INTELLIGENT THREE PORT
RS-232 SERIAL I/O INTERFACE

User's Manual

COPYRIGHT ©1982
GIMIX Inc.
1337 W. 37th Place
Chicago, IL 60609
312-927-5510 * TWX 910-221-4055

All Rights Reserved

Reproduction of this manual, in whole or part, by any means,without express written permission from GIMIX, Inc. is strictly prohibited.

GIMIX® amd GMX™ are trademarks of GIMIX, Inc. Chicago, IL 60609

GIMIX INTELLIGENT THREE PORT RS-232C SERIAL I/O INTERFACE BOARD

The GIMIX INTELLIGENT SERIAL INTERFACE is designed to increase throughput in multi-user systems by relieving the host CPU of much of the overhead involved in handling data transfers between the system and the user's terminals. It can also be used in applications such as code conversion (5/7 level, ASCII/EBCDIC, etc.), printer buffering, etc.; where I/O preprocessing can be advantageous.

FEATURES

- On board 2 MHz 68B09 CPU
- Up to 20K of on board memory (EPROM and RAM) †
- Buffered data transfers between the host and on board CPUs, using a Z8038 FIO I/O Interface Unit with:

  128-byte bidirectional, asynchronous FIFO buffer
  Mailbox registers for CPU to CPU communication
  Pattern-match capabilities
  Versatile interrupt options for both CPUs

- Three serial I/O ports using 6551A ACIAs, each with:

  Software selectable baud rates (50 to 19,200)
  Programmable word length, stop bits, and parity
  High performance RS-232C receivers and drivers
  Data set/modem control functions
  Buffered, full duplex operation

- Jumper programmable I/O connector pinouts for each port
- Two 8 bit sense switches for selecting software options
- Software controlled LED status indicators
- On board CPU can be reset by the host

† The board has a total of four sockets for memory devices. One holds either a 2K, 4K, or 8Kx8 EPROM; one a 2K, 4K, or 8Kx8 EPROM or a 2Kx8 static RAM; and two that can hold 2Kx8 static RAMs. This allows a maximum of 16K of EPROM with 4K of RAM or, for applications that require more than 4K of RAM, 6K of RAM with up to 8K of EPROM (14K total). The standard version of the board, without software, includes 4K of RAM.

©1982 GIMIX INC

INTRODUCTION

In a multi-user system, the CPU often spends a considerable amount of its time servicing interrupts from the users terminals. An intelligent I/O interface can relieve much of this burden by buffering the data transfers between the system and the users and by performing a certain amount of preprocessing of the data. This preprocessing can greatly reduce the number of interrupts to the main CPU, increasing the overall throughput of the system.

The GIMIX Intelligent Three Port RS-232C Serial I/O Interface Board is a completely independent processor, with its own memory and I/O devices. It is linked via on board serial interfaces to from one to three terminals or other peripherals, and to the host processor by a bidirectional FIFO (First In First Out) buffer that appears as several memory locations in the hosts address space. The board occupies one slot on the 30 pin SS-50 I/O bus and one or more boards can be installed in the system to service the required number of users or peripherals.

Unlike most I/O boards, the Intelligent I/O Interface requires on board software, specific to the application, in addition to the appropriate software drivers in the operating system. The board is available without application software for users who wish to write their own, or with software for certain specific applications. Contact GIMIX for information on the available software.

Sections I through V of this manual cover the basic hardware configuration of the board and general information on its use. Section VI, included for users who will be writing their own software, is intended to be used in conjunction with the manufacturer's data on the Z8038. For more information on specific applications, see the documentation for the software being used and/or the manufacturers literature on the specific devices involved (68B09, Z8038, 6551A, etc.).

©1982 GIMIX Inc.
1337 W 37th Place
Chicago, IL 60609
(312) 927-5510 • TWX 910-221-4055

This entire manual is copyright 1982 by GIMIX Inc. Reproduction of this manual by any means, in whole or part, without express written permission from GIMIX Inc. is prohibited.

TABLE OF CONTENTS

(cont.)

† IMPORTANT INFORMATION - PLEASE READ

©1982 GIMIX INC.

VI: PROGRAMMERS NOTES (cont.)

† IMPORTANT INFORMATION - PLEASE READ

©1982 GIMIX INC.

SECTION I: HARDWARE CONFIGURATION OPTIONS

The board has several hardware options which may need to be properly configured before the board is used.

If the board was purchased with software installed, all of the options for interrupts and PROM/RAM socket configuration will be properly configured and should not normally need to be changed. The only options that may require reconfiguration are the I/O connector pinouts (JA-5,6, and 7) and the sense switches (SW1 and 2). The configuration of these options will depend on the software and the terminals or other peripherals being used. See the software documentation and the appropriate sections of this manual for more information on option configuration.

If the board was purchased without software, it is up to the user to determine the appropriate options to use. In particular the PROM/RAM socket jumpers must be configured to suit the memory devices being used.

NOTE: The standard version of the board (w/o software) includes two 2Kx8 static RAMs installed at U-9 and U-11. The user must provide the necessary EPROM(s) and software for the board.

********** IMPORTANT **********

Due to the slow timing requirements of the Z8038 FIO I/O Interface Unit, the motherboard MUST HAVE A SLOW I/O CIRCUIT and IT MUST BE ENABLED when using the GIMIX Intelligent I/O Board at bus speeds above 1 MHz. Failure to have the slow I/O circuitry enabled when using the Intelligent I/O board will cause unpredictable results.

On current production GIMIX motherboards, those that have a separate 10 pin baud rate generator board, the slow I/O option is enabled by setting jumper area JA-3 as shown in figure "G" of the JUMPER OPTIONS & SWITCHES drawing included in the motherboard documentation. On earlier versions of the GIMIX motherboard that have slow I/O capabilities, the option is enabled by a DIP-switch. See the motherboard documentation for more information.

I-I: ACIA INTERRUPT OPTION JUMPER (JA-1)

JA-1 determines the type of interrupt (IRQ or FIRQ) that can be generated by the three 6551A ACIAs. The interrupt output of the 6551As is connected ONLY to the selected input of the on-board 6809, it does not affect the host (system) CPU.

The board is factory configured to generate IRQ interrupts. To select FIRQ interrupts from the ACIAs, the user must cut the soldered jumper or PC board trace at JA-1 and install a new jumper as indicated in the JUMPER OPTIONS drawing.

The type of interrupt required will depend on the software used. See the software documentation for information on the proper interrupt option to use.

I-II: Z8038 FIO PORT 2 INTERRUPT OPTION JUMPER (JA-2)

JA-2 determines the type of interrupt that can be generated by port 2 of the Z8038 FIO. The port 2 interrupt output of the FIO is connected ONLY to the on-board 6809, it does not affect the host (system) CPU.

The board is factory configured to generate IRQ interrupts. To select FIRQ interrupts from the FIO, cut the soldered jumper or PC board trace at JA-2 and install a new jumper as indicated in the JUMPER OPTIONS drawing.

The type of interrupt required will depend on the software used. See the software documentation for information on the proper interrupt option to use.

NOTE: Port 1 of the Z8038 FIO is permanently connected to the host (system) CPUs IRQ interrupt line through the 30 pin bus.

I-III: PROM SOCKET U10 OPTION JUMPER (JA-3)

JA-3 allows PROM socket U10 to be configured for either 2, 4 or 8K EPROMs. PROM socket U10 occupies the upper 8K of the on board CPUs address space ($E000-$FFFF). When using a 2K EPROM JA-3 must be placed in the UPPER position. For 4 or 8K devices JA-3 must be placed in the LOWER position (see the JUMPER OPTIONS drawing).

NOTE: PROM socket U10 is a 28 pin socket that can hold either 24 pin (2 or 4K) or 28 pin (8K) devices. See section II for information on the type of PROMs that can be used and their installation.

I-IV: PROM/RAM SOCKET U12 OPTION JUMPER (JA-4)

JA-4 allows PROM/RAM socket U12 to be configured for 2, 4, or 8K EPROMs or for a 2K static RAM. PROM socket U12 occupies 8K of the on board CPUs address space ($C000-$DFFF). When using a 2K EPROM JA-4

must be placed in the LOWER position. For 4 or 8K EPROMs JA-4 must be placed in the CENTER position. When a 2K static RAM is to be installed in U-12, JA-4 must be placed in the UPPER position (see the JUMPER OPTIONS drawing).

NOTE: PROM socket U12 is a 28 pin socket that can hold either 24 pin (2 or 4K) or 28 pin (8K) devices. See section II for information on the type of PROMs that can be used and their installation.

I-V: RS-232 I/O CONNECTOR CONFIGURATION JUMPERS (JA-5,6,and 7)

JA-5,6, and 7 are used to configure their respective RS-232 output connectors (J4,3, and 2) for the required pinouts. The signal designations shown on the JUMPER OPTION drawing (RX, TX, RTS, etc.) correspond (except for PU which is described below) to the signal definitions of the 6551 ACIAs. The direction of the signals is shown in the table below. The pin numbers shown on the JUMPER OPTIONS drawing correspond to the pinout of the 26 pin headers located at the top of the board (when the pins are counted as shown on the component layout) and to the pinout of the "D" connectors when GIMIX standard cable sets are used.

TABLE I-I: RS-232 SIGNAL DIRECTION

I/O BOARD		PERIPHERAL
RX	<----------	
TX	---------->	
DCD	<----------	*
RTS	---------->	
CTS	<----------	*
DTR	---------->	
DSR	<----------	*
PU	---------->	†

NOTE: Pin 7 of all connectors is connected to signal ground

* The input lines all have on board pull-up resistors and may be left open (unconnected) if they are not needed.

† PU is an RS-232 High level signal that may be used to pull unused inputs on the peripheral device to an active state.

For most applications, when high speed terminals that do not require handshaking are used, only the RX and TX lines and signal ground (pin 7) need be connected. If only the RX and TX lines are needed, the remaining jumpers should be removed or a three wire cable used to connect the board to the peripheral. This will prevent possible interference from unused signals that may be present on the peripheral's I/O connector.

If handshaking is required by the peripheral device (modems, printers, etc.), the choice of signals will depend largely on the particular software used. The I/O connector configuration jumpers

allow the most common pinout arrangements by simply installing the appropriate jumper blocks. Other arrangements can be achieved by wire-wrapping the appropriate connections at the jumper area.

The JUMPER OPTIONS drawing shows the pinout of the jumper areas and some sample jumper configurations.

NOTE: Due to a peculiarity in the design of the 6551 ACIA the CTS input can not be used, as it normally is with the 6850, to control the flow of data from the I/O board to a printer or other peripheral that requires start/stop control of the data. When the CTS input to the 6551 is made inactive to temporarily stop the data flow, the transmission of data from the 6551 is immediately halted and the TX line goes to the "MARK" condition. This causes any data byte being transmitted when CTS changes to be lost. To provide start/stop control one of the other control lines (DCD or DSR) must be used. This peculiarity should not affect the use of the CTS line in normal modem control applications. See the software documentation or contact the software vendor for information on the appropriate control lines to use in a particular application.

SECTION II: ON BOARD MEMORY

********** CAUTION **********

Both PROM socket U10 and PROM/RAM socket U12 are 28 pin sockets that can be used with either 24 or 28 pin memory devices. All U10 and U12 pin number references in this documentation refer to the pin numbers for the 28 pin devices. When 24 pin devices are installed in these sockets, they MUST be offset in the socket so that pins 1, 2, 27, and 28 of the 28 pin socket are unused. The table below shows the conversion between the 24 and 28 pin pin numbers.

TABLE II-I: PIN NUMBER CONVERSION FOR 24 AND 28 PIN DEVICES.

	1	28	
	2	27	
1	3	26	24
2	4	25	23
3	5	24	22
4	6	23	21
5	7	22	20
6	8	21	19
7	9	20	18
8	10	19	17
9	11	18	16
10	12	17	15
11	13	16	14
12	14	15	13

The board features four sockets which will accept a variety of memory devices. The choice of these devices is determined by the requirements of the software. The four memory sockets are mapped into the on board 6809s address space as shown in the memory map. Each

socket occupies 8K of address space, regardless of the size (2, 4, or 8K) of the memory device used. When 2 or 4K devices are used, they repeat within the 8K space allocated to the socket.

When installing memory devices, be sure to observe the proper orientation of pin 1 and, in the case of 24 pin devices in U10 or U12, the proper offset in the socket. The configuration jumpers for U10 (JA-3) and U12 (JA-4) must also be properly set.

II-I: PROM SOCKET U10 ($E000-$FFFF)

This socket occupies the top 8K of the on board CPUs address space and must contain an EPROM with the 6809 reset and interrupt vectors in the top 16 bytes. The socket will accept standard pinout, single supply voltage 2, 4, or 8K EPROMs with a 350 ns. or faster access time. Compatible devices include: 2516, 2716 (5V only), 2732, 2732A, and 2764.

The board is NOT compatible with 2532 and 2564 type devices.

Jumper area JA-3 must be configured for the device size being used.

II-II: PROM/RAM SOCKET U12 ($C000-$DFFF)

This socket occupies 8K of the on board CPUs address space and will optionally contain PROM or RAM depending on the requirements of the software. It will accept the same EPROMs as socket U10 (see above). In addition this socket can be configured for 2Kx8 static RAMs. Compatible RAMs include: 6116 CMOS, 2116 NMOS, and M58725 NMOS SRAMs.

Jumper area JA-4 must be configured for the device size and type being used.

II-III: RAM SOCKETS U11 ($1800-$1FFF) and U9 ($2000-$27FF)

Sockets U11 and U9 are for RAM only. These sockets may each contain a 2Kx8 SRAM. Each socket actually occupies 8K of address space, with the 2K device repeated 4 times within the 8K space (see the memory map). Using the addresses shown above ($1800-$1FFF and $2000-$27FF) gives the user 4K of continuous RAM when both devices are installed. These sockets are compatible with the same RAMs listed above for U12 (6116, 2016, and M58725).

III-I: SENSE SWITCHES (SW1 and 2)

Switches SW1 and 2 are 8 position DIP-switches that can be used to select options for the on board software. Each switch occupies 8K of the on board CPUs address space (SW1 = $4000-$5FFF, SW2 = $6000-$7FFF). The settings on either switch can be determined by reading from any address within the 8K allocated to that switch.

Section 1 of each switch corresponds to the least significant bit (D0) of the data read from the switch, section 2 to the next most significant bit (D1), ... , section 8 corresponds to the most significant bit (D7). When a switch section is set ON (CLOSED) the corresponding bit in the data word will read "1". When a switch section is set OFF (OPEN) the corresponding bit in the data word will read "0".

The actual settings for switches SW1 and SW2 will be determined by the software and the options selected. See the software documentation for the required settings. Typical uses for the sense switches include selection of initial baud rates, and selection of options such as self-test modes.

III-II: STATUS LEDs (DS1 and DS2)

The status LEDs can be used as a visual indication of proper operation of the board or to indicate the results of internal software diagnostics. The LEDs can be illuminated under program control by repeatedly accessing (read or write) the appropriate memory locations in the on board address space.

The status LEDs "occupy" the same address space as the sense switches. DS1 can be illuminated by accessing any address associated with SW1 ($4000-$5FFF) and DS2 by accessing the addresses associated with SW2 ($6000-$7FFF).

The status LEDs are not latched; they can not simply be turned on or off by reading or writing to an address. Each time an address associated with one of the LEDs is accessed the LED receives a brief pulse of current. The duration of this pulse is too short to be visible so the address must be accessed repeatedly for the entire time the LED is to be illuminated. The time between repeated accesses determines the brightness of the LED; the faster the accesses the brighter the LED will be.

See the software documentation for information on interpretation of the status LED display.

©1982 GIMIX INC

The board has three 6551 Asynchronous Communications Interface Adapters (ACIAs) which provide serial communication between the board and up to 3 peripherals (terminals, printers, etc). The 6551 was chosen for this application primarily for its internal baud rate generation and selection capabilities. The 6551 allows program controlled selection of standard baud rates from 50 to 19,200 baud.

IV-I: 6551 ADDRESSING

Each of the three 6551 ACIAs requires four bytes of address space. All three ACIAs are mapped into one 8K block of the on board address space (see the memory map). Although each device appears at multiple locations in the 8K block, they should be accessed only through the addresses shown in the memory map. Three groups of four addresses provide direct access to one each of the ACIAs (U15 = $A004-$A007, U14 = $A008-$A00B , U13 = $A010-$A013) The fourth group of four addresses ($A01C-$A01F) provides write-only access to all three ACIAs simultaneously. This last group of addresses can be used when the three ACIAs are being initialized. Any data written to one of this fourth group of addresses will be written to all three ACIAs.

CAUTION: Addresses $A01C-$A01F can only be used to write the same data to all three ACIAs simultaneously. Reading from these addresses will cause buffer conflicts on the internal bus and possibly damage the ACIAs.

IV-II: 6551 REGISTER ADDRESS ASSIGNMENTS

Each ACIA has four internal registers. The functions of these registers are described in the 6551 data sheet. The following table shows the address assignments for each of the registers in the three ACIAs:

	PORT 1 U15/J4	PORT 2 U14/J3	PORT 3 U13/J2	All PORTS WRITE ONLY
CONTROL REGISTER	$A007	$A00B	$A013	$A01F
COMMAND REGISTER	$A006	$A00A	$A012	$A01E
STATUS REGISTER	$A005	$A009	$A011	$A01D
DATA REGISTER	$A004	$A008	$A010	$A01C

TABLE IV-I: 6551 ADDRESS ASSIGNMENTS

©1982 GIMIX INC

IV-III: SERIAL I/O CONNECTIONS

The three 26-pin headers (J2, 3 and 4), located at the top edge of the board, provide external connection points for the I/O lines of the 6551s. All inputs and outputs are RS-232C compatible. The pin numbering of these connectors is indicated on the COMPONENT LAYOUT diagram. These pin numbers match the numbering of the standard 25-pin "D" type data connectors on GIMIX serial cable sets.

Pin 7 of each of the connectors is connected to signal ground. Pins 2, 3, 4, 5, 6, 8, 11, and 20 are connected to pins in the jumper areas (JA5, 6, and 7) located directly below each connector. These jumper areas allow the pinout to be arranged to suit different applications and peripherals (see section I-V). The remaining pins are unused and unconnected.

SECTION V: Z8038 FIO I/O INTERFACE UNIT

The Z8038 provides a path for data and command transfer between the on board CPU and the host system. It is the only means available for transfering information between the two. The Z8038 provides two methods of data transfer, the mailbox registers and a bidirectional FIFO (First In First Out) buffer. The mailbox allows transfer of data one byte at a time in both directions and would normally be used to transfer command and status information between the processors. The bidirectional FIFO buffer is 128 bytes deep and would normally be used to transfer blocks of data.

The Z8038 FIO has two "sides", port 1 is accessed by the host (system) CPU and port 2 is accessed by the on board CPU. Each port has its own set of 16 data and control registers. The port 2 registers are disabled on power up or after a reset and must be enabled from port 1 before they can be accessed.

See the Z8038 Data Sheet , Technical Manual, and section VI for more information.

V-I: Z8038 PORT 2 REGISTERS ($8000 and $8001)

The on board CPU accesses the Z8038 registers (port 2) through two address ($8000 and $8001). The remaining addresses in the 8K ($8000-$9FFF) are a repeat of these first two bytes.

The CPUs low order address line (A0) is connected to the Z8038 C/D (Control/Data) input so that when address $8000 (or any even address between $8000 and $9FFF) is accessed, the C/D line is LOW (0) and when address $8001 (or any odd address between $8000 and $9FFF) is accessed, C/D is high (1). The C/D line, in conjunction with a read or write operation, determines which of two internal states (0 or 1) the Z8038 is in (see the Z8038 documentation and section VI for more information).

Any of the 16 internal FIO registers can be accessed through these two bytes. In general, accessing a particular register is a two step process. The address of the desired register is first written to the FIO to select the register and then data can be read from or written to the register. See the section VI and the Z8038 TECHNICAL MANUAL for details on register accessing.

V-II: Z8038 PORT 1 REGISTERS (SS-30 BUS)

The host (system) CPU accesses the Z8038 registers (port 1) through four bytes in one of the 30 pin I/O bus slots. The actual address depends on the slot in which the board is installed. The table below shows the mapping of the board relative to the first (lowest) address of the I/O slot (board address 0).

BOARD ADDRESS 0	FIO DATA ADDRESS (C/D =0)
BOARD ADDRESS 1	FIO CONTROL ADDRESS (C/D =1)
BOARD ADDRESS 2	INTERRUPT VECTOR (READ) HARDWARE RESET (WRITE)
BOARD ADDRESS 3	SAME AS BOARD ADDRESS 2

The remaining 12 bytes of the I/O slot are a repeat of the above.

TABLE V-I: 30 PIN BUS REGISTER ASSIGNMENTS

The first two addresses are used to access the 16 internal registers of the FIO (port 1) as described above for port 2.

The third byte (board address 2) serves a dual purpose. Any write to this address causes a hardware reset of the on board CPU, the FIO and the ACIAs. This allows the I/O board to be reset without affecting the operation of the host CPU or the rest of the system. NOTE: After causing a hardware reset of the board by writing this byte, the software should allow approximately 2 microseconds for the board to come out of the reset state before attempting to initialize the FIO.

After port 1 of the FIO generates an interrupt, a read of the third byte returns the contents of the Interrupt Vector Register in the FIO, which can be used to identify the cause of the interrupt. The interrupt vector function must be enabled by proper programming of the FIO registers.

Note: Reading the Interrupt Vector Register returns invalid data if the FIO has not generated an interrupt. Because of this, the Interrupt Vector Register alone can not be used to determine if the FIO actually generated an interrupt. See SECTION VI-VI for more information.

V-III: Z8038 OPERATING MODE

Both sides of the FIO are used in the non Z-bus mode. Port 1 is hardware configured for non Z-bus operation. Port 2 of the FIO must be programmed for non Z-bus operation when port 1 is initialized.

SECTION VI: PROGRAMMER'S NOTES ON THE Z8038

VI-I: TRANSFERING FIFO DIRECTION CONTROL

The direction of data flow through the FIFO can be controlled from either the port 1 or the port 2 side of the Z8038. Bit 5 of port 1, Control Register 3 determine which side of the Z8038 has control over the direction of data transfer through the FIFO. After a reset, this bit is clear (0); port 1 has control of the direction. Control can be transferred to port 2 by setting the bit (1).

In order to transfer control back to port 1, you MUST first clear the FIFO buffer (b6 of Control Register 3 = 0) and set the Data Direction for port 1-to-port 2 transfer by setting b4 of Control Register 3, port 2. Only after the above conditions are met can control be transferred back to port 1.

VI-II: ACCESSING REGISTERS

In the "non Z-BUS CPU mode" mode, all registers must be accessed in a two step process. First, the internal register address must be written to a pointer in the Z8038 to select the register. Then data can be read from or written to the selected register. When the Z8038 is waiting for a new address, this is called state 0; when the register is selected for access, that is state 1. The series of reads and writes at the start of the sample program (labeled "reset sequence for FIO" in the comments) is intended to force state 0, regardless of the starting state of the Z8038.

NOTE: A read can be performed in state 0: the Z8038 just returns the value of the last register accessed. Thus a register can be pointed to once and then read several times. However, a register must be reselected for each write.

CAUTION: Interrupts must be masked when accessing the FIO registers if it is possible for the interrupt service routine to access the same FIO. If an interrupt were to occur after an FIO register was selected but before the data read or write, the register address written by the interrupt routine would be interpreted as data by the FIO! This could cause some very strange (and hard to locate) bugs.

VI-III: SOFTWARE RESETTING THE Z8038

Software resetting the Z8038 requires forcing it into state 0, then toggling bit 0 of Control Register 0. For an example, see the code from the sample program mentioned above. For proper operation, both sides must be reset. However, port 1 must be reset first, and port 2 enabled (by setting bit 0 of port 1's Control Register 2) before port 2 can be accessed at all.

Note: A software reset of the Z8038 does not affect the CPU or the ACIAs.

VI-IV: FIFO BUFFER ACCESS

The 128 byte FIFO memory in the Z8038 is directly accessible. Writes to or reads from board address 0 (port 1) or address $8000 (port 2) move data in or out of the FIFO; subject to the direction control bit in the FIO registers. Also see below on the Data Buffer register.

VI-V: THE DATA BUFFER REGISTER

The Data Buffer Register is actually the last byte of the FIFO memory. It is readable only at the current output end of the FIO, and always reads 00 at the input end. Accessing the DBR as control address 15 is the same as accessing the data address of the Z8038, and causes data in the FIFO to be bumped along.

VI-VI: INTERRUPT OPTIONS

The FIO interrupt operations are rather confusing. There are seven functions which can cause an interrupt. These are: byte count match, pattern match, FIO empty, FIO full, overflow/underflow, data direction change, and mailbox message. For each of these functions there are three bits in a register to control them. These bits are "read/write" but not directly.

When read these bits give the status of the function. One bit is Interrupt Pending (IP), which indicates that the condition causing the interrupt has occurred but not yet been serviced. The next bit is Interrupt Enable (IE), which indicates whether the function has been enabled to generate an interrupt to the CPU. The third bit is Interrupt Under Service, which indicates that the CPU is processing the interrupt.

These bits can also be written to; but the three bits written are not stored directly; rather they are taken as a command to set or clear one or two of the bits in accordance with with the "Masks for interrupt control" table in the register description section.

The Interrupt Vector Register indicates which of the seven functions generated the interrupt, provided that Control Register 0 has the following settings:

```
b7 (Master Interrupt Enable) = 1
b5 (No vector on IRQ)        = 0
b4 (Vector includes status)  = 1
```

This register is accessible directly on port 1 (HOST CPU) reading address 2. On port 2, the Interrupt Vector Register is accessible only through the point-and-read method. Directly reading the Interrupt Vector register causes an FIO Interrupt Acknowledge Cycle. If an interrupt is pending, the data returned will be the contents of this register. Also the IUS bit of the appropriate function will be set. This bit can only be cleared by writing 001 or 011 to the control bits for the function.

The value read from the Interrupt Vector Register is an 8-bit number. Bit 0 and bits 4 through 7 return whatever the CPU writes there. Bits 1, 2, and 3 indicate the function which caused the interrupt. Thus the contents of this register can be used to point into a table of interrupt vectors, to select the correct handler for the current interrupt automatically. The other bits can be set to 0, or to the base address of the table.

NOTE: direct access of this register is possible only when the 8038 has an interrupt pending. At any other time, the device does not respond to the read and the data returned invalid. This may occur if some other device generated the interrupt. The handler program must check the Interrupt Status Registers to confirm that the interrupt did indeed come from the Z8038.

VI-VII: MESSAGE INTERRUPT

The Message interrupt operates as follows: the IP bit is set when the other CPU writes a message byte to the mailbox. IP is cleared when the Message In register is read. This is the only IP bit which does not have to be cleared by an 001 or 101 mask.

VI-VIII: PATTERN MATCH INTERRUPT

The operation of the Pattern Match function is rather peculiar. The match is against the byte in the Data Buffer Register at the same end of the FIO as the Pattern Match Register. Thus IP is set by the last byte written or the NEXT byte to be read. Note that this differs from the brief description in the manual, which says that the byte matched is the last byte read or written. A match on read is triggered before the byte is read; that is, when the preceding byte is read and the matching byte drops into the Data Buffer Register.

There is a Pattern Match bit in addition to the Pattern Match IP bit, which is set when the match occurs. However, this bit is cleared

if a data read or write moves new data into the Data Buffer Register (provided the new data does not match), or if the pattern match value or pattern match mask are changed so that the match no longer occurs. The IP bit remains set until cleared by an 001 or 101 mask.

VI-IX: BUFFER EMPTY INTERRUPT

The Buffer Empty interrupt is triggered when the last byte is read from the FIFO buffer. The Buffer Empty bit is set at this time. The IP bit is also set by this transition. The Buffer Empty bit is reset as soon as data is written to the FIFO buffer, but IP remains latched. This interrupt is also triggered when the buffer is cleared by the Clear FIFO Buffer function controlled by b6 of Control Register 3. It is NOT triggered when the part is reset even though the buffer is empty and the Buffer Empty bit is set after reset. Note: when the Buffer Empty interrupt is triggered, IP is set at both ends, and must be separately cleared at both ends. However, if IP is set with a 100 mask, it is set only at that end.

VI-X: BUFFER FULL INTERRUPT

The Buffer Full interrupt is triggered when the 128th byte is written to the FIFO buffer. The IP bit is set by this transition, and the Buffer Full bit is also set at this time. The Buffer Full bit is reset as soon as data is read from the FIFO buffer, but IP remains latched. Note: when the Buffer Full interrupt is triggered, IP is set at both ends, and must be separately cleared at both ends. However, if IP is set with a 100 mask, it is set only at that end.

VI-XI: BYTE COUNT INTERRUPT

The Byte Count interrupt is triggered whenever the byte count changes and the new value matches the current trigger value, or the trigger value is changed and matches the current byte count. This can happen whenever a byte is read from or written to the buffer, or the buffer is cleared and the match value is 0. IP remains latched even if the byte count or match value or both are changed. NOTE: the Byte Count trigger value is limited to seven bits. This means that a value of 128 is the same as a value of 0. Therefore if the trigger is 0 and the byte count is 128 a spurious match will be generated.

VI-XII: DATA DIRECTION CHANGE INTERRUPT

The Data Direction Change interrupt is triggered whenever b4 of Control Register 3 of the controlling port is changed. It is also triggered when the controlling port is changed, and the other port has a different setting for the Data Direction bit. Note: when the Data Direction interrupt is triggered, IP is set at both ends, and must be separately cleared at both ends. However, if IP is set with a 100 mask, it is set only at that end.

VI-XIII: OVERFLOW/UNDERFLOW INTERRUPT

The Overflow/Underflow Interrupt can be triggered by either of two events: when a data byte is written to the buffer while the buffer is full, or when a data byte is read from the buffer while the buffer is empty. The first event sets the Overflow Error bit and Overflow/Underflow IP bit at the sending end; the second sets the Underflow Error bit and Overflow/Underflow IP bit at the receiving end. These error bits remain set until the Overflow/Underflow IP bit at that end is cleared with an 001 or 101 mask.

Underflow and Overflow occur when data is lost or false data is generated. In the case of Underflow, the byte which is returned in response to the excess read is the last byte in the buffer (or 0 if the buffer has been cleared). This value will appear until new data is written at the other end. In the case of Overflow, the data lost is the excess byte written. Data movement through the buffer stops when the 128th byte is written; any further writes are disregarded.

The IP bit for any function can be set by writing a 100 mask to the control bits. This has the same effects as a normal interrupt, including the setting of the vector register, interrupt output, and setup for the INTACK cycle. If the message IP bit is set this way, it is still clearable by reading the Message Input register, and the Mailbox Full bit is set on the other port.

SECTION VII: SAMPLE PROGRAMS

Following are two sample programs which demonstrate some of the functions of the board. The first program, FIODEMO 1, resets the board, initializes the port 1 (host) side of the FIO, and then enables port 2. The second program, FIODEMO 2, initializes the 6551s using data read from the DIP-switch to determine the baud rate. It then waits for the FIO (port 2) to be enabled by port 1. If the FIO is not enabled within a prescribed time limit, a message is sent to the serial port and one of the status LEDs is illuminated. If the FIO is enabled within the time limit, FIO port 2 is initialized, a different message is sent, and the other status LED is illuminated.

```
                * FIODEMO 1
                * Sample reset sequence for the GIMIX intelligent
                * Serial I/O board. Hardware resets the board, programs
                * port 1, and enables port 2 for programming

                * Base address of I/O slot
E010    SLOT    EQU     $E010

                * FIO registers
E010    FIODAT  EQU     SLOT
E011    FIOCTL  EQU     SLOT+1
E012    FIORST  EQU     SLOT+2

                * Offsets for indirect addressing of FIO registers
0000    CTLRG0  EQU     0       reset/mode control
0001    CTLRG1  EQU     1       mailbox/req control
0002    ISTAT0  EQU     2       message IRQ control
0003    ISTAT1  EQU     3       data dir & pat mat IRQ ctl
0004    ISTAT2  EQU     4       byte cnt & ovr/undrflo IRQ ctl
0005    ISTAT3  EQU     5       full/empty IRQ ctl
0006    IRQVEC  EQU     6       interrupt table pointer
0007    BYTCNT  EQU     7       byte cnt
0008    BYTCMP  EQU     8       byte cnt trigger
0009    CTLRG2  EQU     9       port 2 ctl
000A    CTLRG3  EQU     10      data dir ctl
000B    MSGOUT  EQU     11      outgoing message byte
000C    MSGIN   EQU     12      incoming message byte
000D    PATMAT  EQU     13      pattern to match
000E    PATMSK  EQU     14      pattern match don't care mask
000F    DATBFR  EQU     15      last byte read or written to FIFO


                * Hardware reset the board

0000 B7 E012    START   STA     FIORST  Write resets 6809, FIO, ACIAs
0003 8D 22              BSR     DELAY   Wait to be sure its out of reset


                * Soft reset sequence for FIO
                * resets FIO and guarantees state 0
                * regardless of its initial state.

0005 B6 E011            LDA     FIOCTL
0008 4F                 CLRA
0009 B7 E011            STA     FIOCTL
000C 86 01              LDA     #1
000E B7 E011            STA     FIOCTL
0011 4F                 CLRA
0012 B7 E011            STA     FIOCTL

                * FIO is now out of reset in state 0
                * and ready for programming
```

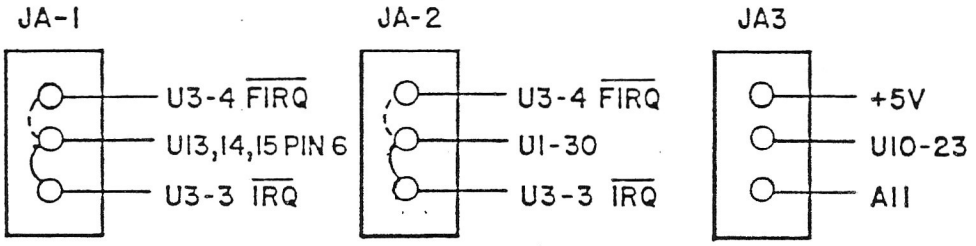
```
                    * Init port 1 side and enable port 2

                    * Port 2 must be setup for NON-Z-BUS mode

0015 86 00                  LDA     #CTLRG0   Set port 2 mode to NON-Z-BUS &
0017 C6 16                  LDB     #$16      enable VIS & RJA
0019 8D 0F                  BSR     STUFF

001B 86 0A                  LDA     #CTLRG3   Enable FIO buffer (side 1 has
001D C6 40                  LDB     #$40      control, direction = out)
001F 8D 09                  BSR     STUFF

                    * Port 2 must be enabled before it can be programmed

0021 86 09                  LDA     #CTLRG2   Enable port 2 side
0023 C6 01                  LDB     #1
0025 8D 03                  BSR     STUFF

                    *
                    * End of port 1 init sequence
                    *

                    * Subroutines

                    * Delay a couple of microseconds

0027 8D 00          DELAY   BSR     DELAY1
0029 39             DELAY1  RTS

                    * Write to FIO register

                    * ACCA = offset to register
                    * ACCB = data to write

002A B7 E011        STUFF   STA     FIOCTL    Point to register
002D F7 E011                STB     FIOCTL    Send data
0030 39                     RTS

                            END
```

```
                    * FIODEMO 2
                    * Sample program for side 2 reset and initialization
                    * Demonstrates use of sense switch and status LED

                    * 6551 addresses
               A004 PORT1   EQU    $A004
               A008 PORT2   EQU    $A008
               A010 PORT3   EQU    $A010

                    * switch banks
               4000 SWTCH1  EQU    $4000
               6000 SWTCH2  EQU    $6000

                    * status LEDs
               4000 DISP1   EQU    SWTCH1    DS1
               6000 DISP2   EQU    SWTCH2    DS2

                    * FIO interface addresses
               8000 FIODAT  EQU    $8000
               8001 FIOCTL  EQU    $8001

                    * Offsets for indirect addressing of FIO registers
               0000 CTLRG0  EQU    0         reset/mode control
               0001 CTLRG1  EQU    1         mailbox/REQ control
               0002 ISTAT0  EQU    2         message IRQ ctl
               0003 ISTAT1  EQU    3         data dir & pat mat IRQ ctl
               0004 ISTAT2  EQU    4         byte cnt & ovr/undrflo IRQ ctl
               0005 ISTAT3  EQU    5         full & empty IRQ ctl
               0006 IRQVEC  EQU    6         interrupt table pointer
               0007 BYTCNT  EQU    7         byte count
               0008 BYTCMP  EQU    8         byte count trigger
               0009 CTLRG2  EQU    9         port 2 control
               000A CTLRG3  EQU    10        data direction control
               000B MSGOUT  EQU    11        outgoing message byte
               000C MSGIN   EQU    12        incoming message byte
               000D PATMAT  EQU    13        pattern to match
               000E PATMSK  EQU    14        pattern match don't care mask
               000F DATBFR  EQU    15        last byte read or written to FIO

0000 10CE 27FF  START  LDS    #$27FF    put the stack in RAM #2

                     * Initialize 6551s - get baud rate from switches
0004 8E A004           LDX    #PORT1    point to port 1
0007 B6 4000           LDA    SWTCH1    get switch setting
000A 17 0088           LBSR   SERINT    initialize it
000D 8E A008           LDX    #PORT2    next port
0010 B6 4000           LDA    SWTCH1
0013 44                LSRA             use upper half of switch byte
0014 44                LSRA
0015 44                LSRA
0016 44                LSRA
0017 8D 7C             BSR    SERINT    initialize it
0019 8E A010           LDX    #PORT3    last port
001C B6 6000           LDA    SWTCH2    read second switch
001F 8D 74             BSR    SERINT    initialize it
```

```
                    * This section waits for port 2 to be enabled
                    * If its not enabled in a specified time, assume
                    * somethings wrong with the host CPU and report it.
                    * If it is enabled in time, report that.
                    *
                    * Before port 2 is enabled, its buffers are in
                    * Tri-state and reading FIOCTL returns garbage.
                    * Once it has been enabled it will read 01.
                    * To guard against a false ready indication,
                    * we first look for a 1 and then test for
                    * the prescence of the FIO pattern match register.

                    * Look for 01 at FIOCTL

0021 C6 78                  LDB   #120      set up delay time
0023 34 04                  PSHS  B
0025 108E E86C      WAIT0   LDY   #59500
0029 B6 8001        WAIT    LDA   FIOCTL    look for side 2 to be enabled
002C 81 01                  CMPA  #01       does it read 01?
002E 27 0C                  BEQ   TEST      yes, it may be ready
0030 31 3F                  LEAY  -1,Y      no, count down and try again
0032 26 F5                  BNE   WAIT
0034 6A E4                  DEC   0,S
0036 26 ED                  BNE   WAIT0
0038 35 04                  PULS  B
003A 20 39                  BRA   NOGOOD    until count runs out

                  * Check to see if the FIO is actually enabled

003C B6 8001        TEST    LDA   FIOCTL    do soft reset sequence
003F 4F                     CLRA
0040 B7 8001                STA   FIOCTL
0043 86 01                  LDA   #1
0045 B7 8001                STA   FIOCTL
0048 4F                     CLRA
0049 B7 8001                STA   FIOCTL
004C 86 00                  LDA   #CTLRG0   setup port 2 registers
004E C6 12                  LDB   #$12
0050 8D 5D                  BSR   STUFF
0052 C6 55                  LDB   #$55      test pattern
0054 86 0D                  LDA   #PATMAT   write it in pattern match reg.
0056 8D 57                  BSR   STUFF
0058 8D 5C                  BSR   GRAB
005A 81 55                  CMPA  #$55      does it read back ok?
005C 26 CB                  BNE   WAIT      no, start over
005E C6 AA                  LDB   #$AA      try another pattern
0060 86 0D                  LDA   #PATMAT   write it
0062 8D 4B                  BSR   STUFF
0064 8D 50                  BSR   GRAB
0066 81 AA                  CMPA  #$AA      does it read back?
0068 26 BF                  BNE   WAIT      no, start over
006A 8E 00CD                LDX   #CPUUP    yes, report CPU up
```

```
006D 8D 1F              BSR     SEND
006F 108E 6000          LDY     #DISP2      turn on DS2
0073 20 09              BRA     LEDON

0075 8E 00BD    NOGOOD  LDX     #CPUDWN     wait expired - CPU down
0078 8D 14              BSR     SEND
007A 108E 4000          LDY     #DISP1      turn on DS1

                * This section lights the status LED, pointed to by "Y"

007E EE A4      LEDON   LDU     0,Y         access the area repeatedly
0080 EE A4              LDU     0,Y
0082 EE A4              LDU     0,Y
0084 EE A4              LDU     0,Y
0086 EE A4              LDU     0,Y
0088 EE A4              LDU     0,Y
008A 20 F2              BRA     LEDON

                * Output a string ending in 04

008C 8D 14      SEND0   BSR     SEROUT
008E A6 80      SEND    LDA     0,X+
0090 81 04              CMPA    #4
0092 26 F8              BNE     SEND0
0094 39                 RTS

                * Initialize the 6551 pointed to by the X reg.
                * The lower half of ACCA (the value read from
                * the DIP-switch) is used to determine the
                * baud rate.

0095 A7 01      SERINT  STA     1,X         reset the part
0097 C6 0B              LDB     #$0B
0099 E7 02              STB     2,X         set up control functions
009B 84 0F              ANDA    #$0F
009D 8A 90              ORA     #$90        set baud rate & format
009F A7 03              STA     3,X
00A1 39                 RTS

                * Output the character in ACCA to port 1

00A2 F6 A005    SEROUT  LDB     PORT1+1
00A5 C4 70              ANDB    #$70        wait for TxDRE & DCD & DSR
00A7 C1 10              CMPB    #$10
00A9 26 F7              BNE     SEROUT
00AB B7 A004            STA     PORT1
00AE 39                 RTS

                * Write to selected FIO register

                * ACCA = offset to register
                * ACCB = data to write

00AF B7 8001    STUFF   STA     FIOCTL      select register
```

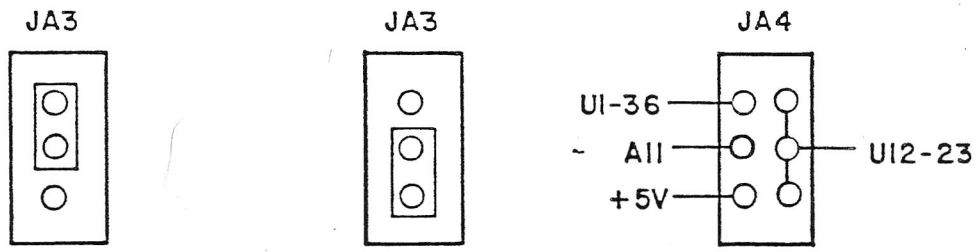
```
00B2 F7 8001            STB     FIOCTL    write to it
00B5 39                 RTS

                * Read a selected FIO register

                * Input:  ACCA = offset to register
                * Output: ACCA = data read

00B6 B7 8001    GRAB    STA     FIOCTL    select register
00B9 B6 8001            LDA     FIOCTL    read from it
00BC 39                 RTS

                * Messages

00BD 53 59      CPUDWN  FCC     /SYSTEM CPU DOWN/,4

00CD 53 59      CPUUP   FCC     /SYSTEM NOW OPERATIONAL/,4

                        END
```



6551 INTERRUPT SELECT FIGURE A.

FIO INTERRUPT SELECT FIGURE B.

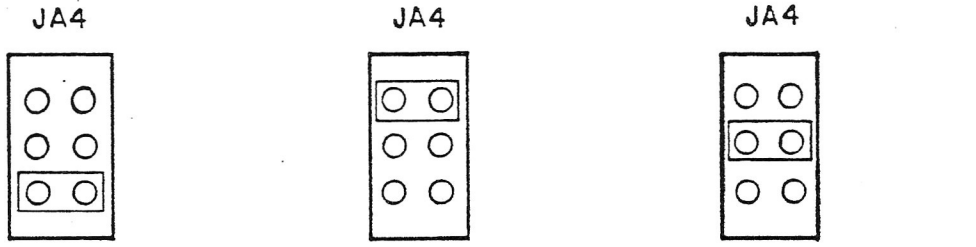
PROM SOCKET U10 SIZE SELECT FIGURE C.



2K DEVICES FIGURE D.

4 & 8K DEVICES FIGURE E.

PROM/RAM SOCKET U12 DEVICE SELECT FIGURE F.

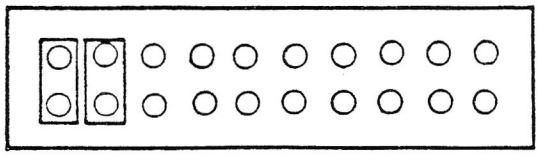


2K PROM FIGURE G.

2K RAM FIGURE H.

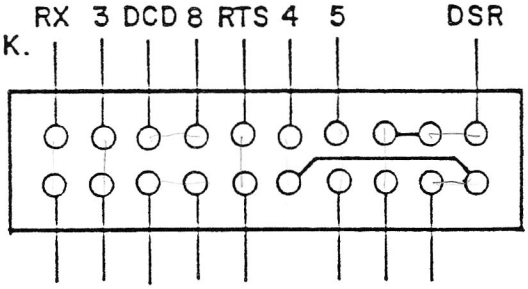
4 & 8K PROM FIGURE I.

FIGURE J.



STANDARD RS-232 TERMINAL CONFIGURATION NO HANDSHAKE

FIGURE K.



JA-5,6,7

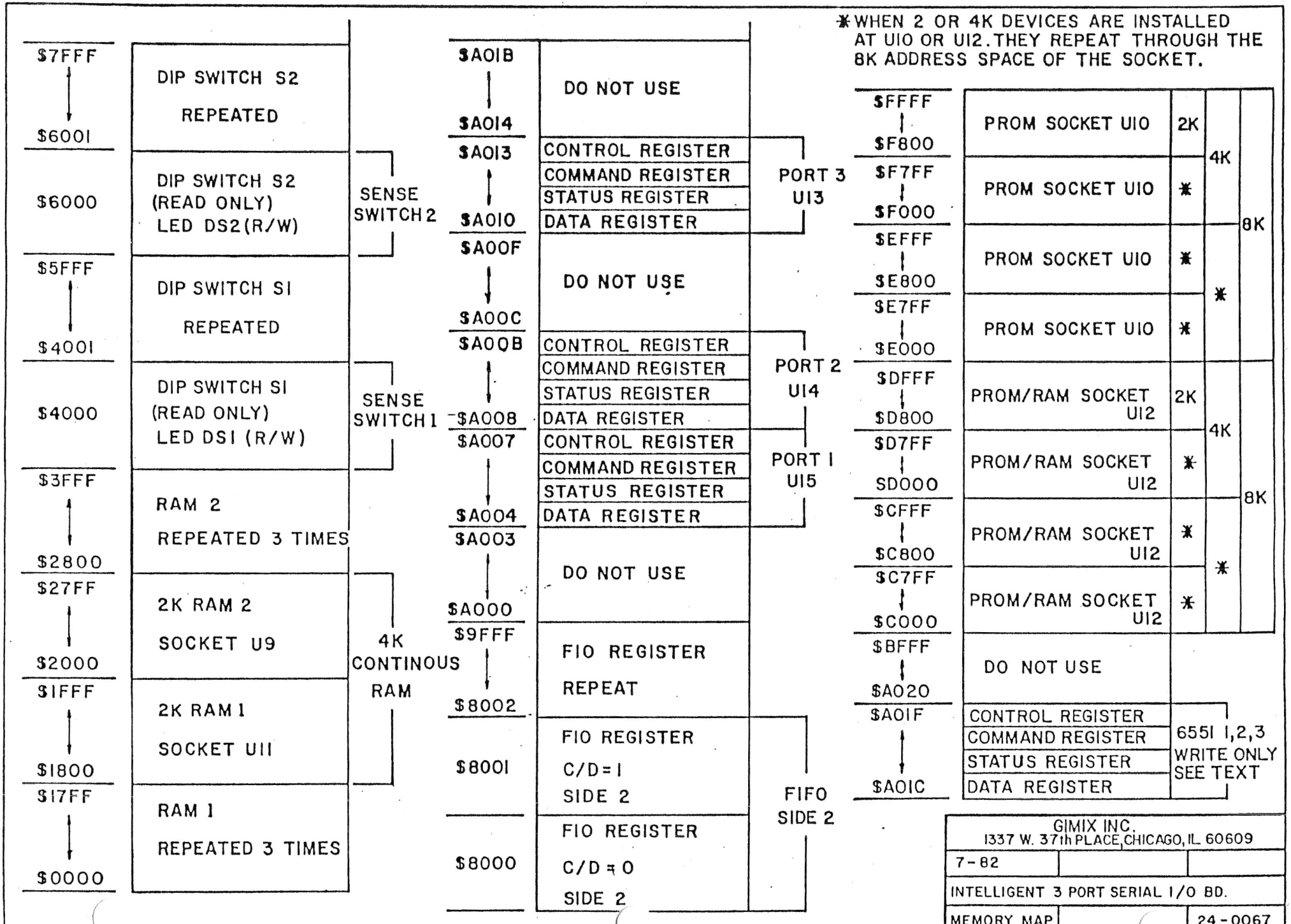
NUMBERS INDICATE 25 PIN "D" CONN. PINOUTS WHEN USED WITH GIMIX CABLES.

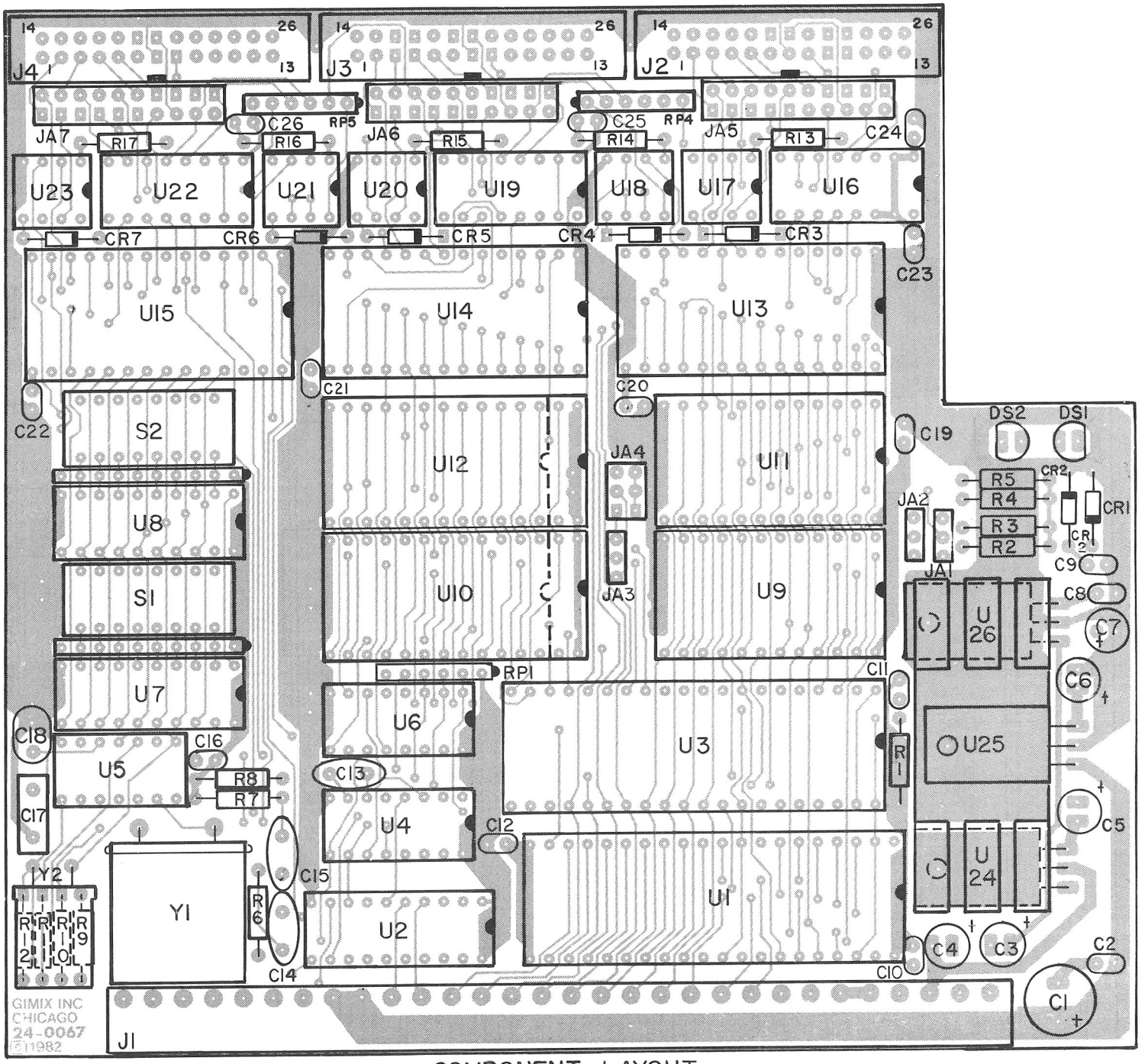
= STANDARD JUMPER = OPTIONAL JUMPER

NOTE

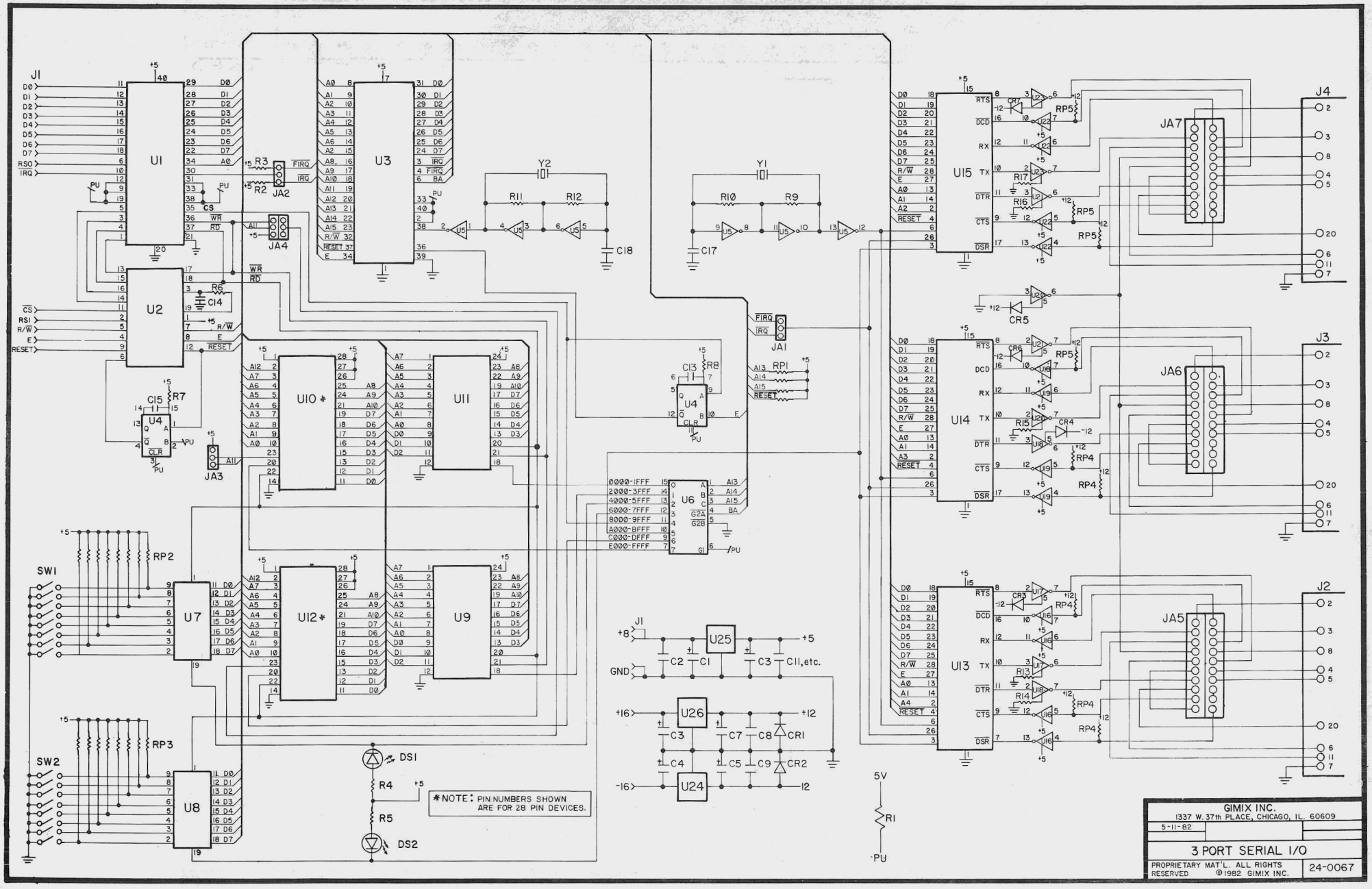
JA1 & JA2 COMES FROM FACTORY WITH STANDARD JUMPERS INSTALLED (= $\overline{IRQ}$).
TO USE $\overline{FIRQ}$ CUT STANDARD JUMPERS AND INSTALL OPTIONAL JUMPERS.

GIMIX INC. 1337 W. 37th PLACE, CHICAGO, IL. 60609		
8-82		
INTELLIGENT 3 PORT SERIAL I/O		
INTERRUPT & PROM SOKT. JPS.		24-0067


$7FFF
$6001
DIP SWITCH S2
REPEATED
$6000
DIP SWITCH S2
(READ ONLY)
LED DS2(R/W)
SENSE SWITCH 2
$5FFF
$4001
DIP SWITCH S1
REPEATED
$4000
DIP SWITCH S1
(READ ONLY)
LED DS1 (R/W)
SENSE SWITCH 1
$3FFF
$2800
RAM 2
REPEATED 3 TIMES
$27FF
$2000
2K RAM 2
SOCKET U9
$1FFF
$1800
2K RAM 1
SOCKET U11
4K CONTINOUS RAM
$17FF
$0000
RAM 1
REPEATED 3 TIMES
$A01B
$A014
DO NOT USE
$A013
$A010
CONTROL REGISTER
COMMAND REGISTER
STATUS REGISTER
DATA REGISTER
PORT 3 U13
$A00F
$A00C
DO NOT USE
$A00B
$A008
CONTROL REGISTER
COMMAND REGISTER
STATUS REGISTER
DATA REGISTER
PORT 2 U14
$A007
$A004
CONTROL REGISTER
COMMAND REGISTER
STATUS REGISTER
DATA REGISTER
PORT 1 U15
$A003
$A000
DO NOT USE
$9FFF
$8002
FIO REGISTER
REPEAT
$8001
FIO REGISTER
C/D=1
SIDE 2
$8000
FIO REGISTER
C/D=0
SIDE 2
FIFO SIDE 2
*WHEN 2 OR 4K DEVICES ARE INSTALLED AT U10 OR U12. THEY REPEAT THROUGH THE 8K ADDRESS SPACE OF THE SOCKET.
$FFFF
$F800
PROM SOCKET U10
2K
$F7FF
$F000
PROM SOCKET U10
*
4K
$EFFF
$E800
PROM SOCKET U10
*
$E7FF
$E000
PROM SOCKET U10
*
*
8K
$DFFF
$D800
PROM/RAM SOCKET U12
2K
$D7FF
$D000
PROM/RAM SOCKET U12
*
4K
$CFFF
$C800
PROM/RAM SOCKET U12
*
$C7FF
$C000
PROM/RAM SOCKET U12
*
*
8K
$BFFF
$A020
DO NOT USE
$A01F
$A01C
CONTROL REGISTER
COMMAND REGISTER
STATUS REGISTER
DATA REGISTER
6551 1,2,3
WRITE ONLY
SEE TEXT
GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL 60609
7-82
INTELLIGENT 3 PORT SERIAL I/O BD.
MEMORY MAP
24-0067



COMPONENT LAYOUT


J1
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10 *
U11
U12 *
U13
U14
U15
U24
U25
U26
Y1
Y2
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
RP1
RP2
RP3
RP4
RP5
C1
C2
C3
C4
C5
C7
C8
C9
C11,etc.
C13
C14
C15
C17
C18
CR1
CR2
CR3
CR4
CR5
CR6
CR7
DS1
DS2
SW1
SW2
JA1
JA2
JA3
JA4
JA5
JA6
JA7
J2
J3
J4
0000-1FFF
2000-3FFF
4000-5FFF
6000-7FFF
8000-9FFF
A000-BFFF
C000-DFFF
E000-FFFF
*NOTE: PIN NUMBERS SHOWN ARE FOR 28 PIN DEVICES.
GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL. 60609
5-11-82
3 PORT SERIAL I/O
PROPRIETARY MAT'L. ALL RIGHTS RESERVED
©1982 GIMIX INC.
24-0067