE INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 27 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 27, the headings have the following meanings: # = immediate operand, Dn = data register operand, An = address register operand, M = memory operand, and SR = status register.

SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Table 28 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

TABLE 23 — EFFECTIVE ADDRESS CALCULATION TIMING

Addressing Mode		Byte, Word	Long
	Register		
Dn	Data Register Direct	0(0/0)	0(0/0)
An	Address Register Direct	0(0/0)	0(0/0)
	Memory		
An@	Address Register Indirect	4(1/0)	8(2/0)
An@+	Address Register Indirect with Postincrement	4(1/0)	8(2/0)
An@–	Address Register Indirect with Predecrement	6(1/0)	10(2/0)
An@(d)	Address Register Indirect with Displacement	8(2/0)	12(3/0)
An@(d, ix)*	Address Register Indirect with Index	10(2/0)	14(3/0)
xxx.W	Absolute Short	8(2/0)	12(3/0)
xxx.L	Absolute Long	12(3/0)	16(4/0)
PC@(d)	Program Counter with Displacement	8(2/0)	12(3/0)
PC@(d, ix)*	Program Counter with Index	10(2/0)	14(3/0)
#xxx	Immediate	4(1/0)	8(2/0)

*The size of the index register (ix) does not affect execution time.

TABLE 24 — MOVE BYTE AND WORD INSTRUCTION CLOCK PERIODS

Source	Destination								
	Dn	An	An@	An@+	An@−	An@(d)	An@(d,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An@	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@−	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
An@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
An@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
PC@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
PC@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

The size of the index register (ix) does not affect execution time.

TABLE 25 — MOVE LONG INSTRUCTION CLOCK PERIODS

Source	Destination								
	Dn	An	An@	An@+	An@−	An@(d)	An@(d,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An@	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@−	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
An@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
An@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
PC@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(5/2)
PC@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

*The size of the index register (ix) does not affect execution time.

TABLE 26 — STANDARD INSTRUCTION CLOCK PERIODS

Instruction	Size	op <ea>, An	op <ea>, Dn	op Dn, <M>
ADD	Byte, Word	8(1/0)+	4(1/0)+	8(1/1)+
	Long	6(1/0)+**	6(1/0)+**	12(1/2)+
AND	Byte, Word	—	4(1/0)+	8(1/1)+
	Long	—	6(1/0)+**	12(1/2)+
CMP	Byte, Word	6(1/0)+	4(1/0)+	—
	Long	6(1/0)+	6(1/0)+	—
DIVS	—	—	158(1/0)+*	—
DIVU	—	—	140(1/0)+*	—
EOR	Byte, Word	—	4(1/0)***	8(1/1)+
	Long	—	8(1/0)***	12(1/2)+
MULS	—	—	70(1/0)+*	—
MULU	—	—	70(1/0)+*	—
OR	Byte, Word	—	4(1/0)+	8(1/1)+
	Long	—	6(1/0)+**	12(1/1)+
SUB	Byte, Word	8(1/0)+	4(1/0)+	8(1/1)+
	Long	6(1/0)+**	6(1/0)+**	12(1/2)+

+ add effective address calculation time
** total of 8 clock periods for instruction if the effective address is register direct
* indicates maximum value
*** only available effective address mode is data register direct

TABLE 27 — IMMEDIATE INSTRUCTION CLOCK PERIODS

Instruction	Size	op #, Dn	op #, An	op #, M
ADDI	Byte, Word	8(2/0)	—	12(2/1)+
	Long	16(3/0)	—	20(3/2)+
ADDQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1)+
	Long	8(1/0)	8(1/0)	12(1/2)+
ANDI	Byte, Word	8(2/0)	—	12(2/1)+
	Long	16(3/0)	—	20(3/1)+
CMPI	Byte, Word	8(2/0)	8(2/0)	8(2/0)+
	Long	14(3/0)	14(3/0)	12(3/0)+
EORI	Byte, Word	8(2/0)	—	12(2/1)+
	Long	16(3/0)	—	20(3/2)+
MOVEQ	Long	4(1/0)	—	—
ORI	Byte, Word	8(2/0)	—	12(2/1)+
	Long	16(3/0)	—	20(3/2)+
SUBI	Byte, Word	8(2/0)	—	12(2/1)+
	Long	16(3/0)	—	20(3/2)+
SUBQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1)+
	Long	8(1/0)	8(1/0)	12(1/2)+

+ add effective address calculation time
*word only

TABLE 28 — SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Instruction	Size	Register	Memory
CLR	Byte, Word	4(1/0)	8(1/1)+
	Long	6(1/0)	12(1/2)+
NBCD	Byte	6(1/0)	8(1/1)+
NEG	Byte, Word	4(1/0)	8(1/1)+
	Long	6(1/0)	12(1/2)+
NEGX	Byte, Word	4(1/0)	8(1/1)+
	Long	6(1/0)	12(1/2)+
NOT	Byte, Word	4(1/0)	8(1/1)+
	Long	6(1/0)	12(1/2)+
Scc	Byte, False	4(1/0)	8(1/1)+
	Byte, True	6(1/0)	8(1/1)+
TAS	Byte	4(1/0)	10(1/1)+
TST	Byte, Word	4(1/0)	4(1/0)
	Long	4(1/0)	4(1/0)+

+ add effective address calculation time

SHIFT/ROTATE INSTRUCTION CLOCK PERIODS

Table 29 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

BIT MANIPULATION INSTRUCTION CLOCK PERIODS

Table 30 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

CONDITIONAL INSTRUCTION CLOCK PERIODS

Table 31 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS

Table 32 indicates the number of clock periods required for the jump, jump to subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w).

TABLE 29 — SHIFT/ROTATE INSTRUCTION CLOCK PERIODS

Instruction	Size	Register	Memory
ASR, ASL	Byte, Word	6 + 2n(1/0)	8(1/1)+
	Long	8 + 2n(1/0)	—
LSR, LSL	Byte, Word	6 + 2n(1/0)	8(1/1)+
	Long	8 + 2n(1/0)	—
ROR, ROL	Byte, Word	6 + 2n(1/0)	8(1/1)+
	Long	8 + 2n(1/0)	—
ROXR, ROXL	Byte, Word	6 + 2n(1/0)	8(1/1)+
	Long	8 + 2n(1/0)	—

TABLE 30 — BIT MANIPULATION INSTRUCTION CLOCK PERIODS

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	—	8(1/1)+	—	12(2/1)+
	Long	8(1/0)*	—	12(2/0)*	—
BCLR	Byte	—	8(1/1)+	—	12(2/1)+
	Long	10(1/0)*	—	14(2/0)*	—
BSET	Byte	—	8(1/1)+	—	12(2/1)+
	Long	8(1/0)*	—	12(2/0)*	—
BTST	Byte	—	4(1/0)+	—	8(2/0)+
	Long	6(1/0)	—	10(2/0)	—

+ add effective address calculation time

* indicates maximum value

TABLE 31 — CONDITIONAL INSTRUCTION CLOCK PERIODS

Instruction	Displacement	Trap or Branch Taken	Trap or Branch Not Taken
B_{CC}	Byte	10(2/0)	8(1/0)
	Word	10(2/0)	12(2/0)
BRA	Byte	10(2/0)	—
	Word	10(2/0)	—
BSR	Byte	18(2/2)	—
	Word	18(2/2)	—
DB_{CC}	CC true	—	12(2/0)
	CC false	10(2/0)	14(3/0)
CHK	—	40(5/3)+ *	8(1/0)+
TRAP	—	34(4/3)	—
TRAPV	—	34(5/3)	4(1/0)

+ add effective address calculation time

* indicates maximum value

TABLE 32 — JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS

Instr	Size	An@	An@+	An@−	An@(d)	An@(d, ix)*	xxx.W	xxx.L	PC@(d)	PC@(d, ix)*
JMP	—	8(2/0)	—	—	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	—	16(2/2)	—	—	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	—	4(1/0)	—	—	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	—	12(1/2)	—	—	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM M→R	Word	12+4n (3+n/0)	12+4n (3+n/0)	— —	16+4n (4+n/0)	18+4n (4+n/0)	16+4n (4+n/0)	20+4n (5+n/0)	16+4n (4+n/0)	18+4n (4+n/0)
	Long	12+8n (3+2n/0)	12+8n (3+2n/0)	— —	16+8n (4+2n/0)	18+8n (4+2n/0)	16+8n (4+2n/0)	20+8n (5+2n/0)	16+8n (4+2n/0)	18+8n (4+2n/0)
MOVEM R→M	Word	8+5n (2/n)	— —	8+5n (2/n)	12+5n (3/n)	14+5n (3/n)	12+5n (3/n)	16+5n (4/n)	— —	— —
	Long	8+10n (2/2n)	— —	8+10n (2/2n)	12+10n (3/2n)	14+10n (3/2n)	12+10n (3/2n)	16+10n (4/2n)	— —	— —

n is the number of registers to move

* is the size of the index register (ix) does not affect the instruction's execution time

MULTI-PRECISION INSTRUCTION CLOCK PERIODS

Table 33 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as: (r/w).

In Table 33, the headings have the following meanings: Dn = data register operand and M = memory operand.

TABLE 33 — MULTI-PRECISION INSTRUCTION CLOCK PERIODS

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
CMPM	Byte, Word	—	12(3/0)
	Long	—	20(5/0)
SUBX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
ABCD	Byte	6(1/0)	18(3/1)
SBCD	Byte	6(1/0)	18(3/1)

MISCELLANEOUS INSTRUCTION CLOCK PERIODS

Table 34 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

EXCEPTION PROCESSING CLOCK PERIODS

Table 35 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as: (r/w).

TABLE 34 — MISCELLANEOUS INSTRUCTION CLOCK PERIODS

Instruction	Size	Register	Memory	Register → Memory	Memory → Register
MOVE from SR	—	6(1/0)	8(1/1)+	—	—
MOVE to CCR	—	12(2/0)	12(2/0)+	—	—
MOVE to SR	—	12(2/0)	12(2/0)+	—	—
MOVEP	Word	—	—	16(2/2)	16(4/0)
	Long	—	—	24(2/4)	24(6/0)
EXG	—	6(1/0)	—	—	—
EXT	Word	4(1/0)	—	—	—
	Long	4(1/0)	—	—	—
LINK	—	16(2/2)	—	—	—
MOVE from USP	—	4(1/0)	—	—	—
MOVE to USP	—	4(1/0)	—	—	—
NOP	—	4(1/0)	—	—	—
RESET	—	132(1/0)	—	—	—
RTE	—	20(5/0)	—	—	—
RTR	—	20(5/0)	—	—	—
RTS	—	16(4/0)	—	—	—
STOP	—	4(0/0)	—	—	—
SWAP	—	4(1/0)	—	—	—
UNLK	—	12(3/0)	—	—	—

+ add effective address calculation time

TABLE 35 — EXCEPTION PROCESSING CLOCK PERIODS

Exception	Periods
Address Error	50(4/7)
Bus Error	50(4/7)
Interrupt	44(5/3)*
Illegal Instruction	34(4/3)
Privileged Instruction	34(4/3)
Trace	34(4/3)

* The interrupt acknowledge bus cycle is assumed to take four external clock periods

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6821 (1.0 MHz)
MC68A21 (1.5 MHz)
MC68B21 (2.0 MHz)

MOS
(N-CHANNEL, SILICON-GATE, DEPLETION LOAD)

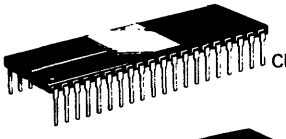
PERIPHERAL INTERFACE ADAPTER

PERIPHERAL INTERFACE ADAPTER (PIA)

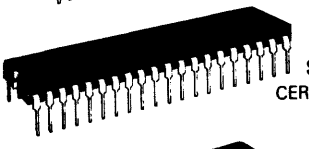
The MC6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the M6800 family of microprocessors. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the overall operation of the interface.

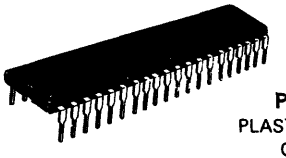
- 8-Bit Bidirectional Data Bus for Communication with the MPU
- Two Bidirectional 8-Bit Buses for Interface to Peripherals
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines; Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance Three-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- TTL-Compatible
- Static Operation



L SUFFIX
CERAMIC PACKAGE
CASE 715

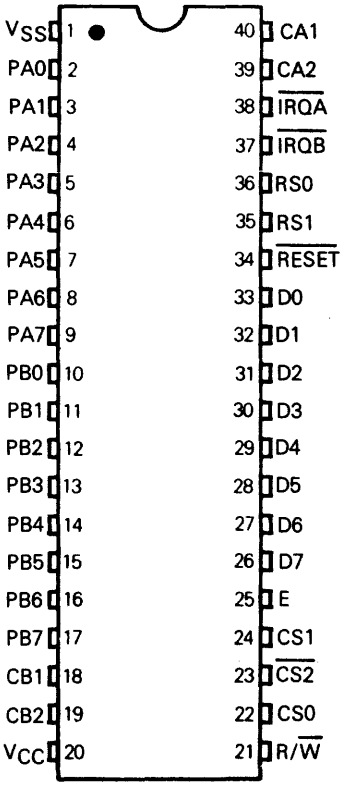


S SUFFIX
CERDIP PACKAGE
CASE 734



P SUFFIX
PLASTIC PACKAGE
CASE 711

PIN ASSIGNMENT



MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6821, MC68A21, MC68B21 MC6821C, MC68A21C, MC68B21C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

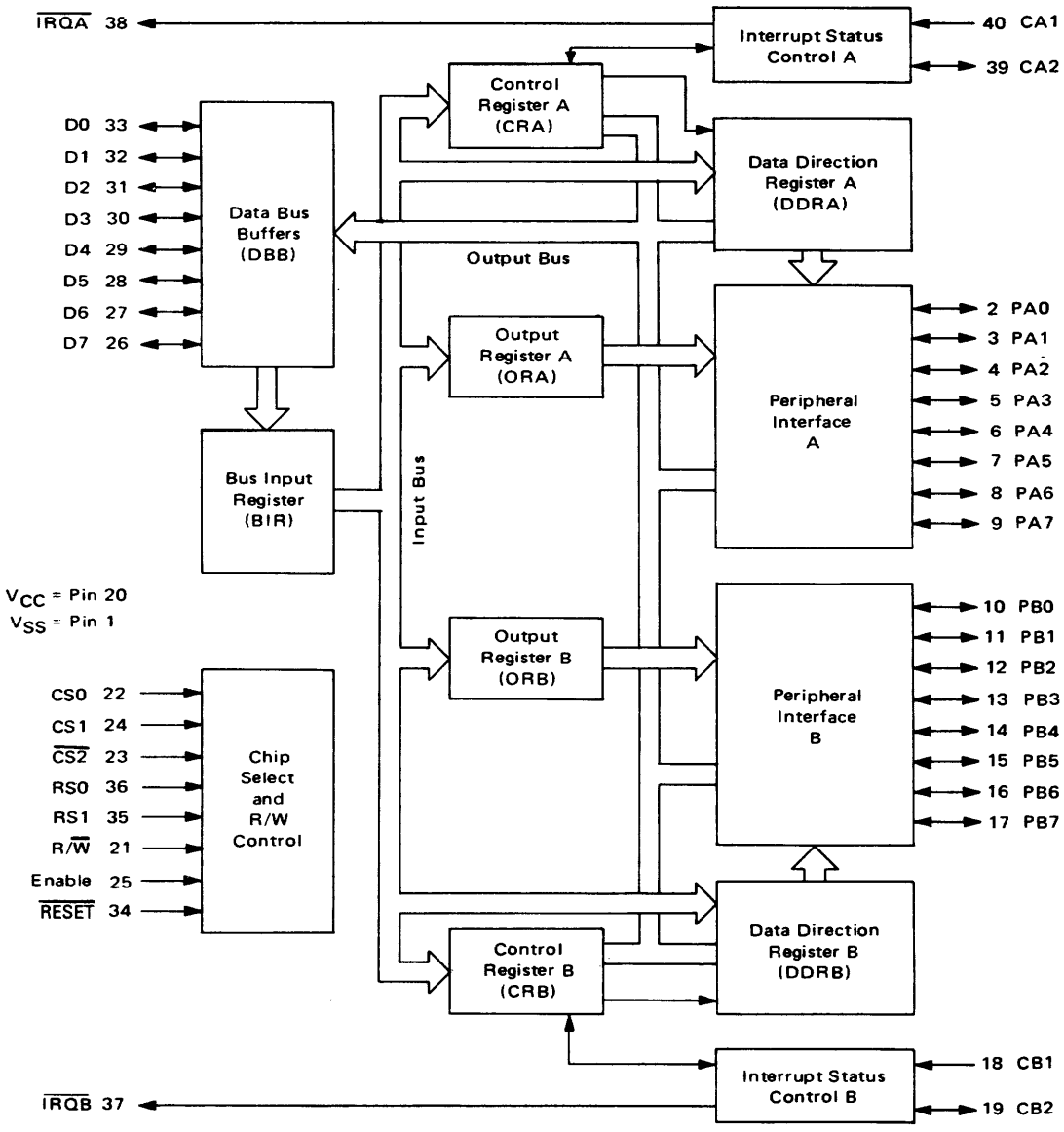
Characteristic	Symbol	Value	Unit
Thermal Resistance Ceramic Plastic Cerdip	θ_{JA}	 50 100 60	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (i.e., either V_{SS} or V_{CC}).

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, $T_A = T_L$ to T_H unless otherwise noted).

Characteristic		Symbol	Min	Typ	Max	Unit
INTERRUPT OUTPUTS ($\overline{IRQA}$, $\overline{IRQB}$)						
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	V_{SS} + 0.4	V
Three-State Output Leakage Current		I_{OZ}	–	1.0	10	μA
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{out}	–	–	5.0	pF
PERIPHERAL BUS (PA0-PA7, PB0-PB7, CA1, CA2, CB1, CB2)						
Input Leakage Current (V_{in} = 0 to 5.25 V)	R/$\overline{W}$, $\overline{RESET}$, RS0, RS1, CS0, CS1, $\overline{CS2}$, CA1, CB1, Enable	I_{in}	–	1.0	2.5	μA
Three-State Input Leakage Current (V_{in} = 0.4 to 2.4 V)	PB0-PB7, CB2	I_{IZ}	–	2.0	10	μA
Input High Current (V_{IH} = 2.4 V)	PA0-PA7, CA2	I_{IH}	–200	–400	–	μA
Darlington Drive Current (V_O = 1.5 V)	PB0-PB7, CB2	I_{OH}	–1.0	–	–10	mA
Input Low Current (V_{IL} = 0.4 V)	PA0-PA7, CA2	I_{IL}	–	–1.3	–2.4	mA
Output High Voltage (I_{Load} = –200 μA) (I_{Load} = –10 μA)	PA0-PA7, PB0-PB7, CA2, CB2 PA0-PA7, CA2	V_{OH}	V_{SS} + 2.4 V_{CC} – 1.0	– –	– –	V
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	V_{SS} + 0.4	V
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	10	pF

FIGURE 16 — EXPANDED BLOCK DIAGRAM



PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bidirectional data buses and four interrupt/control lines for interfacing to peripheral devices.

Section A Peripheral Data (PA0-PA7) — Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines. In the input mode, the internal pullup resistor on these lines represents a maximum of 1.5 standard TTL loads.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "high" on the corresponding data line while a "0" results in a "low." Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read property if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

Section B Peripheral Data (PB0-PB7) — The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to PA0-PA7. They have three-state capabiity, allowing them to enter a high-impedance state when the peripheral data line is used as an input. In addition, data on the peripheral data lines PB0-PB7 will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "high" or above 0.8 V for a "low". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch.

Interrupt Input (CA1 and CB1) — Peripheral input lines CA1 and CB1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

Peripheral Control (CA2) — The peripheral control line CA2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL; as an input the internal pullup resistor on this line represents 1.5 standard TTL loads. The function of this signal line is programmed with Control Register A.

Peripheral Control (CB2) — Peripheral Control line CB2 may also be programmed to act as an interrupt input or peripheral control output. As an input, this line has high input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

INTERNAL CONTROLS

INITIALIZATION

A $\overline{\text{RESET}}$ has the effect of zeroing all PIA registers. This will set PA0-PA7, PB0-PB7, CA2 and CB2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS0 and RS1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Details of possible configurations of the Data Direction and Control Register are as follows:

TABLE 1 – INTERNAL ADDRESSING

RS1	RS0	Control Register Bit CRA-2	Control Register Bit CRB-2	Location Selected
0	0	1	X	Peripheral Register A
0	0	0	X	Data Direction Register A
0	1	X	X	Control Register A
1	0	X	1	Peripheral Register B
1	0	X	0	Data Direction Register B
1	1	X	X	Control Register B

X = Don't Care

PORT A-B HARDWARE CHARACTERISTICS

As shown in Figure 17, the MC6821 has a pair of I/O ports whose characteristics differ greatly. The A side is designed to drive CMOS logic to normal 30% to 70% levels, and incorporates an internal pullup device that remains connected even in the input mode. Because of this, the A side requires more drive current in the input mode than Port B. In contrast, the B side uses a normal three-state NMOS buffer which cannot pullup to CMOS levels without external resistors. The B side can drive extra loads such as Darlingtons without problem. When the PIA comes out of reset, the A port represents inputs with pullup resistors, whereas the B side (input mode also) will float high or low, depending upon the load connected to it.

Notice the differences between a Port A and Port B read operation when in the output mode. When reading Port A, the actual pin is read, whereas the B side read comes from an output latch, ahead of the actual pin.

CONTROL REGISTERS (CRA and CRB)

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA1, CA2, CB1, and CB2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be writ-

ten or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA1, CA2, CB1, or CB2. The format of the control words is shown in Figure 18.

DATA DIRECTION ACCESS CONTROL BIT (CRA-2 and CRB-2)

Bit 2, in each Control Register (CRA and CRB), determines selection of either a Peripheral Output Register or the corresponding Data Direction E Register when the proper register select signals are applied to RS0 and RS1. A "1" in bit 2 allows access of the Peripheral Interface Register, while a "0" causes the Data Direction Register to be addressed.

Interrupt Flags (CRA-6, CRA-7, CRB-6, and CRB-7) — The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

Control of CA2 and CB2 Peripheral Control Lines (CRA-3, CRA-4, CRA-5, CRB-3, CRB-4, and CRB-5) — Bits 3, 4, and 5 of the two control registers are used to control the CA2 and CB2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA-5 (CRB-5) is low, CA2 (CB2) is an interrupt input line similar to CA1 (CB1). When CRA-5 (CRB-5) is high, CA2 (CB2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA2 and CB2 have slightly different loading characteristics.

Control of CA1 and CB1 Interrupt Input Lines (CRA-0, CRB-1, CRA-1, and CRB-1) — The two lowest-order bits of the control registers are used to control the interrupt input lines CA1 and CB1. Bits CRA-0 and CRB-0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA-1 and CRB-1 determine the active transition of the interrupt input signals CA1 and CB1.

FIGURE 17 — PORT A AND PORT B EQUIVALENT CIRCUITS

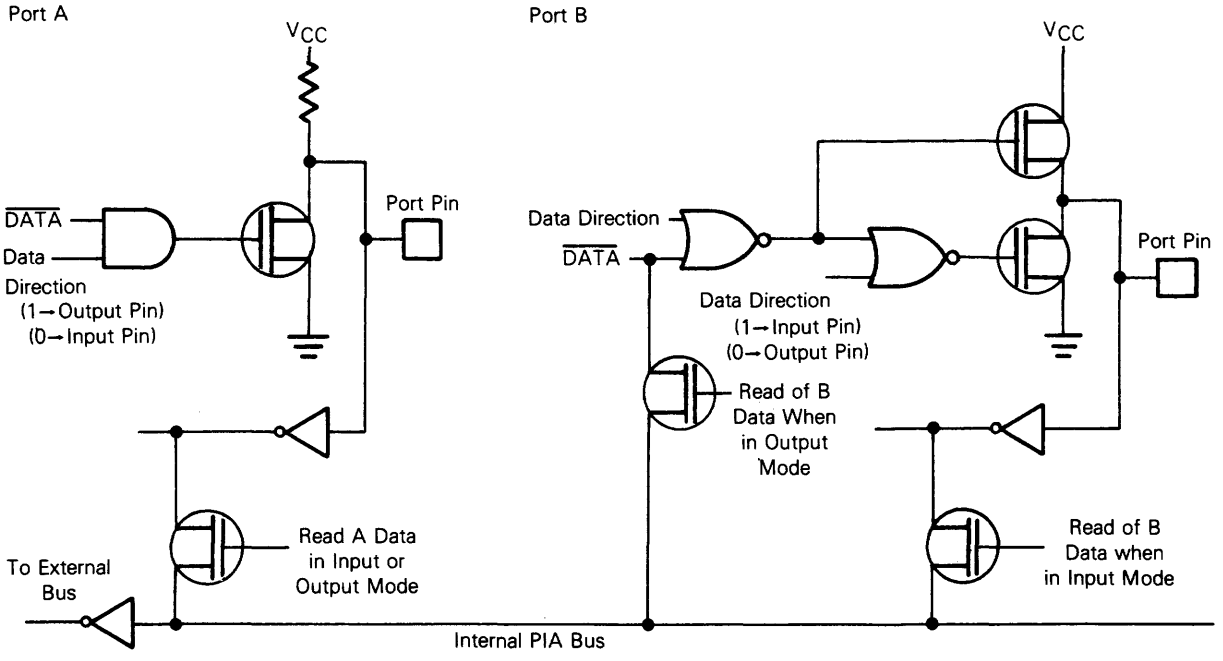


FIGURE 18 — CONTROL WORD FORMAT

Determine Active CA1 (CB1) Transition for Setting Interrupt Flag IRQA(B)1 — (bit 7)

b1 = 0: IRQA(B)1 set by high-to-low transition on CA1 (CB1)

b1 = 1: IRQA(B)1 set by low-to-high transition on CA1 (CB1).

CA1 (CB1) Interrupt Request Enable/Disable

b0 = 0: Disables IRQA(B) MPU Interrupt by CA1 (CB1) active transition.[1]

b0 = 1: Enable IRQA(B) MPU Interrupt by CA1 (CB1) active transition.

1. IRQA(B) will occur on next (MPU generated) positive transition of b0 if CA1 (CB1) active transition occurred while interrupt was disabled.

IRQA(B) 1 Interrupt Flag (bit 7)

Goes high on active transition of CA1 (CB1); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

Control Register

b7	b6	b5	b4	b3	b2	b1	b0
IRQA(B)1 Flag	IRQA(B)2 Flag	CA2 (CB2) Control			DDR Access	CA1 (CB1) Control	

IRQA(B)2 Interrupt Flag (bit 6)

When CA2 (CB2) is an input, IRQA(B) goes high on active transition CA2 (CB2); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

CA2 (CB2) Established as Output (b5 = 1): IRQA(B) 2 = 0, not affected by CA2 (CB2) transitions.

Determines Whether Data Direction Register Or Output Register is Addressed

b2 = 0: Data Direction Register selected.

b2 = 1: Output Register selected.

, CA2 (CB2) Established as Output by b5 = 1

(Note that operation of CA2 and CB2 output functions are not identical)

b5 b4 b3

1 0 → CA2

b3 = 0: **Read Strobe with CA1 Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next active CA1 transition, as specified by bit 1.

b3 = 1: **Read Strobe with E Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next high-to-low E transition during a deselect.

→ CB2

b3 = 0: **Write Strobe with CB1 Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next active CB1 transition as specified by bit 1. CRB-b7 must first be cleared by a read of data.

b3 = 1: **Write Strobe with E Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next low-to-high E transition following an E pulse which occurred while the part was deselected.

b5 b4 b3

1 1 → **Set/Reset CA2 (CB2)**

CA2 (CB2) goes low as MPU writes b3 = 0 into Control Register.

CA2 (CB2) goes high as MPU writes b3 = 1 into Control Register.

CA2 (CB2) Established as Input by b5 = 0

b5 b4 b3

0 → **CA2 (CB2) Interrupt Request Enable/Disable**

b3 = 0: Disables IRQA(A) MPU Interrupt by CA2 (CB2) active transition.*

b3 = 1: Enables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.

*IRQA(B) will occur on next (MPU generatted) positive transition of b3 if CA2 (CB2) active transition occurred while interrupt was disabled.

→ **Determines Active CA2 (CB2) Transition for Setting Interrupt Flag IRQA(B)2 — (Bit b6)**

b4 = 0: IRQA(B)2 set by high-to-low transition on CA2 (CB2).

b4 = 1: IRQA(B)2 set by low-to-high transition on CA2 (CB2).

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6840 (1.0 MHz)
MC68A40 (1.5 MHz)
MC68B40 (2.0 MHz)

PROGRAMMABLE TIMER MODULE (PTM)

The MC6840 is a programmable subsystem component of the M6800 family designed to provide variable system time intervals.

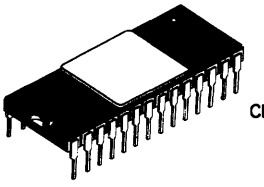
The MC6840 has three 16-bit binary counters, three corresponding control registers, and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The MC6840 may be utilized for such tasks as frequency measurements, event counting, interval measuring, and similar tasks. The device may be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation as well as system interrupts.

- Operates from a Single 5 Volt Power Supply
- Fully TTL Compatible
- Single System Clock Required (Enable)
- Selectable Prescaler on Timer 3 Capable of 4 MHz for the MC6840, 6 MHz for the MC68A40 and 8 MHz for the MC68B40
- Programmable Interrupts ($\overline{IRQ}$) Output to MPU
- Readable Down Counter Indicates Counts to Go Until Time-Out
- Selectable Gating for Frequency or Pulse-Width Comparison
- $\overline{RESET}$ Input
- Three Asynchronous External Clock and Gate/Trigger Inputs Internally Synchronized
- Three Maskable Outputs

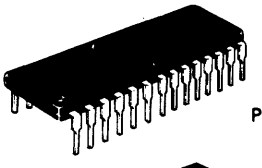
MOS

(N-CHANNEL, SILICON-GATE DEPLETION LOAD)

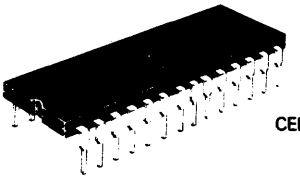
PROGRAMMABLE TIMER



L SUFFIX
CERAMIC PACKAGE
CASE 719



P SUFFIX
PLASTIC PACKAGE
CASE 710



S SUFFIX
CERDIP PACKAGE
CASE 733

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range — T_L to T_h MC6840, MC68A40, MC68B40 MC6840C, MC68A40C	T_A	0 to +70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Cerdip Plastic Ceramic	θ_{JA}	 65 115 60	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

FIGURE 1 — PIN ASSIGNMENT

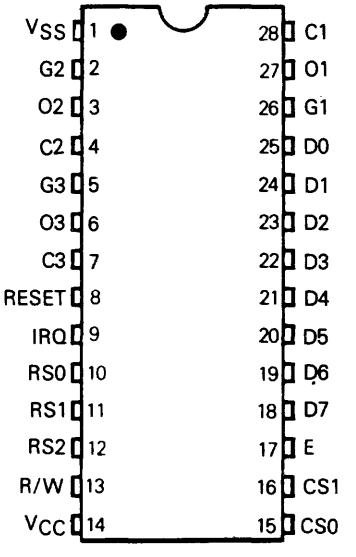


FIGURE 2 — BLOCK DIAGRAM

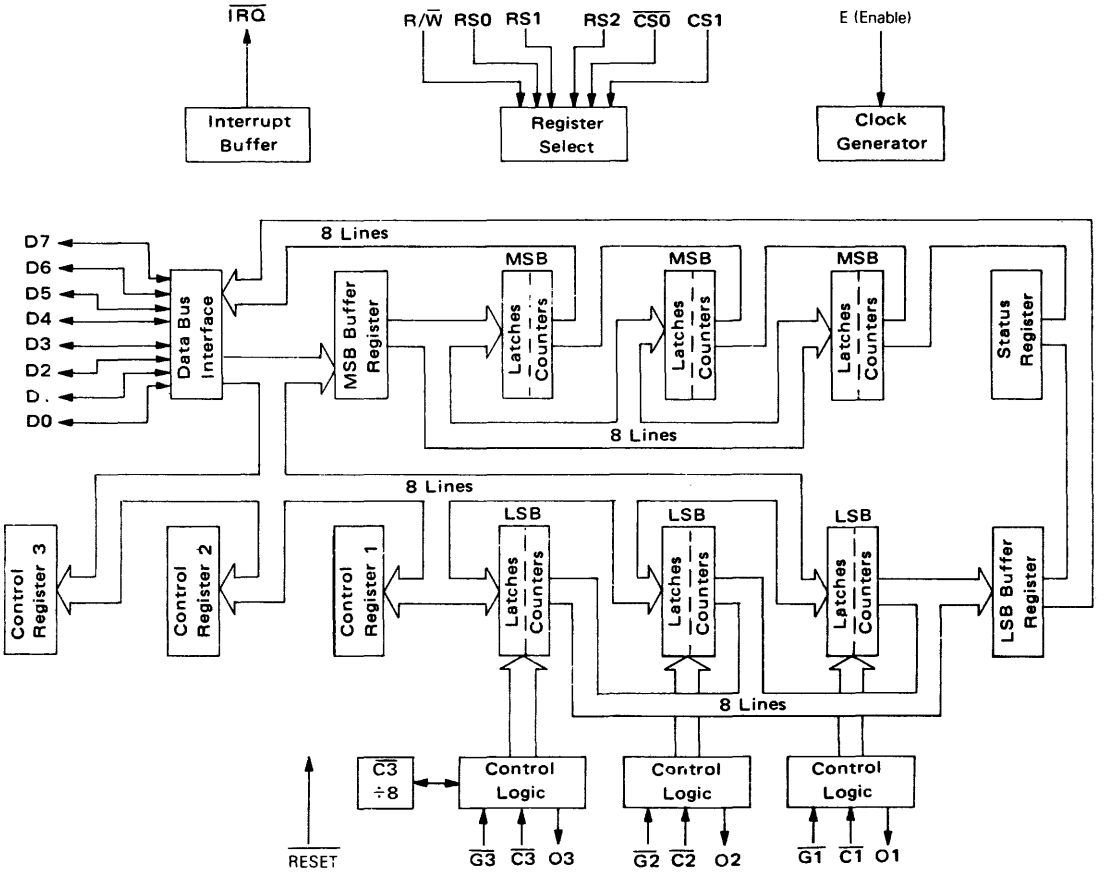


TABLE 8 — FREQUENCY COMPARISON MODE

Mode	Bit 3	Bit 4	Control Reg. Bit 5	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
Frequency	1	0	0	$\overline{G1}\downarrow \cdot I \pm (\overline{CE} + TO) + R$	$\overline{G1}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	$W + R + I$	$\overline{G1}\downarrow$ Before TO
Comparison	1	0	1	$\overline{G1}\downarrow \cdot \overline{I} + R$	$\overline{G1}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	$W + R + I$	TO Before $\overline{G1}\downarrow$
Pulse Width	1	1	0	$\overline{G1}\downarrow \cdot \overline{I} + \overline{R}$	$\overline{G1}\downarrow \overline{W} \cdot \overline{R} \cdot \overline{I}$	$W + R + I + G$	$\overline{G1}\uparrow$ Before TO
Comparison	1	1	1	$\overline{G1}\downarrow \cdot \overline{I} + \overline{R}$	$\overline{G1}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	$W + R + I + G$	$\overline{G1}\uparrow$ Before TO

$\overline{G1}\downarrow$ = Negative transition of $\overline{\text{Gate}}$ input.
W = Write Timer Latches Command.
R = Timer Reset (CR10 = 1 or External $\overline{\text{RESET}} = 0$).
N = 16-Bit Number in Counter Latch.
TO = Counter Time Out (All Zero Condition)
I = Interrupt for a given timer.

*All time intervals shown above assume the Gate ($\overline{G}$) and Clock ($\overline{C}$) signals are sycnhronized to the system clock (E) with the specified setup and hold time requirements.

DEVICE OPERATION

The MC6840 is part of the M6800 microprocessor family and is fully bus compatible with M6800 systems. The three timers in the MC6840 operate independently and in several distinct modes to fit a wide variety of measurement and synthesis applications.

The MC6840 is an integrated set of three distinct counter/timers (Figure 1). It consists of three 16-bit data latches, three 16-bit counters (clocked independently), and the comparison and enable circuitry necessary to implement various measurement and synthesis functions. In addition, it contains interrupt drivers to alert the processor that a particular function has been completed.

In a typical application, a timer will be loaded by first storing two bytes of data into an associated Counter Latch. This data is then transferred into the counter via a Counter Initialization cycle. If the counter is enabled, the counter decrements on each subsequent clock period which may be an external clock, or Enable (E) until one of several predetermined conditions causes it to halt or recycle. The timers are thus programmable, cyclic in nature, controllable by external inputs or the MPU program, and accessible by the MPU at any time.

BUS INTERFACE

The Programmable Timer Module (PTM) interfaces to the M6800 Bus with an 8-bit bidirectional data bus, two Chip Select lines, a Read/Write line, a clock (Enable) line, and Interrupt Request line, an external Reset line, and three Register select lines. VMA should be utilized in conjunction with an MPU address line into a Chip Select of the PTM when using the MC6800/6802/6808.

BIDIRECTIONAL DATA (D0-D7) — The bidirectional data lines (D0-D7) allow the transfer of data between the MPU and PTM. The data bus output drivers are three-state devices which remain in the high-impedance (off) state except when the MPU performs a PTM read operation (Read/Write and Enable lines high and PTM Chip Selects activated).

CHIP SELECT ($\overline{CS0}$, CS1) — These two signals are used to activate the Data Bus interface and allow transfer of data from the PTM. With $\overline{CS0}=0$ and $CS1=1$, the device is selected and data transfer will occur.

READ/WRITE (R/$\overline{W}$) — This signal is generated by the MPU to control the direction of data transfer on the Data Bus. With the PTM selected, a low state on the PTM R/$\overline{W}$ line enables the input buffers and data is transferred from the MPU to the PTM on the trailing edge of the E (Enable) clock. Alternately, (under the same conditions) R/$\overline{W}=1$ and Enable high allows data in the PTM to be read by the MPU.

ENABLE (E CLOCK) — The E clock signal synchronizes data transfer between the MPU and the PTM. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the PTM.

INTERRUPT REQUEST ($\overline{IRQ}$) — The active low Interrupt Request signal is normally tied directly (or through priority interrupt circuitry) to the $\overline{IRQ}$ input of the MPU. This is an "open drain" output (no load device on the chip) which permits other similar interrupt request lines to be tied together in a wire-OR configuration.

The $\overline{IRQ}$ line is activated if, and only if, the Composite Interrupt Flag (Bit 7 of the Internal Status Register) is asserted. The conditions under which the $\overline{IRQ}$ line is activated are discussed in conjunction with the Status Register.

$\overline{RESET}$ — A low level at this input is clocked into the PTM by the E (Enable) input. Two Enable pulses are required to synchronize and process the signal. The PTM then recognizes the active "low" or inactive "high" on the third Enable pulse. If the $\overline{RESET}$ signal is asynchronous, an additional Enable period is required if setup times are not met. The $\overline{RESET}$ input must be stable High/Low for the minimum time stated in the AC Operating Characteristics.

Recognition of a low level at this input by the PTM causes the following action to occur:

a. All counter latches are preset to their maximum count values.
b. All Control Register bits are cleared with the exception of CR10 (internal reset bit) which is set.
c. All counters are preset to the contents of the latches.
d. All counter outputs are reset and all counter clocks are disabled.
e. All Status Register bits (interrupt flags) are cleared.

REGISTER SELECT LINES (RS0, RS1, RS2) — These inputs are used in conjunction with the R/$\overline{W}$ line to select the internal registers, counters and latches as shown in Table 1.

NOTE:

The PTM is accessed via MPU Load and Store operations in much the same manner as a memory device. The instructions available with the M6800 family of MPUs which perform read-modify-write operations on memory should not be used when the PTM is accessed. These instructions actually fetch a byte from memory, perform an operation, then restore it to the same address location. Since the PTM uses the R/$\overline{W}$ line as an additional register select input, the modified data will not be restored to the same register if these instructions are used.

CONTROL REGISTER

Each timer in the MC6840 has a corresponding write-only Control Register. Control Register #2 has a unique address space (RS0=1, RS=0, RS2=0) and therefore may be written into at any time. The remaining Control Registers (#1 and #3) share the Address Space selected by a logic zero on all Register Select inputs.

CR20 — The least-significant bit of Control Register #2 (CR20) is used as an additional addressing bit for Control Registers #1 and #3. Thus, with all Register selects and R/W inputs at logic zero, Control Register #1 will be written into if CR20 is a logic one. Under the same conditions, Control Register #3 can also be written into after a $\overline{RESET}$ low condition has occurred, since all control register bits (except CR10) are cleared. Therefore, one may write in the sequence CR3, CR2, CR1.

TABLE 1 — REGISTER SELECTION

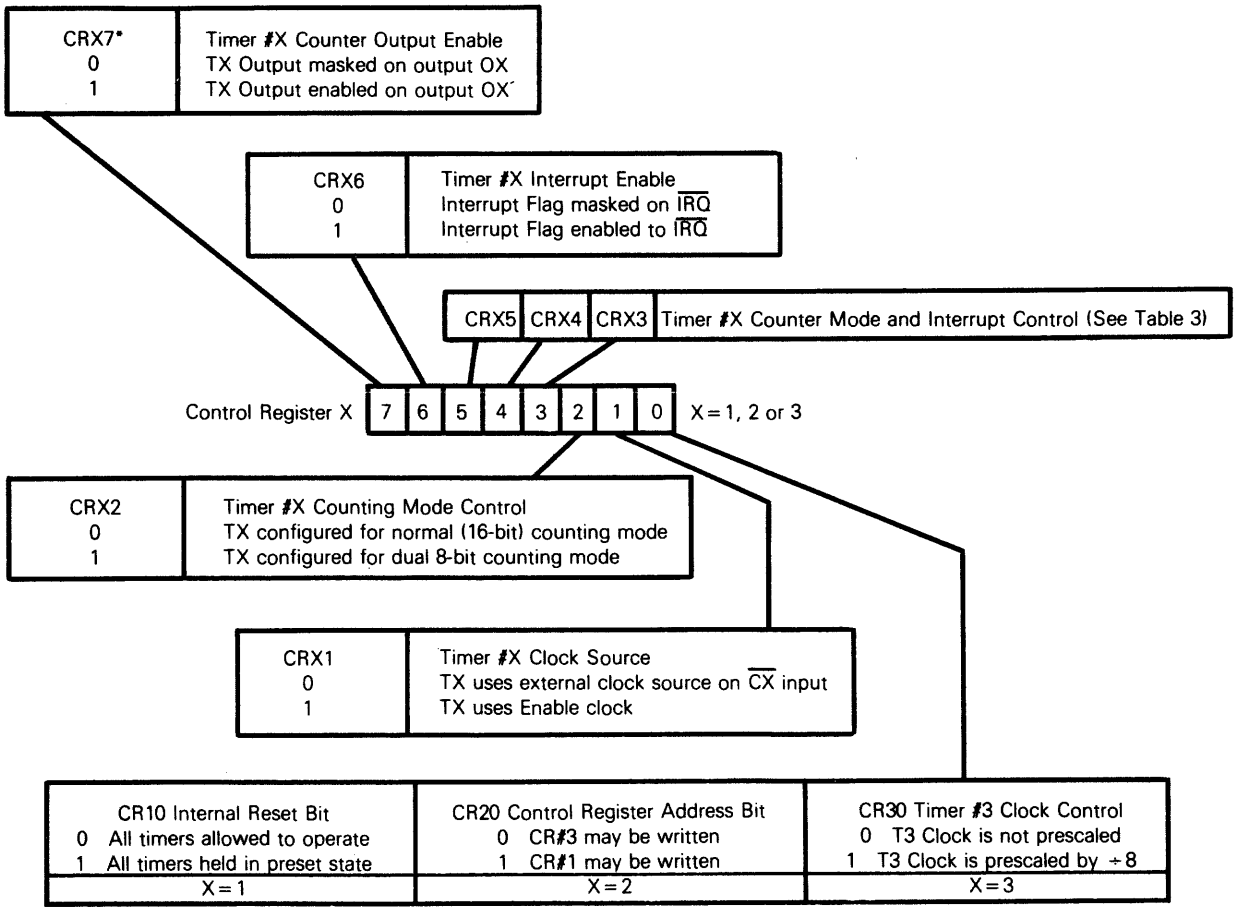
Register Select Inputs			Operations	
RS2	RS1	RS0	R/$\overline{W}$ = 0	R/$\overline{W}$ = 1
0	0	0	CR20 = 0 Write Control Register #3 CR20 = 1 Write Control Register #1	No Operation
0	0	1	Write Control Register #2	Read Status Register
0	1	0	Write MSB Buffer Register	Read Timer #1 Counter
0	1	1	Write Timer #1 Latches	Read LSB Buffer Register
1	0	0	Write MSB Buffer Register	Read Timer #2 Counter
1	0	1	Write Timer #2 Latches	Read LSB Buffer Register
1	1	0	Write MSB Buffer Register	Read Timer #3 Counter
1	1	1	Write Timer #3 Latches	Read LSB Buffer Register

CR10 — The least-significant bit of Control Register #1 is used as an Internal Reset bit. When this bit is a logic zero, all timers are allowed to operate in the modes prescribed by the remaining bits of the control registers. Writing a "one" into CR10 causes all counters to be preset with the contents of the corresponding counter latches, all counter clocks to be disabled, and the timer outputs and interrupt flags (Status Register) to be reset. Counter Latches and Control Registers are undisturbed by an Internal Reset and may be written into regardless of the state of CR10.

The least-signifcant bit of Control Register #3 is used as a selector for a ÷8 prescaler which is available with Timer #3 only. The prescaler, if selected, is effectively placed between the clock input circuitry and the input to Counter #3. It can therefore be used with either the internal clock (Enable) or an external clock source.

CR30 — The functions depicted in the foregoing discussions are tabulated in Table 2 for ease of reference.

TABLE 2 — CONTROL REGISTER BITS



Control Register Bits CR10, CR20, and CR30 are unique in that each selects a different function. The remaining bits (1 through 7) of each Control Register select common functions, with a particular Control Register affecting only its corresponding timer.

CRX1 — Bit 1 of Control Register #1 (CR11) selects whether an internal or external clock source is to be used with Timer #1. Similarly, CR21 selects the clock source for Timer #2, and CR31 performs this function for Timer #3. The function of each bit of Control Register "X" can therefore be defined as shown in the remaining section of Table 2.

CRX2 — Control Register Bit 2 selects whether the binary information contained in the Counter Latches (and subsequently loaded into the counter) is to be treated as a single 16-bit word or two 8-bit bytes. In the single 16-bit Counter Mode (CRX2=0) the counter will decrement to zero after N+1 enabled (G=0) clock periods, where N is defined as the 16-bit number in the Counter Latches. With CRX2=1, a similar Time Out will occur after (L+1)•(M+1) enabled clock periods, where L and M, respectively, refer to the LSB and MSB bytes in the Counter Latches.

CRX3-CRX7 — Control Register Bits 3, 4, and 5 are explained in detail in the Timer Operating Mode section. Bit 6 is an interrupt mask bit which will be explained more fully in conjunction with the Status Register, and bit 7 is used to enable the corresponding Timer Output. A summary of the control register programming modes is shown in Table 3.

STATUS REGISTER/INTERRUPT FLAGS

The MC6840 has an internal Read-Only Status Register which contains four Interrupt Flags. (The remaining four bits of the register are not used, and defaults to zeros when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits, while Bit 7 is a Composite Interrupt Flag. This flag bit will be asserted if any of the individual flag bits is set while Bit 6 of the corresponding Control Register is at a logic one. The conditions for asserting the composite Interrupt Flag bit can therefore be expressed as:

$$INT = I1 \cdot CR16 + I2 \cdot CR26 + I3 \cdot CR36$$

where INT = Composite Interrupt Flag (Bit 7)
I1 = Timer #1 Interrupt Flag (Bit 0)
I2 = Timer #2 Interrupt Flag (Bit 1)
I3 = Timer #3 Interrupt Flag (Bit 2)

An interrupt flag is cleared by a Timer Reset condition, i.e., External $\overline{RESET}=0$ or Internal Reset Bit (CR10)=1. It will also be cleared by a Read Timer Counter Command provided that the Status Register has previously been read while the interrupt flag was set. This condition on the Read Status Register-Read Timer Counter (RS-RT) sequence is designed to prevent missing interrupts which might occur after the status register is read, but prior to reading the Timer Counter.

An Individual Interrupt Flag is also cleared by a Write Timer Latches (W) command or a Counter Initialization (CI) sequence, provided that W or CI affects the Timer corresponding to the individual Interrupt Flag.

COUNTER LATCH INITIALIZATION

Each of the three independent timers consists of a 16-bit addressable counter and a 16-bit addressable latch. The counters are preset to the binary numbers stored in the latches. Counter initialization results in the transfer of the latch contents to the counter. See notes in Table 4 regarding the binary number N, L, or M placed into the Latches and their relationship to the output waveforms and counter Time-Outs.

Since the PTM data bus is 8-bits wide and the counters are 16-bits wide, a temporary register (MSB Buffer Register) is provided. This "write only" register is for the Most-Significant Byte of the desired latch data. Three addresses are provided for the MSB Buffer Register (as indicated in Table 1), but they all lead to the same Buffer. Data from the MSB Buffer will automatically be transferred into the Most-Significant Byte of Timer #X when a Write Timer #X Latches Command is performed. So it can be seen that the MC6840 has been designed to allow transfer of two bytes of data into the counter latches provided that the MSB is transferred first. The storage order must be observed to ensure proper latch operation.

In many applications, the source of the data will be an M6800 Family MPU. It should be noted that the 16-bit store operations of the M6800 family microprocessors (STS and STX) transfer data in the order required by the PTM. A Store Index Register Instruction, for example, results in the MSB of the X register being transferred to the selected address, then the LSB of the X register being written into the next higher location. Thus, either the index register or stack pointer may be transferred directly into a selected counter latch with a single instruction.

A logic zero at the $\overline{RESET}$ input also initializes the counter latches. In this case, all latches will assume a maximum count of $65{,}535_{10}$. It is important to note that an Internal

TABLE 3 — PTM OPERATING MODE SELECTION

CRX3	CRX4	CRX5	
0	0	0	Continuous Operating Mode: Gate ↓ or Write to Latches or Reset Causes Counter Initialization
0	0	1	Frequency Comparison Mode: Interrupt If Gate ↑⎍↓ is < Counter Time Out
0	1	0	Continuous Operating Mode: Gate ↓ or Reset Causes Counter Initialization
0	1	1	Pulse Width Comparison Mode: Interrupt if Gate ↑⎍↑ is < Counter Time Out
1	0	0	Single Shot Mode: Gate ↓ or Write to Latches or Reset Causes Counter Initialization
1	0	1	Frequency Comparison Mode: Interrupt If Gate ↑⎍↓ is > Counter Time Out
1	1	0	Single Shot Mode: Gate ↓ or Reset Causes Counter Initialization
1	1	1	Pulse Width Comparison Mode: Interrupt If Gate ↑⎍↑ is > Counter Time Out

Reset (Bit zero of Control Register 1 Set) has no effect on the counter latches.

COUNTER INITIALIZATION

Counter Initialization is defined as the transfer of data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition ($\overline{RESET}=0$ or CR10=1) is recognized. It can also occur — depending on Timer Mode — with a Write Timer Latches command or recognition of a negative transition of the Gate input.

Counter recycling or re-initialization occurs when a negative transition of the clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter.

ASYNCHRONOUS INPUT/OUTPUT LINES

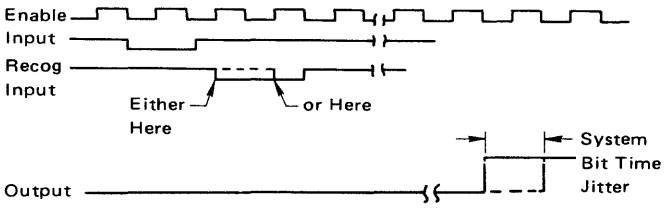
Each of the three timers within the PTM has external clock and gate inputs as well as a counter output line. The inputs are high-impedance, TTL-compatible lines and ouputs are capable of driving two standard TTL loads.

CLOCK INPUTS ($\overline{C1}$, $\overline{C2}$, and $\overline{C3}$) — Input pins $\overline{C1}$, $\overline{C2}$, and $\overline{C3}$ will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3, respectively. The high and low levels of the external clocks must each be stable for at least one system clock period plus the sum of the setup and hold times for the clock inputs. The asynchronous clock rate can vary from dc to the limit imposed by the Enable Clock Setup, and Hold times.

The external clock inputs are clocked in by Enable pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay between a clock input transition and internal recognition of that transition by the PTM. All references to C inputs in this document relate to internal recognition of the input transition. Note that a clock high or low level which does not meet setup and hold time specifications may require an additional Enable pulse for recognition. When observing recurring events, a lack of synchronization will result in "jitter" being observed on the output of the PTM when using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with Enable, permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time.

"Input jitter" can be as great as the time between input signal negative going transitions plus the system jitter, if the first transition is recognized during one system cycle, and not recognized the next cycle, or vice versa. See Figure 11.

FIGURE 11 — INPUT JITTER



CLOCK INPUT $\overline{C3}$ (÷8 PRESCALER MODE) — External clock input $\overline{C3}$ represents a special case when Timer #3 is programmed to utilize its optional ÷8 prescaler mode.

The divide-by-8 prescaler contains an asynchronous ripple counter; thus, input setup (t_{su}) and hold times (t_{hd}) do not apply. As long as minimum input pulse widths are maintained, the counter will recognize and process all input clock ($\overline{C3}$) transitions. However, in order to guarantee that a clock transition is processed during the current E cycle, a certain amount of synchronization time (t_{sync}) is required between the $\overline{C3}$ transition and the falling edge of Enable (see Figure 9). If the synchronization time requirement is not met, it is possible that the $\overline{C3}$ transition will not be processed until the following E cycle.

The maximum input frequency and allowable duty cycles for the ÷8 prescaler mode are specified under the AC Operating Characteristics. Internally, the ÷8 prescaler output is treated in the same manner as the previously discussed clock inputs.

GATE INPUTS ($\overline{G1}$, $\overline{G2}$, $\overline{G3}$) — Input pins $\overline{G1}$, $\overline{G2}$, and $\overline{G3}$ accept asynchronous TTL-compatible signals which are used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the PTM by the E (enable) clock in the same manner as the previously discussed clock inputs. That is, a Gate transition is recognized by the PTM on the fourth Enable pulse (provided setup and hold time requirements are met), and the high or low levels of the Gate input must be stable for at least one system clock period plus the sum of setup and hold times. All references to G transition in this document relate to internal recognition of the input transition.

The Gate inputs of all timers directly affect the internal 16-bit counter. The operation of $\overline{G3}$ is therefore independent of the ÷8 prescaler selection.

TIMER OUTPUTS (O1, O2, O3) — Timer outputs O1, O2, and O3 are capable of driving up to two TTL loads and produce a defined output waveform for either Continuous or Single-Shot Timer modes. Output waveform definition is accomplished by selecting either Single 16-bit or Dual 8-bit operating modes. The Single 16-bit mode will produce a square-wave output in the continuous mode and a single pulse in the single-shot mode. The Dual 8-bit mode will produce a variable duty cycle pulse in both the continuous and single-shot timer modes. One bit of each Control Register (CRX7) is used to enable the corresponding output. If this bit is cleared, the output will remain low (V_{OL}) regardless of the operating mode. If it is cleared while the output is high the output will go low during the first enable cycle following a write to the Control Register.

The Continuous and Single-Shot Timer Modes are the only ones for which output response is defined in this data sheet. Refer to the Programmable Timer Fundamentals and Applications manual for a discussion of the output signals in other modes. Signals appear at the outputs (unless CRX7=0) during Frequency and Pulse Width comparison modes, but the actual waveform is not predictable in typical applications.

TIMER OPERATING MODES

The MC6840 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of each control register (CRX3, CRX4, and CRX5) to define different operating modes of the Timers. These modes are divided into WAVE SYNTHESIS and WAVE MEASUREMENT modes, and are outlined in Table 4.

TABLE 4 — OPERATING MODES

Control Register			Timer Operating Mode	
CRX3	CRX4	CRX5		
0	*	0	Continuous	Synthesizer
0	*	1	Single-Shot	
1	0	*	Frequency Comparison	Measurement
1	1	*	Pulse Width Comparison	

*Defines Additional Timer Function Selection.

One of the WAVE SYNTHESIS modes is the Continuous Operating mode, which is useful for cyclic wave generation. Either symmetrical or variable duty-cycle waves can be generated in this mode. The other wave synthesis mode, the Single-Shot mode, is similar in use to the Continuous operating mode, however, a single pulse is generated, with a programmable preset width.

The WAVE MEASUREMENT modes include the Frequency Comparison and Pulse Width Comparison modes which are used to measure cyclic and singular pulse widths, respectively.

In addition to the four timer modes in Table 4, the remaining control register bit is used to modify counter initialization and enabling or interrupt conditions.

WAVE SYNTHESIS MODES

CONTINUOUS OPERATING MODE (TABLE 5) — The continuous mode will synthesize a continuous wave with a period proportional to the preset number in the particular timer latches. Any of the timers in the PTM may be programmed to operate in a continuous mode by writing zeroes into bits 3 and 5 of the corresponding control register. Assuming that the timer output is enabled (CRX7 = 1), either a square wave or a variable duty cycle waveform will be generated at the Timer Output, OX. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = 1 or External Reset = 0) condition or internal recognition of a negative transition of the Gate input results in Counter Initialization. A Write Timer latches command can be selected as a Counter Initialization signal by clearing CRX4.

The counter is enabled by an absence of a Timer Reset condition and a logic zero at the Gate input. In the 16-bit mode, the counter will decrement on the first clock cycle during or after the counter initialization cycle. It continues to decrement on each clock signal so long as G remains low and no reset condition exists. A Counter Time Out (the first clock after all counter bits = 0) results in the Individual Interrupt Flag being set and reinitialization of the counter.

In the Dual 8-bit mode (CRX2 = 1) [refer to the example in Figure 12 and Tables 5 and 6] the MSB decrements once for every full countdown of the LSB + 1. When the LSB = 0, the MSB is unchanged; on the next clock pulse the LSB is reset to the count in the LSB Latches, and the MSB is decremented by 1 (one). The output, if enabled, remains low during and after initialization and will remain low until the counter MSB is all zeroes. The output will go high at the beginning of the next clock pulse. The output remains high until both the LSB and MSB of the counter are all zeroes. At the beginning of the next clock pulse the defined Time Out (TO) will occur and the output will go low. In the Dual 8-bit mode the period of the output of the example in Figure 12 would span 20 clock pulses as opposed to 1546 clock pulses using the normal 16-bit mode.

A special time-out condition exists for the dual 8-bit mode (CRX2 = 1) if L = 0. In this case, the counter will revert to a mode similar to the single 16-bit mode, except Time Out occurs after M + 1* clock pulses. The output, if enabled, goes low during the Counter Initialization cycle and reverses state at each Time Out. The counter remains cyclical (is reinitialized at each Time Out) and the Individual Interrupt Flag is set when Time Out occurs. If M = L = 0, the internal counters do not change, but the output toggles at a rate of ½ the clock frequency.

TABLE 5 — CONTINUOUS OPERATING MODES

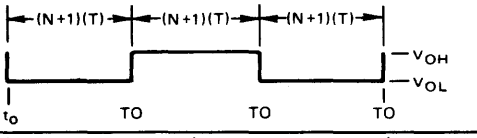
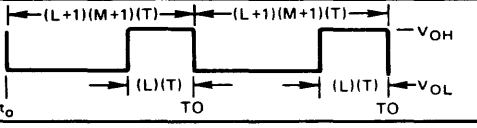
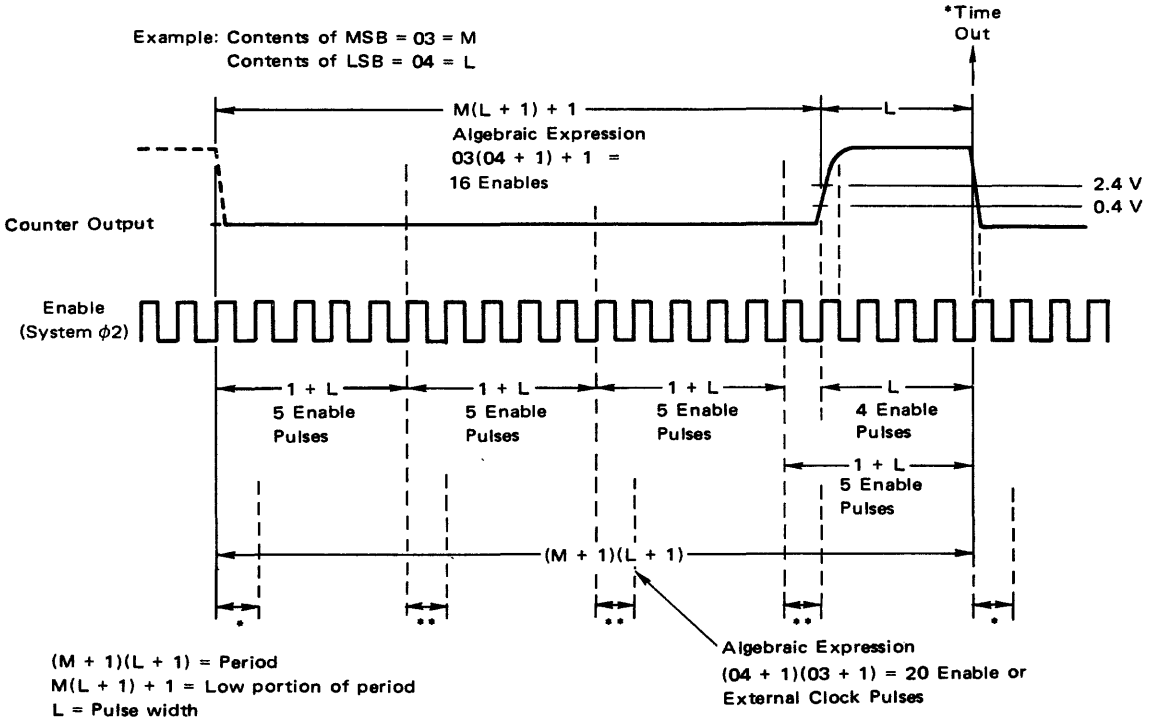
Synthesis Modes		CONTINUOUS MODE (CRX3 = 0, CRX5 = 0)	
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	*Timer Output (OX) (CRX7 = 1)
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

FIGURE 12 — TIMER OUTPUT WAVEFORM EXAMPLE
(Continuous Dual 8-Bit Mode Using Internal Enable)



*Preset LSB and MSB to Respective Latches on the negative transition of the Enable
**Preset LSB to LSB Latches and Decrement MSB by one on the negative transition of the Enable

The discussion of the Continuous Mode has assumed that the application requires an output signal. It should be noted that the Timer operates in the same manner with the output disabled (CRX7=0). A Read Timer Counter command is valid regardless of the state of CRX7.

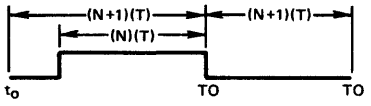
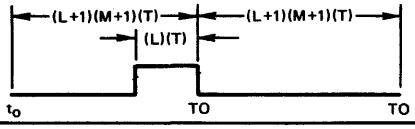
SINGLE-SHOT TIMER MODE — This mode is identical to the Continuous Mode with three exceptions. The first of these is obvious from the name — the output returns to a low level after the initial Time Out and remains low until another Counter Initialization cycle occurs.

As indicated in Table 6, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the Gate input level reamining in the low state for the Single-Shot mode.

Another special condition is introduced in the Single-Shot mode. If L=M=0 (Dual 8-bit) or N=0 (Single 16-bit), the output goes low on the first clock received during or after Counter Initialization. The output remains low until the Operating Mode is changed or nonzero data is written into the Counter Latches. Time Outs continue to occur at the end of each clock period.

TABLE 6 – SINGLE-SHOT OPERATING MODES

Synthesis Modes	SINGLE-SHOT MODE (CRX3 = 0, CRX7 = 1, CRX5 = 1)		
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	Timer Output (OX)
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

Symbols are as defined in Table 5.

The three differences between Single-Shot and Continous Timer Mode can be summarized as attributes of the Single-Shot mode:

1. Output is enabled for only one pulse until it is reinitialized.
2. Counter Enable is independent of Gate.
3. L = M = 0 or N = 0 disables output.

Aside from these differences, the two modes are identical.

WAVE MEASUREMENT MODES

TIME INTERVAL MODES — The Time Interval Modes are the Frequency (period) Measurement and Pulse Width Comparison Modes, and are provided for those applications which require more flexibility of interrupt generation and Counter Initialization. Individual Interrupt Flags are set in these modes as a function of both Counter Time Out and transitions of the $\overline{\text{Gate}}$ input. Counter Initialization is also affected by Interrupt Flag status.

A timer's output is normally not used in a Wave Measurement mode, but it is defined. If the output is enabled, it will operate as follows. During the period between reinitialization of the timer and the first Time Out, the output will be a logical zero. If the first Time Out is completed (regardless of its method of generation), the output will go high. If further TO's occur, the output will change state at each completion of a Time-Out.

The counter does operate in either Single 16-bit or Dual 8-bit modes as programmed by CRX2. Other features of the Wave Measurement Modes are outlined in Table 7.

Frequency Comparison Or Period Measurement Mode (CRX3 = 1, CRX4 = 0) — The Frequency Comparison Mode with CRX5 = 1 is straightforward. If Time Out occurs prior to the first negative transition of the $\overline{\text{Gate}}$ input after a Counter Initialization cycle, and Individual Interrupt Flag is set. The counter is disabled, and a Counter Initialization cycle cannot begin until the interrupt flag is cleared and a negative transition on $\overline{\text{G}}$ is detected.

If CRX5 = 0, as shown in Tables 7 and 8, an interrupt is generated if $\overline{\text{Gate}}$ input returns low prior to a Time Out. If a Counter Time Out occurs first, the counter is recycled and continues to decrement. A bit is set within the timer on the initial Time Out which precludes further individual interrupt generation until a new Counter Initialization cycle has been completed. When this internal bit is set, a negative transition of the $\overline{\text{Gate}}$ input starts a new Counter Initialization cycle. (The condition of $\overline{\text{G}}\downarrow\bullet\overline{\text{I}}\bullet\text{TO}$ is satisfied, since a Time Out has occurred and no individual Interrupt has been generated.)

Any of the timers within the PTM may be programmed to compare the period of a pulse (giving the frequency after calculations) at the $\overline{\text{Gate}}$ input with the time period requested for Counter Time Out. A negative transition of the $\overline{\text{Gate}}$ Input enables the counter and starts a Counter Initialization cycle — provided that other conditions, as noted in Table 8, are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 8 that an interrupt condition will be generated if CRX5 = 0 and the period of the pulse (single pulse or measured separately repetitive pulses) at the Gate input is less than the Counter Time Out period. If CRX5 = 1, an interrupt is generated if the reverse is true.

Assume now with CRX5 = 1 that a Counter Initialization has occurred and that the Gate input has returned low prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each Gate input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

Pulse Width Comparison Mode (CRX3 = 1, CRX4 = 1) — This mode is similar to the Frequency Comparison Mode except for a positive, rather than negative, transition of the Gate input termintes the count. With CRX5 = 0, an Individual Interrupt Flag will be generated if the zero level pulse applied to the Gate input is less than the time period required for Counter Time Out. With CRX5 = 1, the interrupt is generated when the reverse condition is true.

As can be seen in Table 8, a positive transition of the Gate input disables the counter. With CRX5 = 0, it is therefore possible to directly obtain the width of any pulse causing an interrupt. Similar data for other Time Interval Modes and conditions can be obtained, if two sections of the PTM are dedicated to the purpose.

FIGURE 7 — OUTPUT DELAY

CRX3 = 1			
CRX4	**CRX5**	**Application**	**Condition for Setting Individual Interrupt Flag**
0	0	Frequency Comparison	Interrupt Generated if $\overline{\text{Gate}}$ Input Period (1/F) is less than Counter Time Out (TO)
0	1	Frequency Comparison	Interrupt Generated if $\overline{\text{Gate}}$ Input Period (1/F) is greater than Counter Time Out (TO)
1	0	Pulse Width Comparison	Interrupt Generated if $\overline{\text{Gate}}$ Input "Down Time" is less than Counter Time Out (TO)
1	1	Pulse Width Comparison	Interrupt Generated if $\overline{\text{Gate}}$ Input "Down Time" is greater than Counter Time Out (TO)

SEE PAGE 2 FOR TABLE 8

MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6850
(1.0 MHz)
MC68A50
(1.5 MHz)
MC68B50
(2.0 MHz)

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

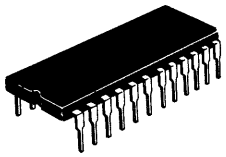
The MC6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the MC6800 Microprocessing Unit.

The bus interface of the MC6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bidirectional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking. The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation, three control lines are provided. These lines allow the ACIA to interface directly with the MC6860L 0-600 bps digital modem.

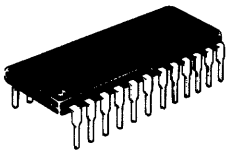
- 8- and 9-Bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Programmable Control Register
- Optional ÷1, ÷16, and ÷64 Clock Modes
- Up to 1.0 Mbps Transmission
- False Start Bit Deletion
- Peripheral/Modem Control Functions
- Double Buffered
- One- or Two-Stop Bit Operation

MOS
(N-CHANNEL, SILICON-GATE)

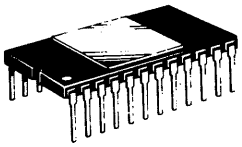
ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER



S SUFFIX
CERDIP PACKAGE
CASE 623

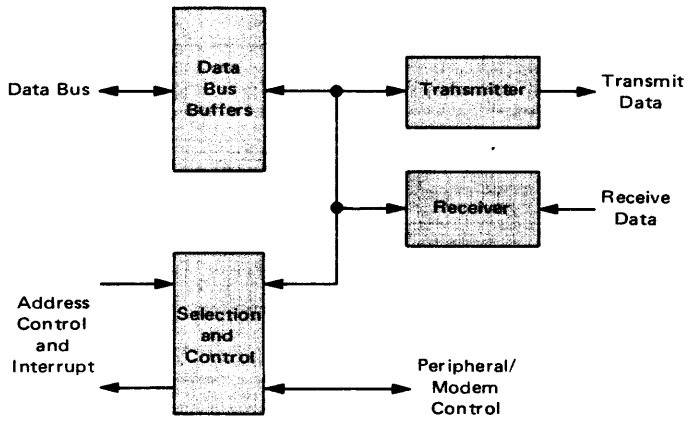


P SUFFIX
PLASTIC PACKAGE
CASE 709



L SUFFIX
CERAMIC PACKAGE
CASE 716

MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER BLOCK DIAGRAM



PIN ASSIGNMENT

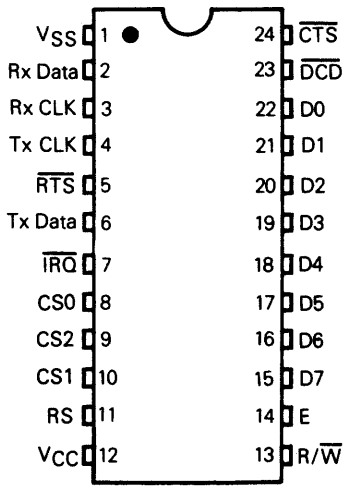
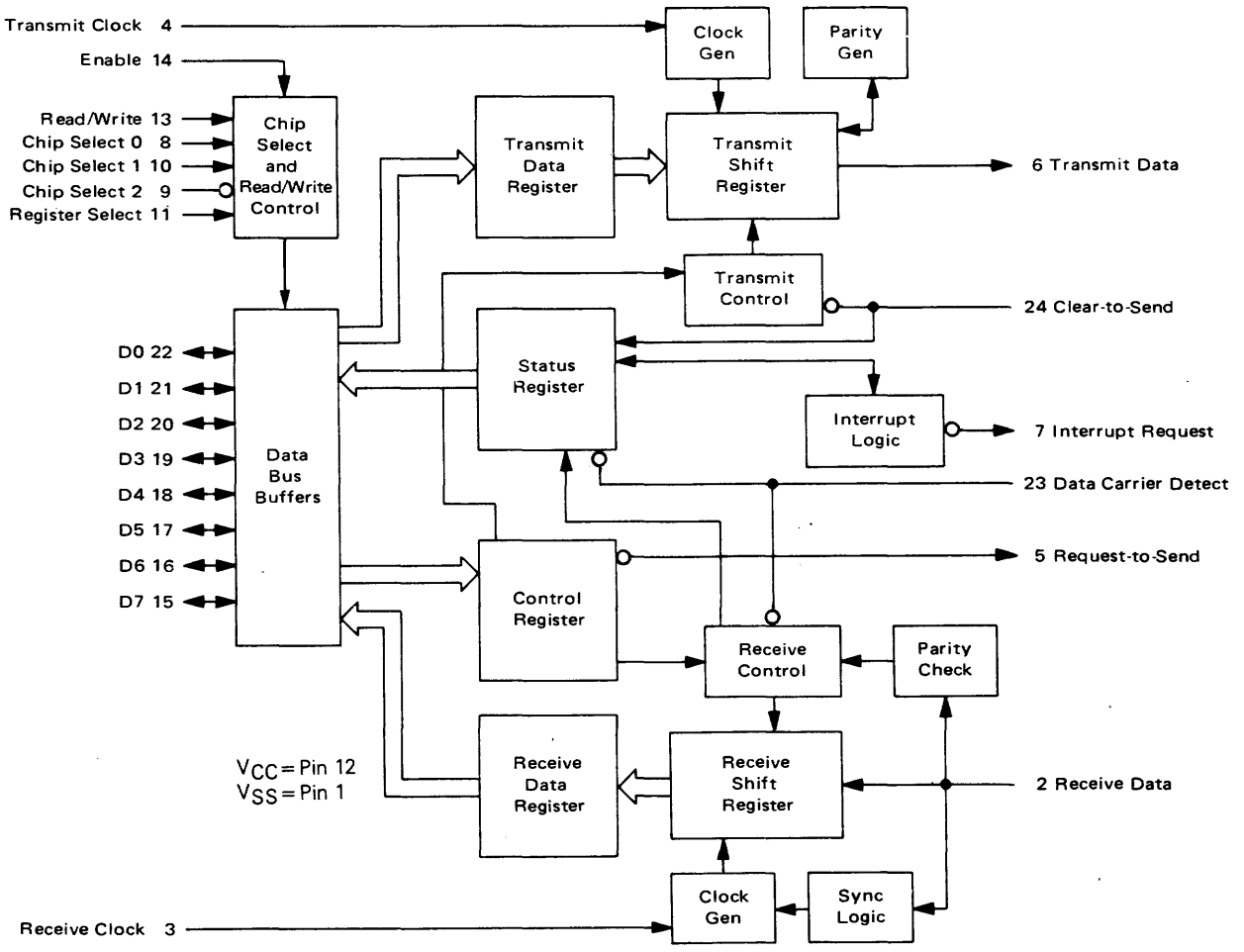


FIGURE 9 – EXPANDED BLOCK DIAGRAM



DEVICE OPERATION

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.

POWER ON/MASTER RESET

The master reset (CR0, CR1) should be set during system initialization to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. During the first master reset, the $\overline{IRQ}$ and $\overline{RTS}$ outputs are held at level 1. On all other master resets, the $\overline{RTS}$ output can be programmed high or low with the $\overline{IRQ}$ output held high. Control bits CR5 and CR6 should also be programmed to define the state of $\overline{RTS}$ whenever master reset is utilized. The ACIA also contains internal power-on reset logic to detect the power line turn-on transition and hold the chip in a reset state to prevent erroneous output transitions prior to initialization. This circuitry depends on clean power turn-on transitions. The power-on reset is released by means of the bus-programmed master reset which must be applied prior to operating the ACIA. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, parity (even, odd, or none), etc.

TRANSMIT

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of

double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

RECEIVE

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of 8 or 32 low samples on the receive line in the divide-by-16 and 64 modes respectively. False start bit deletion capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for a 7-bit word (7 bits plus parity), the receiver strips the parity bit (D7 = 0) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. The above sequence continues until all characters have been received.

INPUT/OUTPUT FUNCTIONS

ACIA INTERFACE SIGNALS FOR MPU

The ACIA interfaces to the M6800 MPU with an 8-bit bidirectional data bus, three chip select lines, a register select line, an interrupt request line, read/write line, and enable line. These signals permit the MPU to have complete control over the ACIA.

ACIA Bidirectional Data (D0-D7) — The bidirectional data lines (D0-D7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs an ACIA read operation.

ACIA Enable (E) — The Enable signal, E, is a high-impedance TTL-compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the MC6800 $\phi 2$ Clock or MC6809 E clock.

Read/Write (R/$\overline{W}$) — The Read/Write line is a high-impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When Read/Write is high (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is low, the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the Read/Write signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS0, CS1, $\overline{CS2}$) — These three high-impedance TTL-compatible input lines are used to address the ACIA. The ACIA is selected when CS0 and CS1 are high and $\overline{CS2}$ is low. Transfers of data to and from the ACIA are then performed under the control of the Enable Signal, Read/Write, and Register Select.

Register Select (RS) — The Register Select line is a high-impedance input that is TTL compatible. A high level is used to select the Transmit/Receive Data Registers and a low level the Control/Status Registers. The Read/Write signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$) — Interrupt Request is a TTL-compatible, open-drain (no internal pullup), active low output that is used to interrupt the MPU. The $\overline{IRQ}$ output remains low as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set. The $\overline{IRQ}$ status bit, when high, indicates the $\overline{IRQ}$ output is in the active state.

Interrupts result from conditions in both the transmitter and receiver sections of the ACIA. The transmitter section causes an interrupt when the Transmitter Interrupt Enabled condition is selected (CR5•$\overline{CR6}$), and the Transmit Data Register Empty (TDRE) status bit is high. The TDRE status bit indicates the current status of the Transmitter Data Register except when inhibited by Clear-to-Send ($\overline{CTS}$) being high or the ACIA being maintained in the Reset condition. The interrupt is cleared by writing data into the Transmit Data Register. The interrupt is masked by disabling the Transmitter Interrupt via CR5 or CR6 or by the loss of $\overline{CTS}$ which inhibits the TDRE status bit. The Receiver section causes an interrupt when the Receiver Interrupt Enable is set and the Receive Data Register Full (RDRF) status bit is high, an Overrun has occurred, or Data Carrier Detect ($\overline{DCD}$) has gone high. An interrupt resulting from the RDRF status bit can be cleared by reading data or resetting the ACIA. Interrupts caused by Overrun or loss of $\overline{DCD}$ are cleared by reading the status register after the error condition has occurred and then reading the Receive Data Register or resetting the ACIA. The receiver interrupt is masked by resetting the Receiver Interrupt Enable.

CLOCK INPUTS

Separate high-impedance TTL-compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16, or 64 times the data rate may be selected.

Transmit Clock (Tx CLK) — The Transmit Clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK) — The Receive Clock input is used for synchronization of received data. (In the ÷1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

SERIAL INPUT/OUTPUT LINES

Receive Data (Rx Data) — The Receive Data line is a high-impedance TTL-compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used.

Transmit Data (Tx Data) — The Transmit Data output line transfers serial data to a modem or other peripheral.

PERIPHERAL/MODEM CONTROL

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

Clear-to-Send ($\overline{CTS}$) — This high-impedance TTL-compatible input provides automatic control of the transmitting end of a communications link via the modem Clear-to-Send active low output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

Request-to-Send ($\overline{RTS}$) — The Request-to-Send output enables the MPU to control a peripheral or modem via the data bus. The RTS output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the $\overline{RTS}$ output is low (the active state). This output can also be used for Data Terminal Ready (DTR).

Data Carrier Detect ($\overline{DCD}$) — This high-impedance TTL-compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem Data Carrier Detect output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low-to-high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set. The Rx CLK must be running for proper $\overline{DCD}$ operation.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers on the chip that are used for the status, control, receiving, and transmitting of data. The content of each of the registers is summarized in Table 1.

TRANSMIT DATA REGISTER (TDR)

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed with RS high and R/$\overline{W}$ low. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go low. Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 1-bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

RECEIVE DATA REGISTER (RDR)

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) in the status buffer to go high (full). Data may then be read through the bus by addressing the ACIA and selecting the Receive Data Register with RS and R/$\overline{W}$ high when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

TABLE 1 — DEFINITION OF ACIA REGISTER CONTENTS

Data Bus Line Number	Buffer Address			
	RS • $\overline{R/W}$ Transmit Data Register (Write Only)	RS • R/$\overline{W}$ Receive Data Register (Read Only)	$\overline{RS}$ • $\overline{R/W}$ Control Register (Write Only)	$\overline{RS}$ • R/$\overline{W}$ Status Register (Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select 1 (CR0)	Receive Data Register Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select 2 (CR1)	Transmit Data Register Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear-to-Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Transmit Control 1 (CR5)	Receiver Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Transmit Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Receive Interrupt Enable (CR7)	Interrupt Request ($\overline{IRQ}$)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7-bit plus parity modes.
*** Data bit is "don't care" in 7-bit plus parity modes.

CONTROL REGISTER

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are low. This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1) — The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set high to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4) — The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6) — Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

CR6	CR5	Function
0	0	$\overline{RTS}$ = low, Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = low, Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = high, Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = low, Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7) — The following interrupts will be enabled by a high level in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a low-to-high transition on the Data Carrier Detect ($\overline{DCD}$) signal line.

STATUS REGISTER

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is low and R/$\overline{W}$ is high. Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0 — Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect being high also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1 — The Transmit Data Register Empty bit being set high indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The low state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2 — The Data Carrier Detect bit will be high when the $\overline{DCD}$ input from a modem has gone high to indicate that a carrier is not present. This bit going high causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains high after the $\overline{DCD}$ input is returned low until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains high after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains high and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3 — The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send status bit.

Framing Error (FE), Bit 4 — Framing error indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the first stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The framing error flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5 — Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has

been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6 – The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver partiy check results are inhibited.

Interrupt Request ($\overline{IRQ}$), Bit 7 – The $\overline{IRQ}$ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is low the $\overline{IRQ}$ bit will be high to indicate the interrupt or service request status. $\overline{IRQ}$ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register.

NOTES

SUGGESTION / PROBLEM REPORT

Omnibyte welcomes your comments on it's products and this publication. Please use this form.

TO: OMNIBYTE CORPORATION
245 West Roosevelt Road (1-5)
West Chicago, Illinois 60185

Attention: 68K Systems Support

COMMENTS
Product:_____ Manual:_____

PLEASE PRINT:

NAME	TITLE
COMPANY	DIVISION
STREET	MAIL DROP PHONE
CITY	STATE ZIP CODE

Support: (312) 231-6880