MEMORY ADDRESSING, VECTORED INTERRUPT, AND SERIAL I/O

Memory and I/O Addressing The CPU/8 processor card may contain up to 3.5K bytes of onboard RAM and ROM. This memory occupies a contiguous 4K address space which is jumper selectable to start at Ø, 8ØØØH, or EØØØH (see Figure 1). The CPU/8 board is shipped prewired to start at address Ø. This may be modified by removing jumper "J" and replacing it with jumpers from J to S or J to T. The onboard I/O port address space follows the onboard memory address space as shown in Figure 1.

Space for three 2708 UV erasable PROM's or three 2308 masked ROM'S is provided on the CPU/8. The ROM's occupy the first 3K bytes of the 4K address space. Also provided on the CPU/8 is 512 bytes of read/write memory (RAM). This RAM occupies the upper 1K of address space. Figure 1 shows to ½K block of RAM. These are in fact the same ½K block of RAM as the address of each cell is not unique. Two addresses exist for each byte of RAM. Address D7FH contains the same information as address F7FH. This is caused by the fact that address bit 8 is not decoded when addressing onboard RAM.

USART and Baud Rate Latch The use of the USART and baud rate latch is discussed in a separate section "Programming the SER/8 serial option."

Real-Time Clock The real-time clock (RTC) circuit generates and interrupts for every positive going edge of the 50Hz or 60 Hz line frequency. This interrupt is latched and remains on until it is reset by issuing an output command to the real-time clock port. It is the programmer's responsibility to reset the RTC, before reenabling interrupts, when he services the clock. This is done by doing an OUT 8, OUT 88H or OUT ØE8H instruction depending upon the location of the onboard ports. Vectored interrupt 1 (trap address 3ØH) is generally used for the real-time clock interrupt service routine. The real-time clock is physically connected on the CPU board by a jumper from the RTC pad in the upper left hand side of the board to VI1 in the interrupt jumper area.

Single-Step Logic The single step logic hardware uses vectored interrupt Ø for the purpose of executing a single instruction of a program being tested and returning to a fixed location (38H) in RAM. Issuing an output instruction to the single-step port causes the single step logic to be enabled. This also causes all other interrupts to be disabled (masked). The logic will count out 2 instruction cycles and then generate an interrupt which is vectored to location 38H. The single-step logic is immediately reset and its interrupts that were masked off are unmasked. Note that it is still not possible to be interrupted until an enable interrupt (EI) instruction is executed.

The two instructions normally executed after the output instruction are a return (to the program bring single-stepped) and one instruction out of the program being single-stepped. For an example of the use of the single-step logic see the assembly listing for the POLY 88 resident monitor.

Using the vectored interrupts The 8 vectored interrupt traps in the 8080A CPU chip are located every 8 bytes in the first 64 bytes of memory (see Figure 2). The first trap is also used by the reset function on the CPU so is generally not available for use as an interrupt. The last trap (38H) is used by the single-step logic on the CPU/8 card so it is also not available if the single-step function is to be used.

Consequently, only 6 (or 7) vectored interrupts are generally available. The 8 bytes at each trap location are usually not enough for the interrupt service routine, but are enough to save the state of the CPU and jump to the actual routine (see Figure 3).

The address of the service routines are stored in a table (ISRTBC) in RAM so that the service routines may be modified dynamically. The routine IORET restores the CPU registers and enables interrupts before returning to the interrupted program.

Power on reset When power is applied to the CPU a reset signal is generated for approximately ½ second. This causes the program counter to be reset to zero and the interrupts to be disabled. When this signal is released execution begins sequentially from location zero. The first 5 locations in memory (Ø to 4) are generally used for a portion of the cold-start routine and a jump in the next 3 (5-7) jumps to the rest of the cold start routine (see POLY 88 resident monitor). This is done because location 8 is generally used for vectored interrupt 6.

CPU/8 I/O ADDRESSING

Address (Hex)*			R/W	Function
J	S	T		
Ø,2	8Ø,82	EØ,E2	R	USART status byte
Ø,2	8Ø,82	EØ,E2	W	USART command/node
1,3	81,83	E1,E3	R	USART received data
1,3	81,83	E1,E3	W	USART transmit data
4-7	84-87	E4-E7	R	Unused
4-7	84-87	E4-E7	W	Baud Rate Latch
8-B	88-8B	E8-EB	R	Unused
8-B	88-8B	E8-EB	W	Reset real-time clock
C-F	8C-8F	EC-EF	R	Unused
C-F	8C-8F	EC-EF	W	Start single step

CPU/8 MEMORY ADDRESSING

Address*			Function
J	S	T	
Ø-3FF	8ØØØ-83FF	EØØØ-E3FF	ROM #1
4ØØ-7FF	84ØØ-87FF	E4ØØ-E7FF	ROM #2
8ØØ-BFF	88ØØ-8BFF	E8ØØ-EBFF	ROM #3
CØØ-DFF	8CØØ-8DFF	ECØØ-EDFF	Onboard RAM
EØØ-FFF	8EØØ-8FFF	EEØØ-EFFF	Onboard RAM **

* Jumpers J, S and T determine the base address of the CPU memory and I/O address space

**This is the same RAM as the 512 bytes below it. (Bit 9 of the address is not decoded).

Figure 1

Vectored Interrupt Locations

Interrupt #	Address (Hex)
Ø	38*
1	3Ø
2	28
3	2Ø
4	18
5	1Ø
6	8
7	Ø**

* This location is also used for the single-step trap.

**This location is also used for the power-on reset.

Figure 2

INTRODUCTION

The SER/8 serial port option for the CPU/8 CPU card provides a very flexible serial communications interface for the POLY-88. The SER/8 provides a universal synchronous/asynchronous receiver/transmitter and a software controllable baud rate generator. The USART may interface to two serial I/O devices. The device and its baud rate may be changed under the control of one output port. These interface through two "minicards" which mount to the rear of the POLY-88 chassis. Interface cards are available for RS-232-C, current loop, Kansas City (Byte) Standard audio cassette, and the 1200 baud bi-phase cassette.

This note describes the SER/8 option from a functional standpoint and then describes the various operating modes of the USART and how the SER/8 may be programmed.

COMMUNICATIONS FORMATS

Serial communications, either on a data link or with a local peripheral, occurs in one of two basic formats; asynchronous or synchronous. These formats are similar in that they both require framing information to be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data, or messages. Since the synchronous format is more efficient than the asynchronous format but requires more complex decoding it is typically found on high-speed data links, while the asynchronous format is used on lower speed lines.

The asynchronous format starts with the basic data bit to be transmitted and adds a "START" bit to the front of them and one or more "STOP" bits behind them as they are transmitted. The START bit is a logical zero, or SPACE, and is defined as the positive voltage level by RS-232-C. The STOP bit is a logical one, or

MARK, and is defined as the negative voltage level by RS-232-C. In current loop applications current flow normally indicates a MARK and lack of current a SPACE. The START bit tells the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter. Since this synchronization only has to last for the duration of the character (the next character will contain a new START bit), this method works quite well assuming a properly designed receiver. One or more STOP bits are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and to give the receiver time to "catch up" with the transmitter if its basic clock happens to be running slightly slower than the transmitter clock. If, on the other hand, the receiver clock happens to be running slightly faster than the transmitter clock, the receiver will perceive gaps between characters but will still correctly decode the data. Because of this tolerance to minor frequency deviations, it is not necessary that the transmitter and receiver clocks be locked to the identical frequency for successful asynchronous communication.

The synchronous format, instead of adding bits to each character, groups characters into records and adds framing characters to the record. The framing characters are generally known as SYN characters and are used by the receiver to determine where the character boundaries are in a string of bits. Since synchronization must be held over a fairly long stream of data, bit synchronization is normally either extracted from the communication channel by the modem or supplied from an external source.

An example of the synchronous and asynchronous formats is shown in Figure 1. The synchronous format shown is fairly typical in that it requires two SYN characters at the start of the message. The asynchronous format, also typical, requires a START bit preceding each character and a single STOP bit following it. In both cases, two 8-bit characters are to be transmitted. In the

asynchronous mode 10 n bits are used to transmit n characters and in the synchronous mode 8n + 16 bits are used. For the example shown the asynchronous mode is actually more efficient, using 20 bits versus 32. To transmit a thousand characters in the asynchronous mode, however, take 10,000 bits versus 8,016 for the synchronous format mode. For long messages the synchronous format becomes much more efficient than the asynchronous format; the crossover point for the examples shown in Figure 1 is eight characters, for which both formats require 80 bits.

In addition to the differences in format between synchronous and asynchronous communication, there are differences with regards to the type of modems that can be used. Asynchronous modems typically employ FSK (Frequency Shift Keying) techniques which simply generate one audio tone for a MARK and another for a SPACE. The receiving modem detects these tones on the telephone line, converts them to logical signals, and presents them to the receiving terminal. Since the modem itself is not concerned with the transmission speed, it can handle baud rates from zero to its maximum speed. Synchronous modems, in contrast to asynchronous modems, supply timing information to the terminal and require data to be presented to them in synchronism with this timing information. Synchronous modems, because of this extra clocking, are only capable of operating at certain preset baud rates. The receiving mode, which has an oscillator running at the same frequency as the transmitting modem, phase locks its clock to that of the transmitter and interprets changes of phase as data. The PolyMorphic bi-phase cassette interface operates in a synchronous mode.

In some cases it is desirable to operate in a hybrid mode which involves transmitting data with the asynchronous format using a synchronous modem. This occurs when an increase in operating speed is required without a change in the basic protocol of the system. This hybrid technique is known as isosynchronous and involves the generation of the start and stop bits associated with the asynchronous format, while still using the modem clock for

bit synchronization. The Byte standard cassette interface operates in an isosynchronous mode.

The 8251 USART (Universal Synchronous/Asynchronous Receiver) has been designed to meet a broad spectrum of requirements in the synchronous, asynchronous, and isosynchronous modes. In the synchronous modes it operates with 5, 6, 7, or 8-bit characters. Even or odd parity can be optionally appended and checked. Synchronization can be achieved internally via SYN character detection. SYN detection can be based on one or two characters which may or may not be the same. The single or double SYN characters are inserted into the data stream automatically if the software fails to supply data in time. The automatic generation of SYN characters is required to prevent the loss of synchronization. In the asynchronous mode the USART operates with the same data and parity structures as it does in the synchronous mode. In addition to appending a START bit to this data, it appends 1, 1½, or 2 STOP bits. Proper framing is checked by the receiver and a status flag set if an error occurs. In the asynchronous mode the USART can be programmed to accept clock rates of 1, 16, or 64 times the required baud rate. Note that X1 operation is only valid if the clocks of the receiver and transmitter are synchronized.

The USART can transmit the three formats in half or full duplex mode and is double-buffered internally (i.e., the software has a complete character time to respond to a service request). Although the USART supports basic data set control signals (e.g., DTR and RST), it does not fully support the signaling described in EIA RS-232-C. Examples of unsupported signals are Ring Indicator (CE), and the second channel signals. The serial option does not interface to the voltage levels required by EIA RS-232-C; this interface is provided by the SIO/2 card. (The SIO/2 also provides an optically isolated current loop interface).

A block diagram of the SER/8 serial port option is shown in Figure 2. As can be seen in the figure, the USART consists of five major sections which communicate with each other on an internal

data bus. The five sections are the receiver, transmitter, modem control, read/write control, and I/O Buffer. In order to facilitate discussion, the I/O Buffer has been shown broken down into its three major subsections: the status buffer, the transmit data/command buffer, and the receive data buffer.

RECEIVER

The receiver accepts serial data on the RxD pin and converts it to parallel data according to the appropriate format. When the USART is in the asynchronous mode, and it is ready to accept a character (i.e., it is not in the process of receiving a character), it looks for a low level on the RxD line. When it sees the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the RxD line is sampled again. If the line is still low, a valid START bit has probably been received and the USART proceeds to assemble the character. If the RxD line is high when it is sampled, then either a noise pulse has occured on the line or the receiver has become enabled in the middle of the transmission of a character. In either case the receiver aborts its operation and prepares itself to accept a new character. After the successful reception of a START bit the USART clocks in the data, parity, and STOP bits, and then transfers the data on the internal data bus to the receive data register. When operating with less than 8 bits, the characters are right-justified. The RxRDY signal is asserted to indicate that a character is available.

In the synchronous mode the receiver simply clocks in the specified number of data bits and transfers them to the receiver buffer register, setting RxRDY. Since the receiver blindly groups data bits into characters, there must be a means of synchronizing the receiver to the transmitter so that the proper character boundaries are maintained in the serial data stream. This synchronization is achieved in the HUNT mode.

In the HUNT mode the USART shifts in data on the RxD line one bit at a time. After each bit is received, the receiver register is

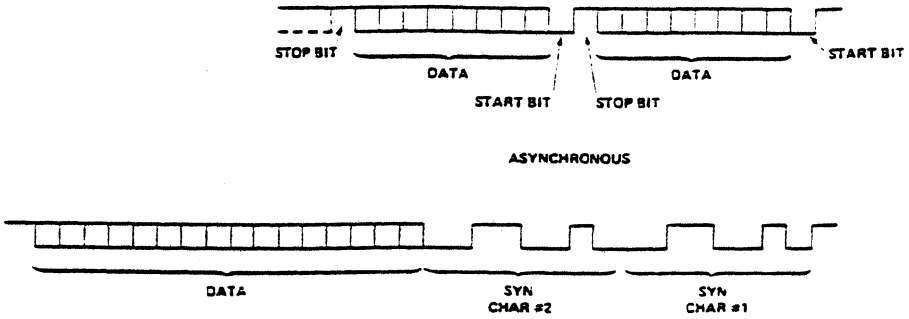


Figure 1. Transmission Formats

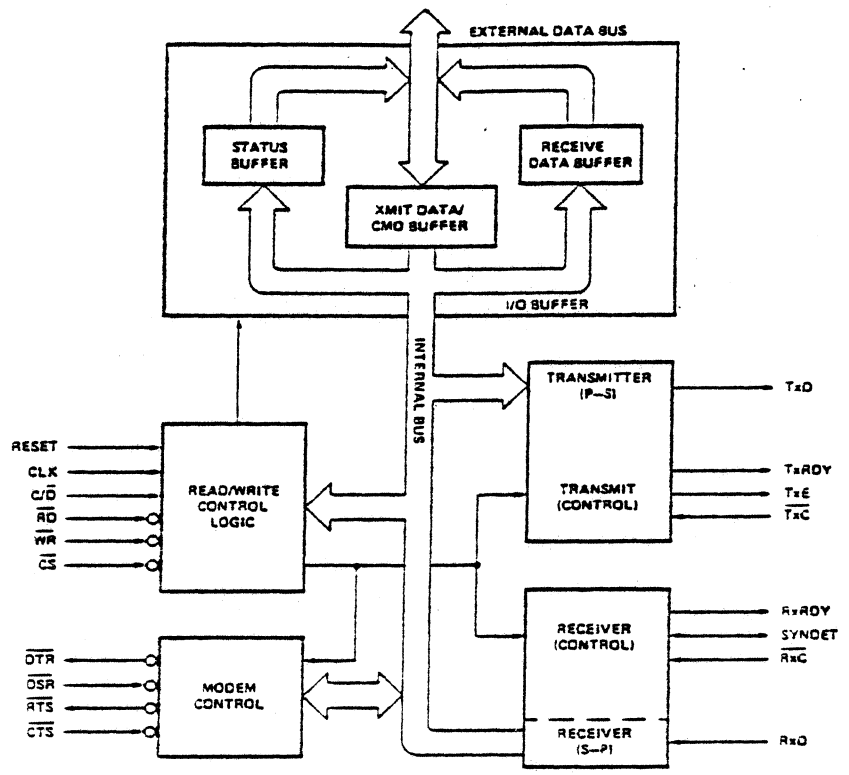


Figure 2. 8251 Block Diagram

CE	C/D	READ	WRITE	Function
0	0	0	1	CPU Reads Data from USART
0	1	0	1	CPU Reads Status from USART
0	0	1	0	CPU Writes Data to USART
0	1	1	0	CPU Writes Command to USART
1	X	X	X	USART Bus Floating (NO-OP)

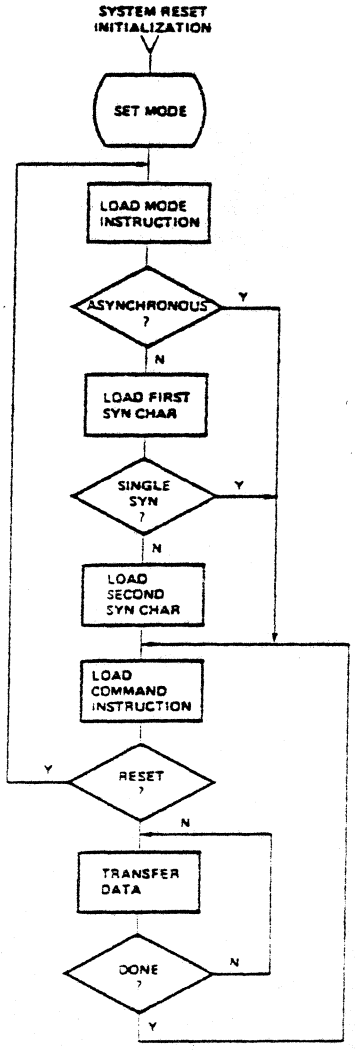


Figure 4. Initialization Flowchart

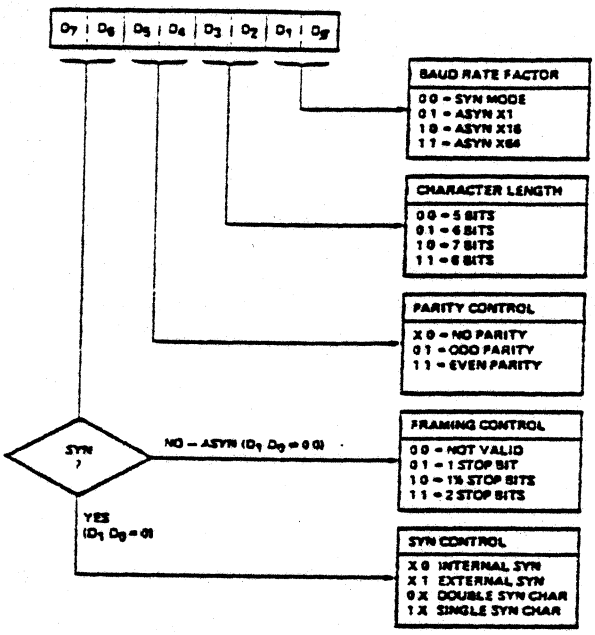


Figure 5. Mode Instruction Format

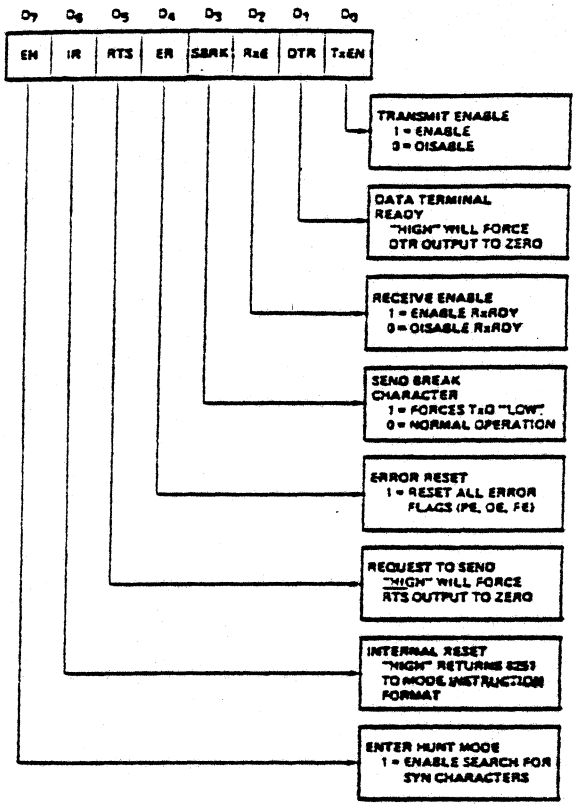


Figure 6. Command Instruction Format

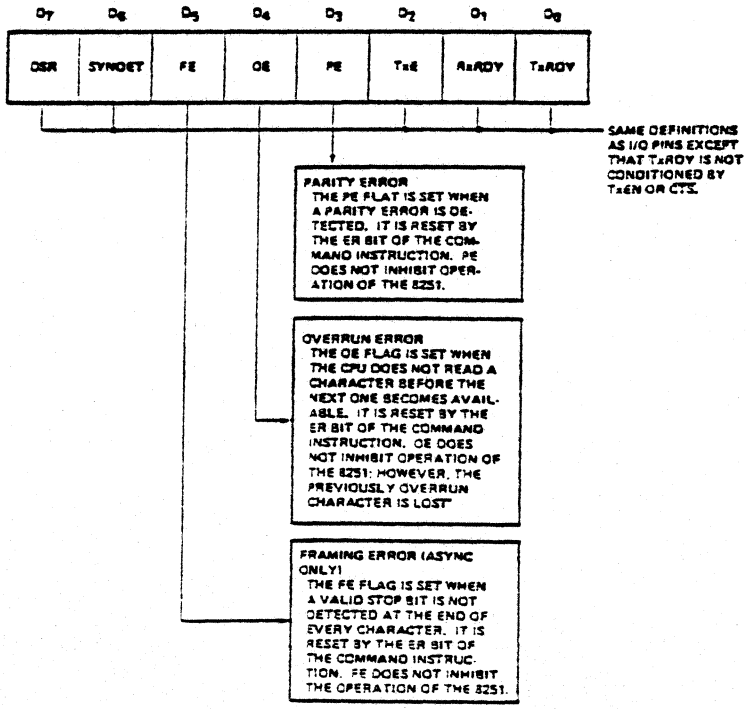


Figure 7. Status Register Format

compared to a register holding the SYN character (program loaded). If the two registers are now equal, the USART shifts in another bit and repeats the comparison. When the registers compare as equal, the USART ends the HUNT mode and raises the SYNDET line to indicate that it has achieved synchronization. If the USART has been programmed to operate with two SYN characters the process is as described above, except that two contiguous characters from the line must compare to the two stores SYN characters before synchronization is declared. Parity is not checked. The USART enters the HUNT mode when it is initialized into the synchronous mode or when it is commanded to do so by the command instruction. Before the receiver is operated, it must be enabled by the RxE bit (D_2) of the command instructions. If this bit is not set the receiver will not assert the RxRDY bit.

TRANSMITTER

The transmitter accepts parallel data from the processor, adds the appropriate framing information, serializes it, and transmits it on the TxD pin. In the asynchronous mode the transmitter always

adds a START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1½ or 2 STOP bits. In the synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter unless the computer fails to send a character to the USART. If the USART is ready to transmit a character and a new character has not been supplied by the computer, the USART will transmit a SYN character. This is necessary since synchronous communications, unlike asynchronous communications, does not allow gaps between characters. If the USART is operating in the dual SYN mode, both SYN characters will be transmitted before the message can be resumed. The USART will not generate SYN characters until the software has supplied at least one character; i.e., the USART will fill 'holes' in the transmission but will not initiate transmission itself. The SYN characters which are to be transmitted by the USART are specified by the software during the initialization procedure. In either the synchronous or asynchronous modes, transmission is inhibited until TxEnable and the $\overline{CTS}$ input are asserted.

An additional feature of the transmitter is the ability to transmit a BREAK. A BREAK is a period of continuous SPACE on the communication line and is used in full fuplex communication to interrupt the transmitting terminal. The USART will transmit a BREAK condition as long as bit 3 (SBRK) of the command register is set.

MODEM CONTROL

The modem control section provides for the generation of $\overline{RTS}$ and the reception of $\overline{CTS}$. In addition, a general purpose output and a general purpose input are provided. The output is labeled $\overline{DTR}$ and the input is labeled $\overline{DSR}$. $\overline{DTR}$ can be asserted by setting bit 2 of the command instruction; $\overline{DSR}$ can be sensed as bit 7 of the status register. Although the USART itself attaches no special significance to these signals, DTR (Data Terminal Ready) is normally assigned to the modem, indicating that the terminal is ready to communicate and DSR (Data Set Ready) is a signal from the modem indicating that it is ready for communications.

I/O CONTROL

The Read/Write Control Logic decodes control signals on the 8080 control bus into signals which gate data on and off the USART's internal bus and controls the external I/O bus (DB_0 - DB_7). The receiver and transmitter buffers are located at port #0 while the status and command buffers are port #1. The I/O buffer contains the STATUS buffer, the RECEIVE DATA buffer and the XMIT DATA/CMD buffer as shown in Figure 2. Note that although there are two registers which store data for transfer to the CPU (STATUS and RECEIVE DATA), there is only one register which stores data being transferred to the USART. The sharing of the input register for both transmit data and command makes it important to ensure that the USART does not have data stored in this register before sending a command to the device. The TxRDY signal can be monitored to accomplish this. Neither data nor commands should be transferred to the USART if TxRDY is low. Failure to perform this check can result in erroneous data being transmitted.

MODE SELECTION

The USART is capable of operating in a number of modes (e.g., synchronous or asynchronous). In order to keep the hardware as flexible as possible (both at the chip and end product levels), these operating modes are selected via a series of control outputs to the USART. These mode control outputs must occur between the time the USART is reset and the time it is utilized for data transfer. Since the USART needs this information to structure its internal logic it is essential to complete the initialization before any attempts are made at data transfer (including reading status).

A flowchart of the initialization process appears in Figure 4. The first operation which must occur following a reset is the loading of the mode control register. The mode control register is loaded by the first control output following a reset. The format of the mode control instruction is shown in Figure 5. The instruction can be considered as four 2-bit fields. The first

2-bit field (D_1D_0) determines whether the USART is to operate in the synchronous (00) or asynchronous mode. In the asynchronous mode this field also controls the clock scaling factor. As an example, if D_1 and D_0 are both ones, the $\overline{RxC}$ and $\overline{TxC}$ will be divided by 64 to establish the baud rate. The second field, D_3 - D_2, determines the number of data bits in the character and the third, D_5 - D_4, controls parity generation. Note that the parity bit (if enabled) is added to the data bits and is not considered as part of them when setting up the character length. As an example, standard ASCII transmission, which is seven bits plus even parity would be specified as:

X X 1 1 1 0 X X

The last field, D_7 - D_6, has two meanings, depending on whether operation is to be in the synchronous or asynchronous mode. For the asynchronous mode (i.e., $D_1D_0 \neq 00$), it controls the number of STOP bits to be transmitted with the character. Since the receiver will always operate with only one STOP bit, D_7 and D_6 only control the transmitter. In the synchronous mode, (D_1D_0=00), this field controls the synchronizing process. Note that the choice of single or double SYN characters is independent of the choice of internal or external synchronization. This is because even though the receiver may operate with external synchronization logic, the transmitter must still know whether to send one or two SYN characters should the CPU fail to supply a character in time.

Following the loading of the mode instruction the appropriate SYN character (or characters) must be loaded if synchronous mode has been specified. The SYN character(s) are loaded by the same control output instruction used to load the mode instruction. The USART determines from the mode instruction whether no, one, or two SYN characters are required and uses the control output to load SYN characters until the required number are loaded.

At completion of the load of SYN characters (or after the mode instruction in the asynchronous mode), a command character is issued to the USART. The command instruction controls the operation of the USART within the basic framework established by the mode instruction. The format of the command instruction is shown in Figure 6. Note that if, as an example, the USART is waiting for a SYN character load and instead is issued an internal reset command, it will accept the command as an SYN character instead of resetting. This situation, which should only occur if two independent programs control the USART, can be avoided by outputting three all zero characters as commands before issuing the internal reset command. The USART indicates its state in a status register which can be read under program control. The format of the status register read is shown in Figure 7.

When operating the receiver it is important to realize that RxE (bit 2 of the command instruction) only inhibits the assertion of RxRDY; it does not inhibit the actual reception of characters. Because the receiver is constantly running, it is possible for it to contain extraneous data when it is enabled. To avoid problems this data should be read from the USART and discarded. The read should be done immediately following the setting of Receive Enable in the asynchronous mode, and following the setting of Enter Hunt in the synchronous mode. It is not necessary to wait for RxRDY before executing the dummy read.

USART INTERRUPTS

The SYNDET, TxRDY, and RxRDY flags will cause an interrupt if they are set. These three flags are ored together and applied to interrupt 3. This connection may be broken, if this interrupt is not used, by cutting trace "K" on the CPU card. Two adjacent pads are provided to reconnect the interrupt later, if desired.

When using the PolyMorphic Systems Monitor ROM the K jumper should be cut for monitor versions 2.Ø and 2.2 and connected for version 3.Ø.

USART ADDRESSING

The transmit and receive buffers are addressed as port Ø on the CPU card if it is set up for operation at Ø. The command and status buffers are located at port 1. If your CPU card is not setup for operation at Ø consult the following table.

SER/8 ADDRESSING

Address Jumper	ROM begins at	Data port	Command & Status Port	Baud Rate Generator Latch
J (factory set)	ØØØØ	Ø	1	4
T	8ØØØ	8Ø	81	84
S	EØØØ	EØ	E1	E4

BAUD RATE GENERATOR OPERATION

The baud rate generator may be accessed through port #4. The byte output to this port will be latched into the baud rate latch and determines the baud rate, device number, and ROM disable or enable. When power is applied to the CPU card or the front panel reset button is pushed, this latch is set to Ø. The command format is as follows:

D_7 D_6	D_5	D_4	D_3	$D_Ø$
Unused	ROM disable	Device #	Baud Rate	

BAUD RATE

The baud rate field may assume 1 of 16 values of Ø through F (base 16). 15 of these are valid baud rates, Ø disables clock generation. Note that the actual baud rate is determined by the USART mode (x1, x16, x64 clock). See Figure 8 for the available baud rates.

SER/8 Baud Rates

Baud Rate Field Binary	Hex	USART mode x1	x16	x64
0001	1	800	50	12.5
0010	2	1200	75	18.75
0011	3	1760	110	27.5
0100	4	2152	134.5	33.62
0101	5	2400	150	37.5
0110	6	4800	300	75
0111	7	9600	600	150
1000	8	14400	900	225
1001	9	19200	1200	300
1010	A	28800	1800	450
1011	B	38400	2400	600
1100	C	57600	3600	900
1101	D	----	4800	1200
1110	E	----	7200	1800
1111	F	----	9600	2400

Figure 8.

DEVICE NUMBER

This bit selects the device to be connected to the USART. Two devices (Ø and 1) may share the USART on the CPU board. When a device is enable it sends data, receive clock, CTS and DSR to the USART through a tri-state buffer. Transmit data, clock, RTS and DTR are anded with the device select signal at the device. Device Ø is normally a cassette interface minicard and device 1 is normally a RS-232/current loop minicard. (Note that the 2 DIP sockets on the CPU do not determine the device number).

ROM DISABLE

Bit 5 of the BRG control word normally is not enabled. When enabled by connecting a jumper on the CPU card, it will disable the onboard ROM and RAM when true. This option should be used with caution as disabling the onboard ROM while executing the monitor routines may cause unpredictable results. (See application note on ROM disable option use).

APPENDIX A
PROGRAMMING HINTS

1. Output of a command to the USART destroys the integrity of a transmission in progress if timed incorrectly.

   Sending a command into the USART wll overwrite any character which is stored in the buffer waiting for transfer to the parallel-to-serial converter in the device. This can be avoided by sending a command if transmission is taking place. Due to the internal structure of the USART, it is also possible to disturb the transmission if a command is sent while SYN character is being generated by the device. (The USART generates a SYN if the software fails to respond to TxRDY). If this occurrence is possible in a system, commands should be transferred only when a positive-going edge is detected on the TxRDY line).

2. RxE only acts as a mask to RxRDY; it does not control the operation of the receiver.

   When the receiver is enabled, it is possible for it to already contain one or two characters. These characters should be read and discarded when the RxE bit is first set. Because of these extraneous characters the proper sequence for gaining synchronization is as follows:

   1. Disable interrupts.

   2. Issue a command to enter hunt mode, clear errors, and enable the receiver (EH,ER,RxE=1).

   3. Read USART data (it is not necessary to check status).

   4. Enable interrupts.

The first RxRDY that occurs after the above sequence will indicate that the SYN character or characters have been detected and the

next character has been assembled and isready to be read.

3. Loss of CTS or dropping TxEnable will immediately clamp the serial output line.

   TxEnable and RTS should remain asserted until the transmission is complete. Note that this implies that not only has the USART completed the transfer of all bits of the last character, but also that they have cleared the modem. A delay of 1 msec following a proper occurence of TxEmpty is usually sufficient (see Item 4). An additional problem can occur in the synchronous mode because the loss of TxEnable clamps the data in at a SPACE instead of the normal MARK. This problem, which does not occur in the asynchronous mode, can be corrected by an external gate combining RTS and the serial output data.

4. Extraneous transitions can occur on TxEmpty while data (including USART generated SYNs) is transferred to the parallel-to-serial converter,

   This situation can be avoided by ensuring that TxEmpty occurs during several consecutive status reads before assuming that the transmitter is truly in the empty state.

5. A BREAK (i.e., long space) detected by the receiver results in a string of characters which have framing errors.

   If reception is to be continued after a BREAK, care must be taken to ensure that valid data is being received; special care must be taken with the last character perceived during a BREAK, since its value, including any framing error associated with it, is indeterminate.

Interfacing to the POLY-88 Bus

Data Bus - Data transfer in the POLY-88 occurs over two unidirectional data busses - the Data In bus, DIØ-7 (device to CPU transfer) and the Data Out bus, DØ1-7 (CPU to device transfer). Both busses are 8 bits wide. The Data Out bus is tri-state and will drive 30 TTL loads or 120 low-power schottky TTL (LSTTL) loads. The Data In bus present 1 LSTTL load. D7 is the most significant bit during arithmetic operations. Data is presented on the bus uncomplemented.

Address Bus - The Address Bus, AØ - A15, is 16 bits wide and is also tri-state for DMA or multiprocessing applications. It also will drive 30 TTL or 120 LSTTL loads. During memory transfers a 16 bit memory address appears on this bus. A15 is the most significant address bit. During I/O transfers two identical 8 bit addresses appear on the bus. A15 and A7 are the most significant address bits. Generally the lower 8-bits (AØ - 7) are used to address I/O devices.

Interrupt Bus - The Interrupt Bus (VIØ - VI7) consists of eight active low interrupt request lines. VI7 is the interrupt with the highest priority and VIØ has the lowest priority. Each of these lines present 1 TTL load to the bus and each is pulled up by a 2200 ohm resistor to +5 volts to ensure unused interrupts are inactive. An interrupt request should remain low until services by the CPU.

Status Bus - These 6 lines indicate the status of the CPU (i.e.- what type of cycles it is running). SINTA goes high during and interrupt acknowledge cycle; SMEMR, when high, signals a memory read cycle; SIN and SOUT, respectively, indicate input and output data transfers. SWO goes low during output or memory write cycles, and HLTA is active (high) only when the CPU is halted. Note that there is not a separate status

line for a memory write cycle. This may be generated by the logic SWO·$\overline{SOUT}$. All status lines, except HLTA, are tri-state and capable of driving 30 TTL loads. HLTA is an open collector output with a resistive pulley of 1000Ω to +5 volts.

Clocks - The two 8080 CPU clocks Ø1 and Ø2 are buffered and brought out to the bus. Both clocks are active high.

Control Bus and Data Strobes - Two lines are provided for cold starting the CPU. The PRESET line (active low) causes the program counter to be reset to zero, and when release (high) starts execution from that point with interrupts disabled. PRESET may be generated by a mechanical switch closure as an RC timing network and pullups are provided on the CPU/8 cards. POC is an active low signal that is asserted whenever PRESET is active. It is open-collector with a 2.2K pullup to +5 volts and is normally used to reset peripheral controllers when power is applied to the system or during a reset. XRDY, PRDY, and BGNT are all anded together and applied to the 8080 CPU as the READY signal. When any of these are low the processor will enter a wait state the next time it tries to access the bus and will remain in a wait state until XRDY PRDY, and BGNT are all high. These three inputs are pulled high through 2200Ω resistors to +5 volts to ensure they are active when unused. The PRDY line is to be used by memory and peripheral controllers to indicate that a transfer cannot be completed within the 500 nanosecond CPU cycle, and will extend this cycle by adding wait cycles, until PRDY is high.

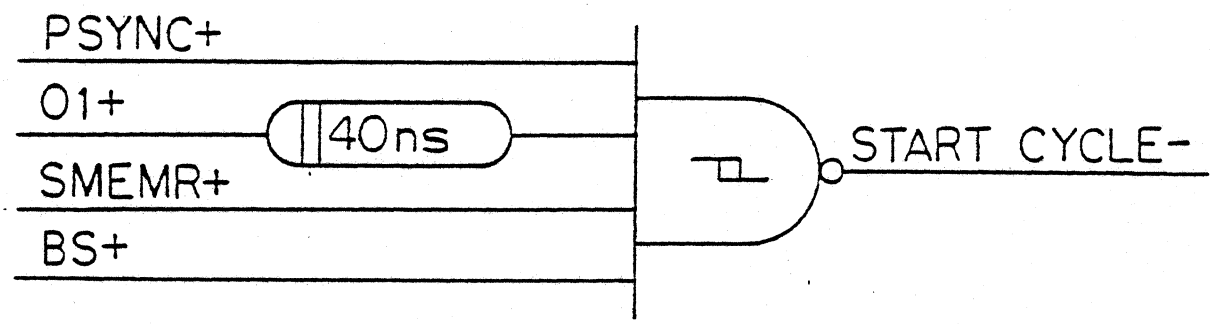
BGNT and XRDY are to be used only by the bus controller card. BGNT, when taken low, will also disable all tri-state bus drivers on the CPU/8 card. The data strobes PDBIN and PWR are processor data bus in and processor write, respectively. PDBIN is active high and PWR is active low. When PDBIN is active data for a transfer to the CPU should be gated onto the data in bus. When PWR is active, valid data is present on the data out bus for a transfer to a peripheral device on memory. The PSYNC

signal is asserted (active high) at the beginning of every memory or I/O cycle. During this time, the processor status word is being sent to the status latch. Shortly after the rising edge of Ø1 during PSYNC the status is available. A delay of 40 ns after the positive edge of Ø1 is satisfactory. The memory or I/O address (AØ - A15) is stable shortly after the rising edge of PSYNC (see timing diagram). An additional strobe, PMWR or MWRITE, is provided on the bus. This is an active high signal that is the logical end of PWR+ and SOUT-. It is to be used as the write strobe for memory transfers.

Using Static Devices on the POLY-88

Bus -- Devices such as static ROM or RAM or I/O parts are easy to interface to the Poly-88 bus. Addresses and chip selects may be decoded directly from the address bus and applied to the memory or I/O devices. PDBIN and the status of the operation (SINP or SMEMR) should be used to gate data onto the bus. When writing MWRITE or PWR·SOUT should be used as the write strobe. If data is to be gated onto an oncard bus, $\overline{\text{SYNC}}$·WO should be used. Otherwise the DO bus should be buffered and applied continuously. When designing an interface to the bus all data strobes should be applied to Schmidt-trigger inputs for noise immunity. See Figure X for an example.

Additional interfacing for dynamic or clocked devices - When interfacing to dynamic RAM a signal is needed to indicate the start if an access cycle. This may be produced by anding together Ø1, SYNC, the appropriate status bit and a board select (BS) signal. BS indicates that a valid address has been decoded. Shown below is a circuit for decoding the start of a memory read cycle.



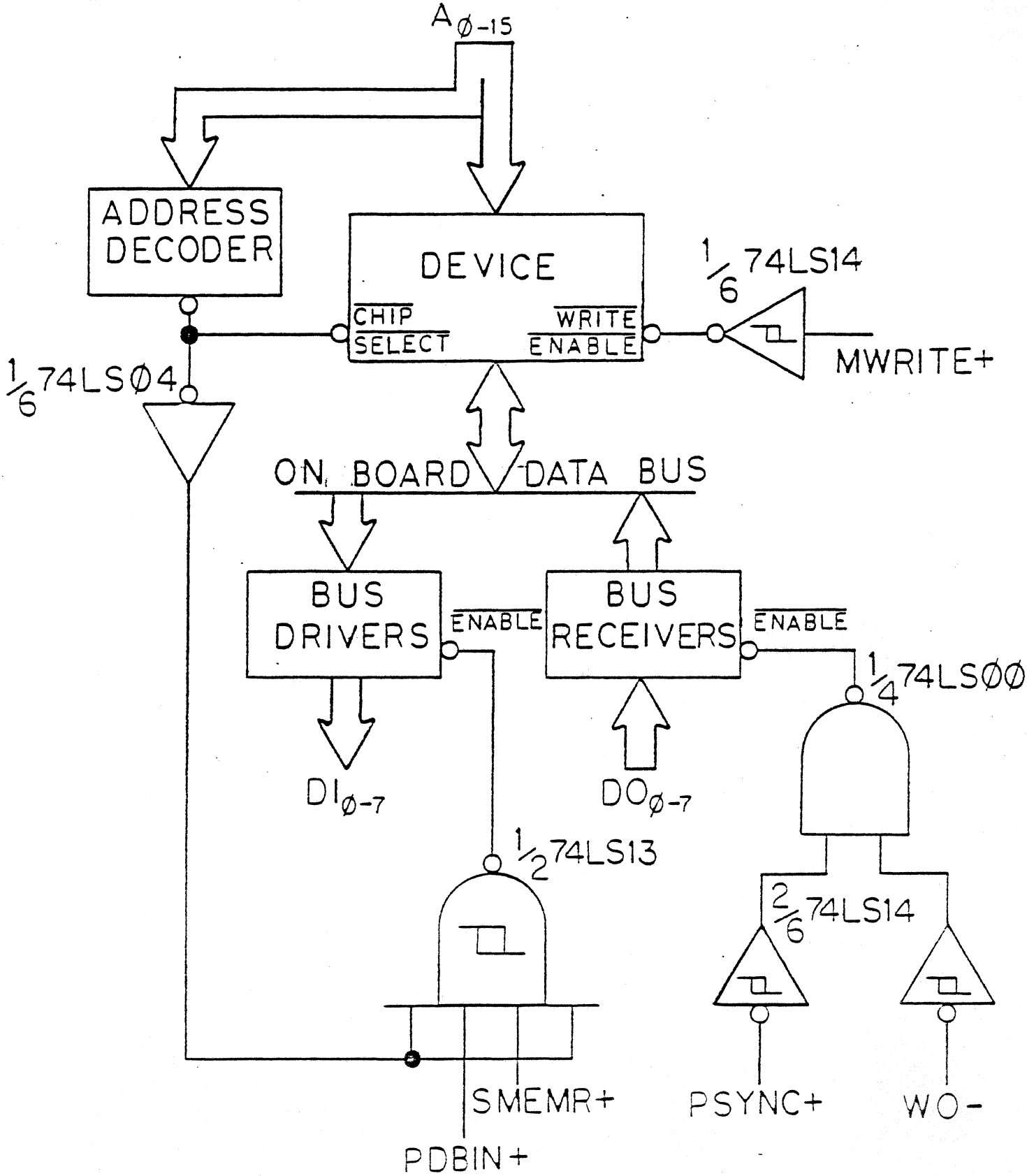
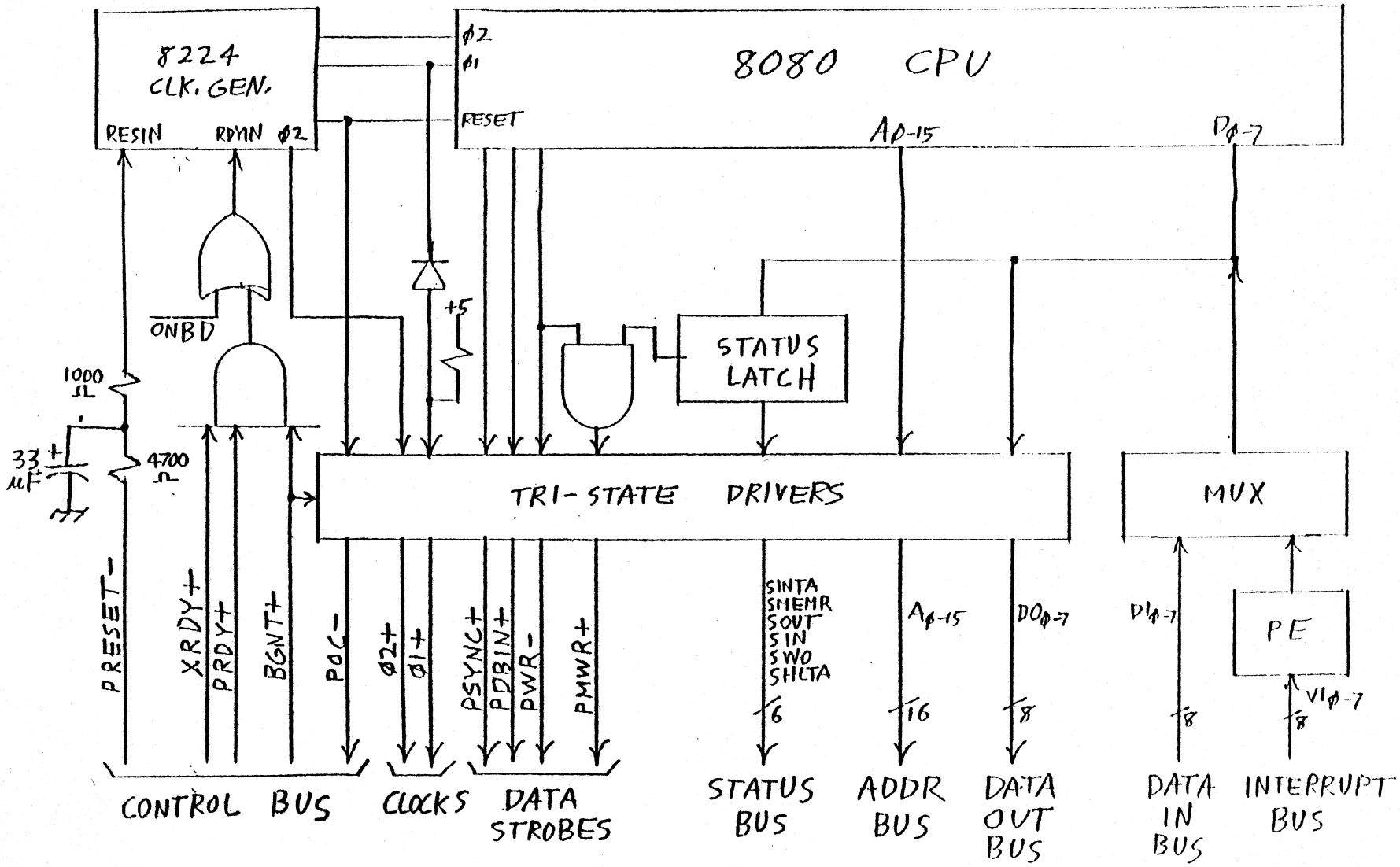

A Ø-15
ADDRESS DECODER
DEVICE
CHIP SELECT
WRITE ENABLE
1/6 74LS14
MWRITE+
1/6 74LSØ4
ON BOARD DATA BUS
BUS DRIVERS
ENABLE
BUS RECEIVERS
ENABLE
1/4 74LSØØ
DI Ø-7
DO Ø-7
1/2 74LS13
2/6 74LS14
SMEMR+
PSYNC+
WO-
PDBIN+

FIG X

CPU/8 BUS INTERFACE



J. D. A. 10-19-76