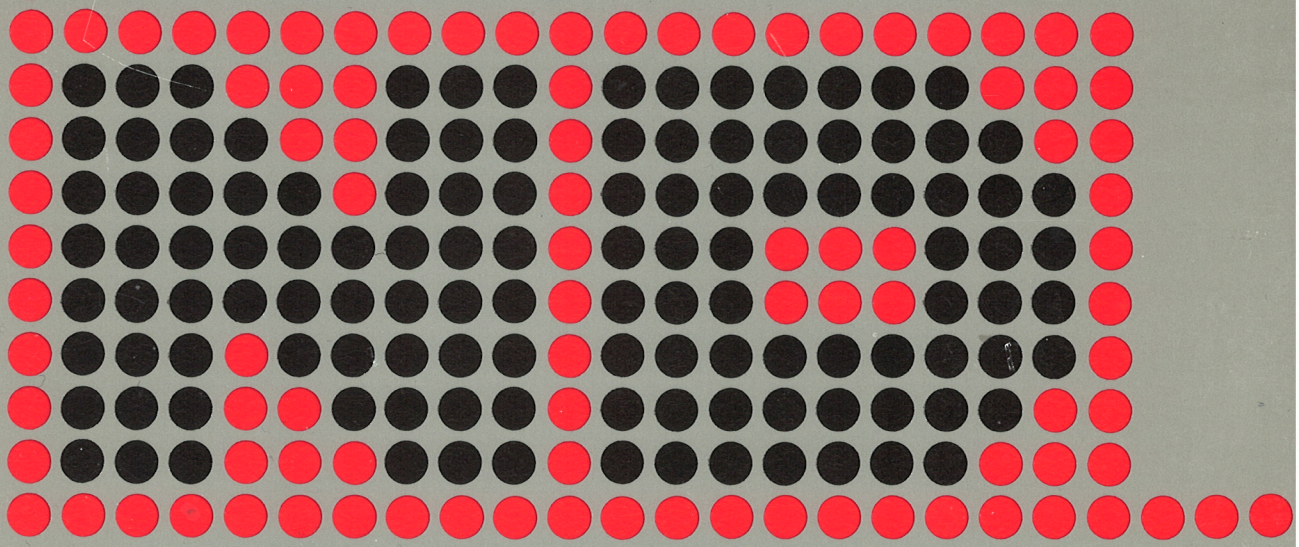
ND-F1-J00

Norsk Data

PIOC Hardware Reference Manual

ND-02.003.01



PIOC Hardware Reference Manual

ND-02.003.01

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

PRINTING RECORD

Printing	Notes
12/82	Version 01

PIOC HARDWARE REFERENCE MANUAL
Publ.No. ND-02.003.01

NORSK DATA A.S
P.O. Box 4, Lindeberg gård
Oslo 10, Norway

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

The Product

PIOC, the Programmable Input Output Controller is an ND-100 interface card capable of handling 4 full duplex serial communication lines. The hardware is built around an MC 68000 microprocessor and 128 Kbytes of Random Access Memory (RAM) with error correction logic.

The Reader

The manual is intended for hardware engineers and other personnel performing maintenance on ND products.

Prerequisite Knowledge

Basic knowledge of the ND-100 computer system is recommended. This knowledge can be obtained by attending NORSK DATA courses and by studying NORSK DATA manuals.

The Manual

The manual contains 6 chapters and 4 appendices. Chapter 1 contains an introduction and gives some specifications. Chapter 2 explains the PIOC ND-100 communication and chapter 3 the Local Interrupt. Chapter 4 describes the memory and chapter 5 the Local I O. Chapter 6 contains the logic diagrams.

Appendix A describes the interrupt controller, appendix B the DMA controllers, appendix C the timing controller and appendix D the serial controllers.

NOTE:

The figure numbers in the appendices are applicable to the particular appendix.

The information in the appendices is taken from AM9500 Peripheral Products Interface Guide issued by Advanced Micro Devices and from the SIO manual issued by Zilog (UK) Limited.
Z80 is a Zilog trademark.

TABLE OF CONTENTS

A P P E N D I X

1 INTRODUCTION

1.1 General

The Programmable Input Output Controller - PIOC - is designed and built around an MC 68000 microprocessor and 128 Kbytes of Random Access Memory (RAM).
The purpose of the PIOC is to relieve the ND-100 of the low level handling of the serial controllers thus giving better performance.
The PIOC consists of 1 ND-100 interface card capable of handling 4 full duplex serial communication lines. All serial channels may exchange data simultaneously with local RAM via Direct Memory Access (DMA). The RAM is also accessible directly from the ND-100 as an ordinary RAM module. The microprocessor may also be programmed for complete communication protocols such as the CCITT X.25.

1.2 Specifications

Number of serial lines	4 full duplex
Local Processor	MC 68000
Local Memory	128 Kbyte EDAC DRAM 8 Kbyte EPROM
Clock Frequency	8 MHz
Electrical Levels	RS 232C (V-24/28) RS 422 (V11/X.27)
Serial Modes	Asynchronous Binary Synchronous HDLC and SDLC
Serial Bit Rate	800 Kbit/s maximum depending on local processing

1.3 Local Processor

The local PIOC processor is a standard MC 68000 16 bit microprocessor running at 8 MHz.
The basic control signals of the microprocessor, HALT and RESET, are directly set by the ND-100. They will also be set when the power goes down. When the HALT and RESET signals are switched off, the processor will fetch the system stack pointer and the restart address from the first 8 bytes of EPROM memory.
The MC 68000 is described in detail in the "MC 68000 16 bit microprocessor users guide" by Motorola.

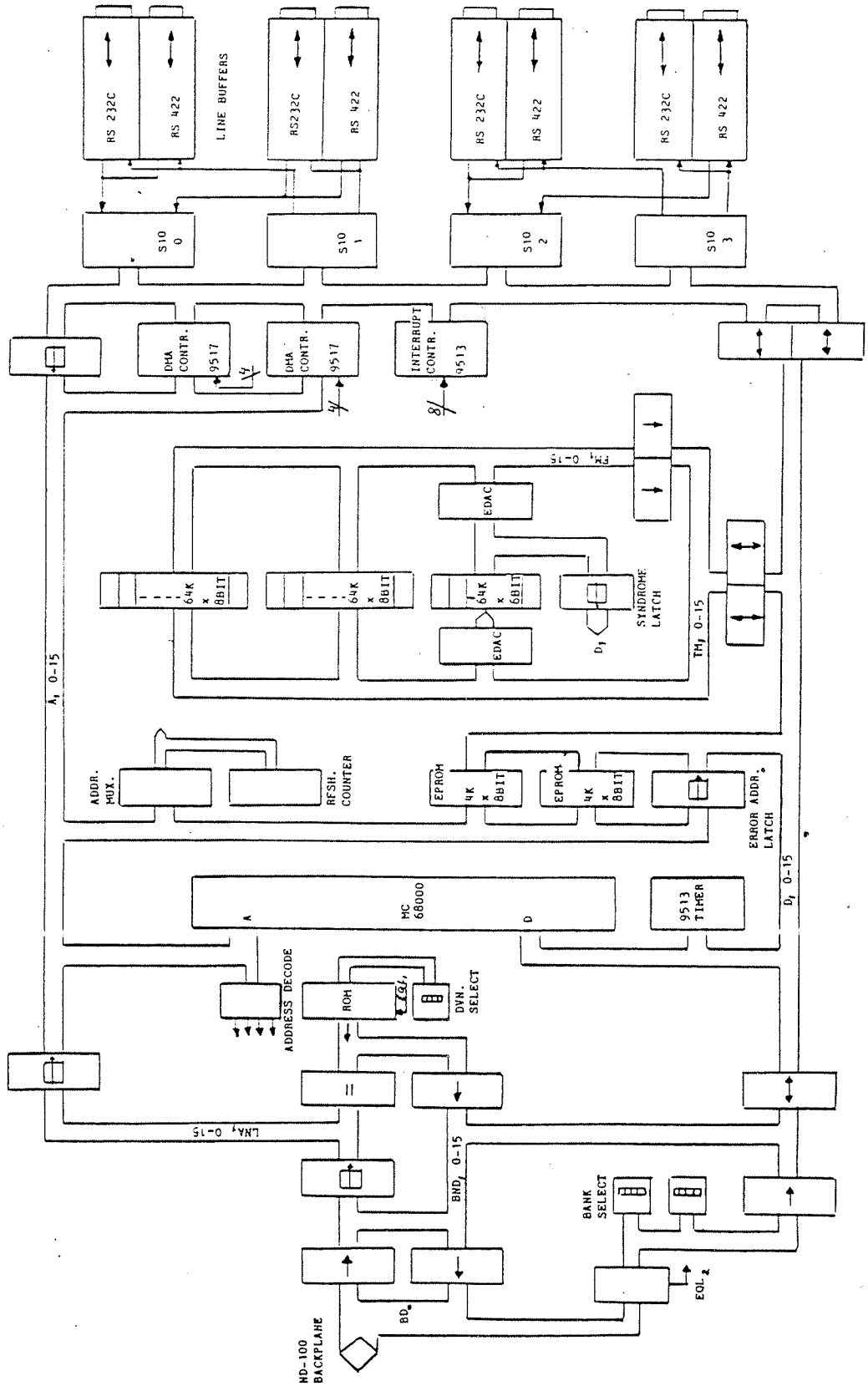


Fig. 1.1. PIOC Block Diagram

2 PIOC ND-100 COMMUNICATION

The 128 Kbytes of the PIOC RAM is accessible from the ND-100 as any other memory bank.
The position of the PIOC RAM in the address space of the ND-100 (Bank Number) may be changed in steps of 128 Kbytes with two thumbwheel switches on the PIOC.
The Bank Number can be read back to the ND-100 via an IOX instruction.

The common memory allows a simple and effective communication between the PIOC and the ND-100. Messages can be transferred via special mailbox areas in RAM.

This requires a synchronization mechanism in software and some kind of attention line in each direction between the processors.
In PIOC, the attention line to ND-100 is an interrupt, (level 12) which is set by any access from the PIOC CPU in the SCIP address (see table 5.6).
In the opposite direction, ND-100 can send two different interrupts to PIOC by setting bits 2 and 7 in the PIOC control word. There is also a very high level interrupt from ND-100 to PIOC intended for situations when normal PIOC/ND-100 communication cannot be used. Each PIOC has a control word which is set directly by ND-100 (IOX PIOC + 3) and a status word that can be read back (IOX PIOC + 2).

2.1 PIOC Control Word

IOX PIOC +3 % Write Control Register

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BIT 0 = 1 - ENABLE FOR SCIP INTERRUPT
BIT 2 = 1 - SET ND CALLING INTERRUPT
BIT 3 = 1 - START OPCOM
BIT 4 = 1 - RESET
BIT 5 = 1 - HALT
BIT 6 = 1 - POWER LOW
BIT 7 = 1 - SET ND READY INTERRUPT
BIT 8 = 1 - DISABLE CHECK BIT WRITE

Table 2.1 PIOC Control WORD

A Master Clear pulse from the ND-100 (or power on) will set the local RESET and HALT signals in PIOC and local I/O will be reset.
To start opration, the MC 68000 requires an IOX PIOC + 3 command with RESET and HALT off.

2.2 PIOC Status Word

IOX PIOC + 2 % Read Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BIT 0 = 1 - STATUS CHANGE IN PIOC INTERRUPT ENABLED
BIT 2 = 1 - STATUS CHANGE IN PIOC (SCIP)
BIT 4 = 1 - PIOC RESET ON
BIT 5 = 1 - PIOC HALTED
BIT 8-15 READS BACK BANK NUMBER OF THE COMMON MEMORY

Table 2.2 PIOC Status Word

2.2.1 Switch Settings

There are 3 thumbwheels on the PIOC. One of these (position 12 J) is for selection of PIOC number, with corresponding device number and identification number and the remaining two thumbwheels will select which Bank Number the PIOC occupies in the address range of ND-100.

PIOC number	Device number (Octal)	Ident code (Octal)
0	140020	140002
1	140024	140003
2	140030	140004
3	140034	140005
4	140040	140006
5	140044	140007

Table 2.3 Standard PIOC Device Number

NOTE: PIOC number equals setting of thumbwheel 12J

Bank number is selected with the thumbwheels in position 7J and 9J with 7J being most significant. Resulting Bank Number is:

7J setting * 16 + 9J setting = Bank Number
The lowest address in PIOC seen from ND-100 will be:
7J setting * 2048K + 9J setting * 128K = Lower PIOC address (bytes)
If 7J is set to 2 and 9J is set to 3 then:
Bank Number = 2 * 16 + 3 = 35
Lower PIOC address = 2 * 2048 + 3 * 128 = 4480 Kbytes

3 LOCAL INTERRUPT

The basic interrupt facilities of the local MC 68000 processor with 8 basic levels, are expanded with 8 sub-levels on level 3.

The expansion is performed by an AM 9519 interrupt controller which is fully described in appendix A.
Each PIOC interrupt is identified by a unique vector.

(Highest) Level	7:	POWER LOW
"	6:	OPCOM
"	5:	MEMORY ERROR
"	4:	SERIAL INPUT
"	3.7:	INPUT DMA ERROR
"	3.6:	OUTPUT DMA FINISHED
"	3.5:	REAL TIME CLOCK
"	3.4:	ND-100 CALLING
"	3.3:	ND-100 READY
"	3.2:	SINGLE MEMORY ERROR
"	3.1:	COMMUNICATION CONTROLLER RESET
"	3.0:	WRITE PROTECT VIOLATION
"	2:	SERIAL OUTPUT
"	1:	SPARE
"	0:	SPARE

Table 3.1. PIOC Interrupt Levels

Level 7: The Power Low interrupt is generated on the leading edge of the ND-100 Master Clear signal, which may be caused by ND-100 program, operators push-button or by a detected power failure.
Internal PIOC Reset is delayed 50 microseconds after Master Clear, enabling the MC 68000 registers to be saved in memory.

Level 6: OPCOM interrupt is directly set by bit 3 in the control IOX word from ND-100 to PIOC.
It must be reset by reading the PIOC OPCL address (see table 5.6).
OPCOM interrupts are intended for actions with higher priority than the normal PIOC/ND-100 communication.

Level 5: The Memory Error interrupt is always generated when a double memory error is detected and on single memory errors when error logging is enabled.

Level 4: Serial Input interrupts are generated directly by the input Serial Line controllers. The two input controllers (SIO 0 and 2) are daisy chained. The devices also have an internal daisy chain giving the following prioryty on interrupt level 4:

Highest:	SIO 0 input A	PIOC input channel 0
	SIO 0 output A	(normally not used)
	SIO 0 ext/status A	(status signals)
	SIO 0 input B	PIOC input channel 1
	SIO 0 output B	(normally not used)
	SIO 0 ext/status B	(statuus signals)
	SIO 2 input A	PIOC input channel 2
	SIO 2 output A	(normally not used)
	SIO 2 ext/status A	(status signals)
	SIO 2 input B	PIOC input channel 3
	SIO 2 output B	(normally not used)
	SIO 2 ext/status B	(status signals)

Level 3.7: The Input DMA Error interrupt is set when the word counter advances from 0 to -1, indicating that the allocated memory space is taken.

Level 3.5: RTC interrupts are generated by the timing controller. The frequency is software selectable.

Level 3.4: ND-100 calling interrupt is set directly by bit 2 in the ND-100 to PIOC control word IOX. It is intended as the "wake-up" signal in normal ND-100 to PIOC communication.

Level 3.3: ND-100 ready interrupt is set directly by bit 7 in the ND-100 to PIOC control word IOX.

Level 3.2: Single memory error is set by all corrected (single) errors, even if error logging is not enabled.

Level 3.1: The Communication Lines Reset interrupt is set when the line controllers are reset by the common watch-dog or when they are reset by ND-100 or MC 68000.

Level 3.0: The Write Protect Violation interrupt is set when MC 68000 user or input DMA attempts writing in a protected area. An attempt by MC 68000 will also lead to bus error, while an attempt by a DMA controller will set interrupt level 3.7.

Level 2: Serial Output interrupts are generated directly by the output serial line controllers. The two output controllers (SIO 1 and 3) are daisy chained. There is also an internal interrupt daisy chain giving a total interrupt chain on level 2 as follows:

Highest:	SIO 1	input A	(normally not used)
	SIO 1	output A	(PIOC output channel 0)
	SIO 1	ext/status	(status signals)
	SIO 1	input B	(normally not used)
	SIO 1	output B	(PIOC output channel 0)
	SIO 1	ext/status B	(status signals)
	SIO 3	input A	(normally not used)
	SIO 3	output A	(PIOC output channel 2)
	SIO 3	ext/status A	(status signals)
	SIO 3	input B	(normally not used)
	SIO 3	output B	(PIOC output channel 3)
	SIO 3	ext/status B	(status signals)

4 MEMORY

The PIOC memory consists of 4K x 16 bits EPROM with an access time of 450 nanoseconds and 64K x 22 bits error correcting dynamic RAM.

Because of the error correction, 1 wait state (125ns) is inserted in memory read cycles and 2 wait states are inserted in memory write cycles.
The EPROM is only accessible from MC 68000, while the RAM can be accessed from ND-100 and DMA as well. Access priority is shown in table 4.1.

(Highest)	1	ND-100
	2	OUTPUT DMA
	3	INPUT DMA
(Lowest)	4	MC 68000

Table 4.1 Priority of PIOC Bus Accesses.

Address:		(Bytes)	Use:	Access from:
0	-	64K	RAM	ND-100, MC 68000*
64K	-	128K	RAM	ND-100, MC 68000*; DMA**
128K	-	15296K	Spare	
15296K	-	15298K	Local I/O	MC 68000
15328K	-	15336K	ROM	MC 68000
15360K	-	15488K	Protect Table	MC 68000 supervisor
15488K	-	16256K	Reserved	
16256K	-	16384K	(RAM Image)	

Table 4.2 PIOC Address Space Utilisation

* : User write access depends on protect table.
** : Input DMA access depends on protect table.

The 128 Kbytes of RAM are also accessible in the upper part of the address space from 16256K to 16384K.
ND-100 and DMA accesses actually makes use of these addresses.

4.1 Error Detection and Correction

A data word read from RAM will always go through a correction cycle. Thus, a single bit error will not be noticed by the processor but the data word in.RAM will still be wrong.

If error logging is enabled, the syndrome code and failing address will be latched and interrupt is set. MC 68000 can then read the Error Address and Syndrome registers, log the error and correct it.

2-bit errors will always be latched and interrupt will be set even if error logging is not enabled. Software has to decide if processing has to stop or not in the case of double errors.

When one error has been latched the syndrome code has to be read before another error can be latched.

If a double error is detected during an ND-100 access in the common memory, the signal MERR will be given back to the ND-100 together with databits 16 and 17. This indicates parity errors in both bytes. Single errors will not be noticed by the ND-100 but may be logged by MC 68000.

Together with the 6 syndrome bits, 2 bits will be latched indicating that the ND-100 or DMA performed the failing access. If none of these bits are set, the access is done by the local processor.

Testing of the error correction logic is possible from the ND-100 using bit 8 of the PIOC control word (Disable Check Bit Write). A test pattern is first written in memory with control bit 8 = 0 as usual. Control bit 8 is then set to 1 and another testpattern is written over the first one. The memory will now contain the data bits of the last test pattern and the check bits of the first test pattern.
In this way different memory errors may be simulated.

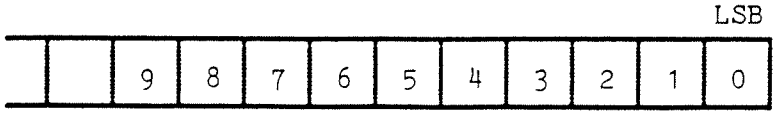


Fig. 4.1. Syndrome Register (Syren)

Bit 7	Bit 6	Error Access by:
0	0	ND-100
1	0	DMA
1	1	MC 68000

Table 4.3. Error Access Codes

Syndrome Code Bits	Failing Bit Number
110100	0
110010	1
110001	2
101100	3
101010	4
101001	5
100101	6
100011	7
011100	8
011010	9
010110	10
010101	11
010011	12
001110	13
001101	14
001011	15
111110	16 (= CB 0)
111101	17 (= CB 1)
111011	18 (= CB 2)
110111	19 (= CB 3)
101111	20 (= CB 4)
011111	21 (= CB 5)
111111	NO ERROR

Table 4.4. Error Syndrome Table

4.2 Write Protection

The RAM memory has a protection system that may protect against writing from input DMA and from MC 68000 user. The protection is divided into segments of 128 bytes. Write access to a segment is obtained by writing 001 in a word with address in the segment concerned + 15360K. Writing 0 in the word will protect from writing. Attempted write in a protected area will lead to time-out of the bus cycle, BUS ERROR and interrupt 3.0.

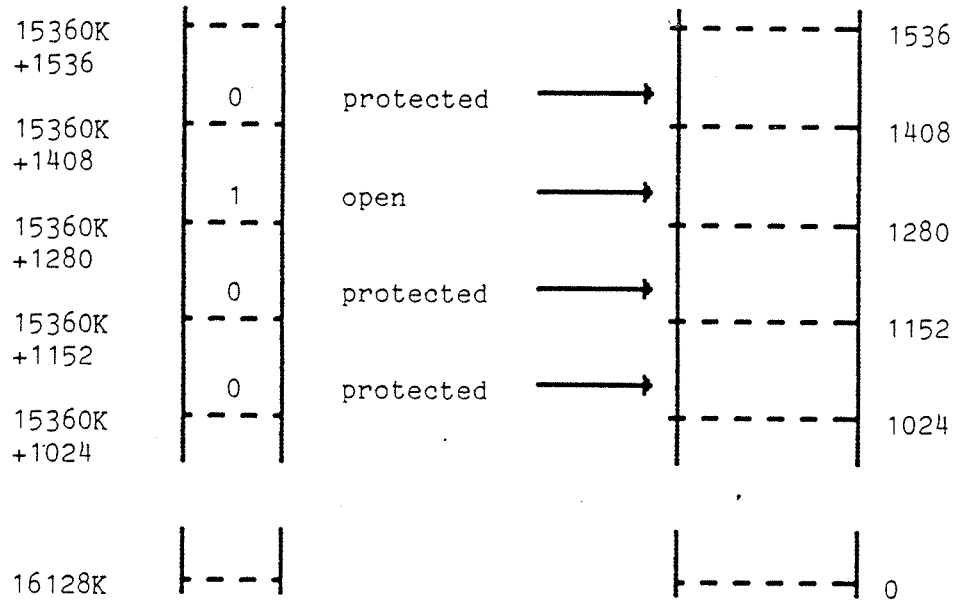


Fig. 4.2. Correspondence between Protect Table and RAM

NOTE: Write protect is set from the supervisor routine. Inquiries to the protect table are ignored in user routines with access to areas otherwise protected. This is used to give the user routine access to the I/O system and all of the memory.

4.3 Power Failure

The refresh system and the memory array itself are connected to the Stand-by power supply so that memory contents will be retained during short periods of power failure. When loss of mains power has been detected by the ND-100, the CPU will go through a Power Fail sequence and activate the Master Clear. In PIOC this leads to a Power Low interrupt and local PIOC Reset after a short delay (50 microseconds). During Power Low interrupt service routine, all registers in MC 68000 are saved.

5 LOCAL I/O

5.1 Internal DMA

In order to releave the PIOC CPU of data transport between local RAM and serial controllers, each of the 4 serial lines has a dedicated DMA channel for incoming data and another for output.
The DMA system is based on 2 AM9517 4-channel DMA controllers (described in detail in appendix B). For priority reasons one of the controllers contains all the input DMA channels while the other is for output only.

Each DMA transfer moves 1 byte of data and occupies the PIOC bus for approximately 1,6 microseconds. Maximum bytecount and address range is 64 Kbytes. DMA can only access the upper half of the PIOC RAM.

5.2 Timing Controller

The PIOC timing functions are carried out in an AM9513 System Timing Controller containing 5 programmable 16 bit counters.
This circuit is described in detail in appendix C. Channel 1 of the device is used as PIOC real time clock while channels 2 to 5 are used as baud rate generators for the 4 PIOC serial lines.
Any practical baud rate and RTC frequency may be selected by proper programming of the AM9513. Input signal to the controller comes from a standard 4,9152 MHz oscillator.

5.3 Serial Lines

The serial line interface consists of 4 identical parts and can handle 4 full duplex channels. The line controllers can be programmed for synchronous or asynchronous transfer and can handle bit oriented (HDLC/SDLC) or byte oriented protocols.
Description and programming specifications for the serial controllers (Z80-SIO) is found in appendix D.

In order to obtain DMA transfer of data in both directions and to divide between input and output interrupts, the 4 2-channels SIO circuits are configured as follows:
In 2 of the circuits only the input side is used while the outputs are used only in the 2 remaining circuits. This leaves 4 unused inputs and 4 unused outputs without DMA possibilities and with reversed interrupt priority.

These channels have no separate connection to external lines but may be used to connect and disconnct X.21 lines while the "main" channels take care of the data transfer.
Therefore the extra channels are connected to the same data and clock lines as the main channels. The outputs are "or'ed" together so that if any of the twin lines are active, the line will be active (figure 6.3).

All 4 (main) channels are buffered for X-type (RS422) and V-type (RS232C) electrical levels. One PIOC will therefore have 4 plugs for V-24 as well as 4 plugs for X-21.
V-type interface is automatically selected when the V-type external cable is plugged in. The interface circuits used are listed in table 5.1.

Panel plug pin number	Modem plug pin number	Signal name
8	8	Ground
4	4	Receive Data A
16	11	Receive Data B
6	6	Clock A
18	13	Clock B
5	5	Indication A
17	12	Indication B
2	2	Transmit Data A
14	9	Transmit Data B
3	3	Control A
15	10	Control B

Table 5.1 X.21 Interface Circuits

Panel plug pin number	Modem plug pin number	Signal name
7	7	Ground
3	3	Receive Data
17	17	Receive Clock
5	5	Ready for Sending
6	6	Data Set Ready
8	8	Signal Detect
2	2	Transmit Data
15	15	Transmit Clock
4	4	Request to Send
20	20	Data Terminal Ready

Table 5.2 V-24 Interface Circuits

In order to disconnect all active serial lines in case of system failure, all 4 serial controllers and all X-type output buffers are connected to a common Watch-Dog circuit.
Normally the PIOC CPU should do a write access with bit 1 set in the WDOP address (table 6.4.3) every 0,5 seconds. The Watch-Dog will reset the serial lines 1 second after the last access.

Depending on the transfer mode there are many different connection schemes for receive and transmit clock. Cross connection of these signals is therefore done in the external cable plug.
When the correct cable is plugged in, routing of the clock signals is done at the same time. If the internal Baud Rate Generators are used the frequency must be software selected for each active channel.

The maximum baud rate for each channel is limited by the serial line controllers to 800 Kbits/second. In most applications the practical transfer rate will be much lower.
Even if the DMA circuits take care of the data transfer within the frames, the MC 68000 will have to program a new base address and byte count into the controller for each new frame or block of data.

Maximum line speed will then be decided by the delay between frames, the interrupt handling in MC 68000 and total load on the MC 68000. 38 Kbits on 4 duplex lines is an early estimate.

Maintenance mode may be selected by software. Serial output is then looped back to serial input of the same channel and internal baud rate is automatically selected.
The serial output will also be connected to the line. RTS will be looped back to RFS.

The X-type output signals may be disabled by a software command, making software controlled multi-drop lines possible.

5.3.1 Modem Control Lines

Each serial controller has 2 inputs and 2 outputs that may be used as modem control signals or as general purpose I/O lines. Table 5.3 shows how these lines are used in the PIOC. When X-type electrical levels (RS 422) are selected, the I-line corresponds to the RFS input and the C-line corresponds to the RTS output. All X-type outputs of a PIOC channel are disabled when DTR of the same channel is switched off.

PIOC function	PIOC channel	SIO number	Signal name	Active logic level
Request to send	0	1	RTS A	1
Ready for sending	0	1	CTS A	1
Data terminal ready	0	1	DTR A	1
Data set ready	0	1	DCD A	1
Maintenance mode enable	0	0	RTS A	1
Ready for sending	0	0	CTS A	1
	0	0	DTR A	1
Signal detect	0	0	DCD A	1
Request to send	1	1	RTS B	1
Ready for sending	1	1	CTS B	1
Data terminal ready	1	1	DTR B	1
Data set ready	1	1	DCD B	1
Maintenance mode enable	1	0	RTS B	1
Ready for sending	1	0	CTS B	1
	1	0	DTR B	1
Signal detect	1	0	DCD B	1
Maintenance mode enable	2	2	RTS A	1
Ready for sending	2	2	CTS A	1
	2	2	DTR A	1
Signal detect	2	2	DCD A	1
Maintenance mode enable	3	3	RTS B	1
Ready for sending	3	3	CTS B	1
	3	3	DTR B	1
Signal detect	3	3	DCD B	1
Request to send	2	2	RTS A	1
Ready for sending	2	2	CTS A	1
Data terminal ready	2	2	DTR A	1
Data set ready	2	2	DCD A	1
Request to send	3	3	RTS B	1
Ready for sending	3	3	CTS B	1
Data terminal ready	3	3	DTR B	1
Data set ready	3	3	DCD B	1

Table 5.3 Modem Control Lines

5.4 PIOC Internal Device Numbers

Hardware function	System function	Byte address (hex)
SIO-0 Data A	Input 0 Data	EF0000
SIO-0 Contr A	Input 0 Control	EF0008
SIO-0 Data B	Input 1 Data	EF0010
SIO-0 Contr B	Input 1 Control	EF0018
SIO-1 Data A	Output 0 Data	EF0020
SIO-1 Contr A	Output 0 Control	EF0028
SIO-1 Data B	Output 1 Data	EF0030
SIO-1 Contr B	Output 1 Control	EF0038
SIO-2 Data A	Input 2 Data	EF0040
SIO-2 Contr A	Input 2 Control	EF0048
SIO-2 Data B	Input 3 Data	EF0050
SIO-2 Contr B	Input 3 Control	EF0058
SIO-3 Data A	Output 2 Data	EF0060
SIO-3 Contr A	Output 2 Control	EF0068
SIO-3 Data B	Output 3 Data	EF0070
SIO-3 Contr B	Output 3 Control	EF0078

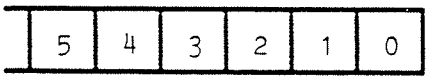
Table 5.4 PIOC Internal Device Numbers
Serial Line Controllers

Channel	Function	R/W	Byte Address Input DMA	Byte Address Output DMA
0	Base Address	R/W	EF0090	EF00B0
0	Word Count	R/W	EF0080	EF00A0
1	Base Address	R/W	EF0092	EF00B2
1	Word Count	R/W	EF0082	EF00A2
2	Base Address	R/W	EF0094	EF00B4
2	Word Count	R/W	EF0084	EF00A4
3	Base Address	R/W	EF0096	EF00B6
3	Word Count	R/W	EF0086	EF00A6
	Command/Status	W	EF0098	EF00B8
	Mask Set/Reset	W	EF009A	EF00BA
	Request reg.	W	EF0088	EF00A8
	Mode reg.	W	EF008A	EF00AA
	Temporary reg.	W	EF008C	EF00AC
	Mask Write	W	EF008E	EF00AE
	Clear FF	W	EF009C	EF00BC
	Reset	W	EF008C	EF00AC

Table 5.5 PIOC Internal Device Numbers
DMA Controllers

Device:	Function		R/W	Byte Address (Hex)
Interrupt Controller:		Data	R/W	EF00C0
Interrupt Controller:		Contr/Stat	R/W	EF00D0
WP:	Write protect			EF00E0
Timing Controller:		Data	R/W	EF0100
Timing Controller:		Contr/Stat	R/W	EF0101
OPCL:	Clear Opcom Interrupt			EF0120
SYREN:	Read Syndrome		R	EF0140
EAREN:	Read Error Address		R	EF0160
SCIP:	Set Interrupt to ND-100		W	EF0180
ELSE:	Error Log Select		W	EF01A0
WDOP:	Trig Watchdog		-	EF01C0

Table 5.6 PIOC Device Numbers
Miscellaneous Devices



ELSE: Error Log Select

Bit 1 = 0 : Disable Error Log
Bit 1 = 1 : Enable Error Log

Write Protection:

MC 68000:	DMA:
*WRITE PROTECT 1 > WP > PROTECT OFF (All of memory accessible by the MC 68000) 0 > WP > PROTECT ON (Supervisor access User no access)	OUTPUT > No restrictions INPUT > Never to protected area

NOTE: Write protect is set from the supervisor. Inquiries to the protect table are ignored in user routines with access to areas otherwise protected. This is used to give the user routine access to the I/O system and all of memory.

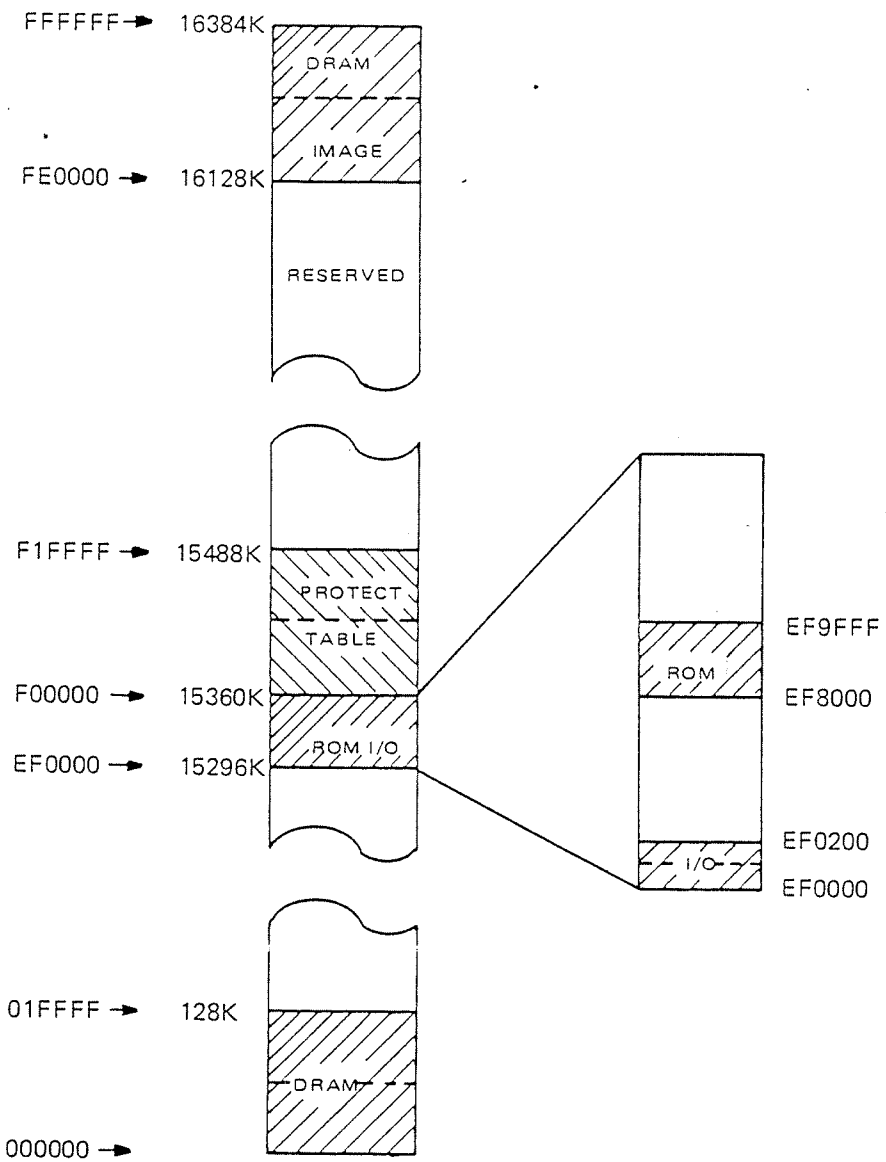


Fig. 5.4. PIOC Address Space (Version C Bytes)

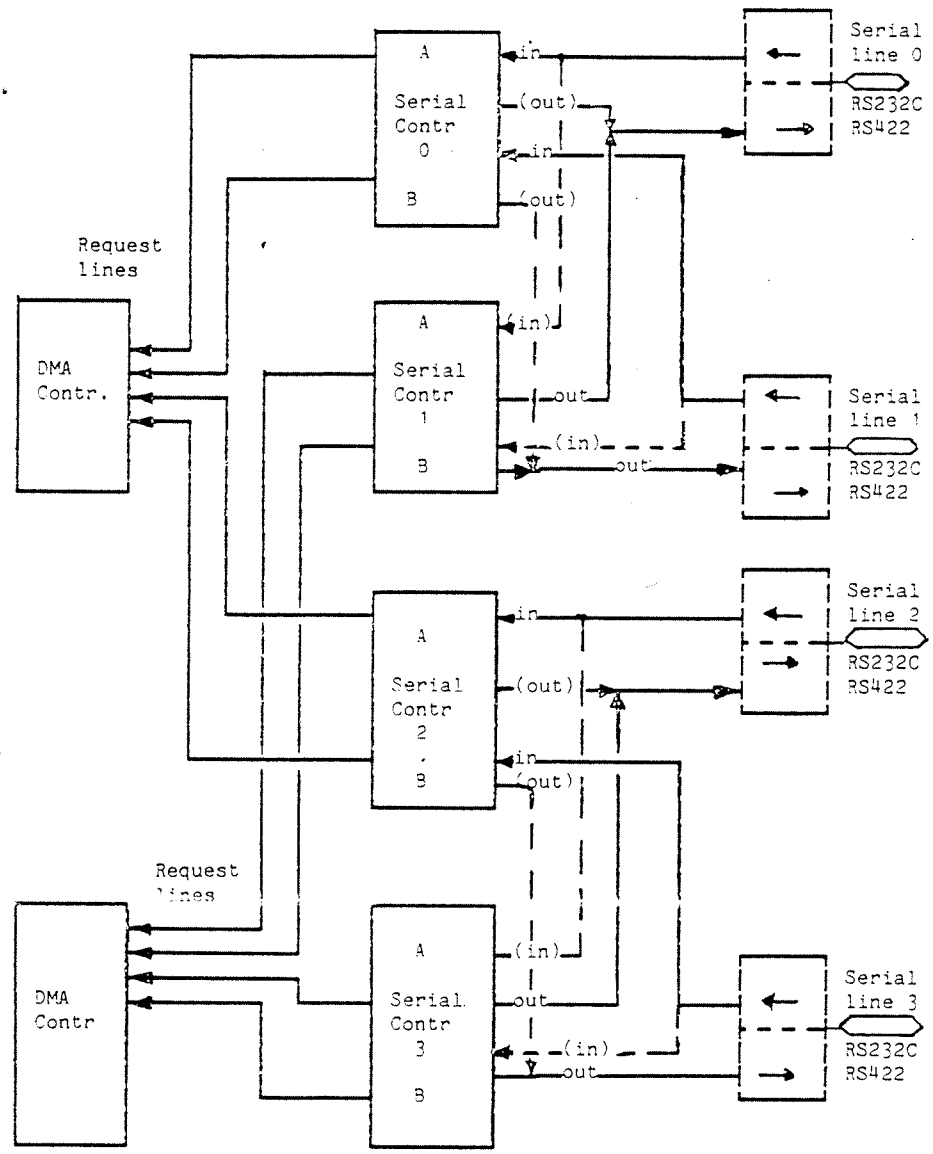
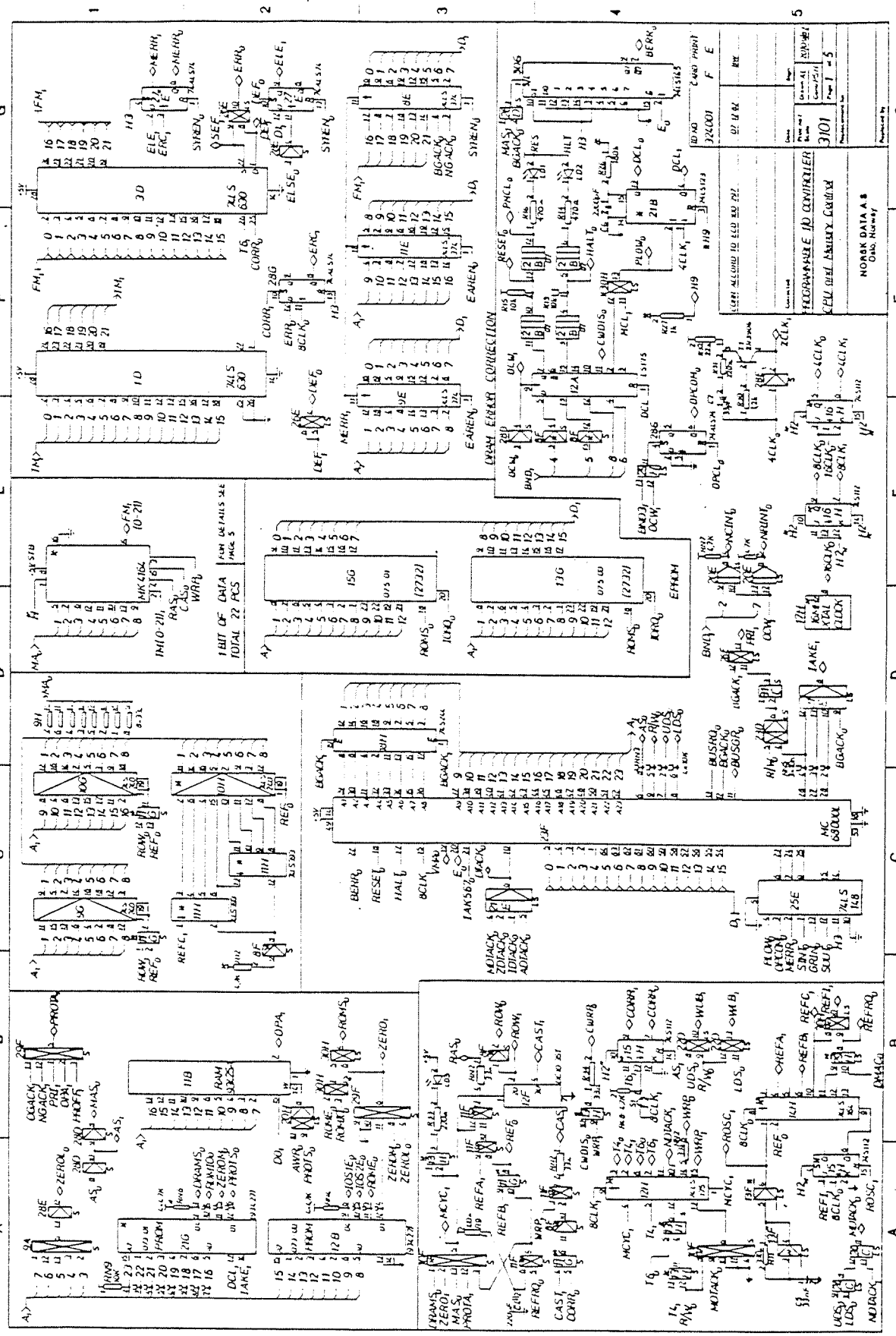


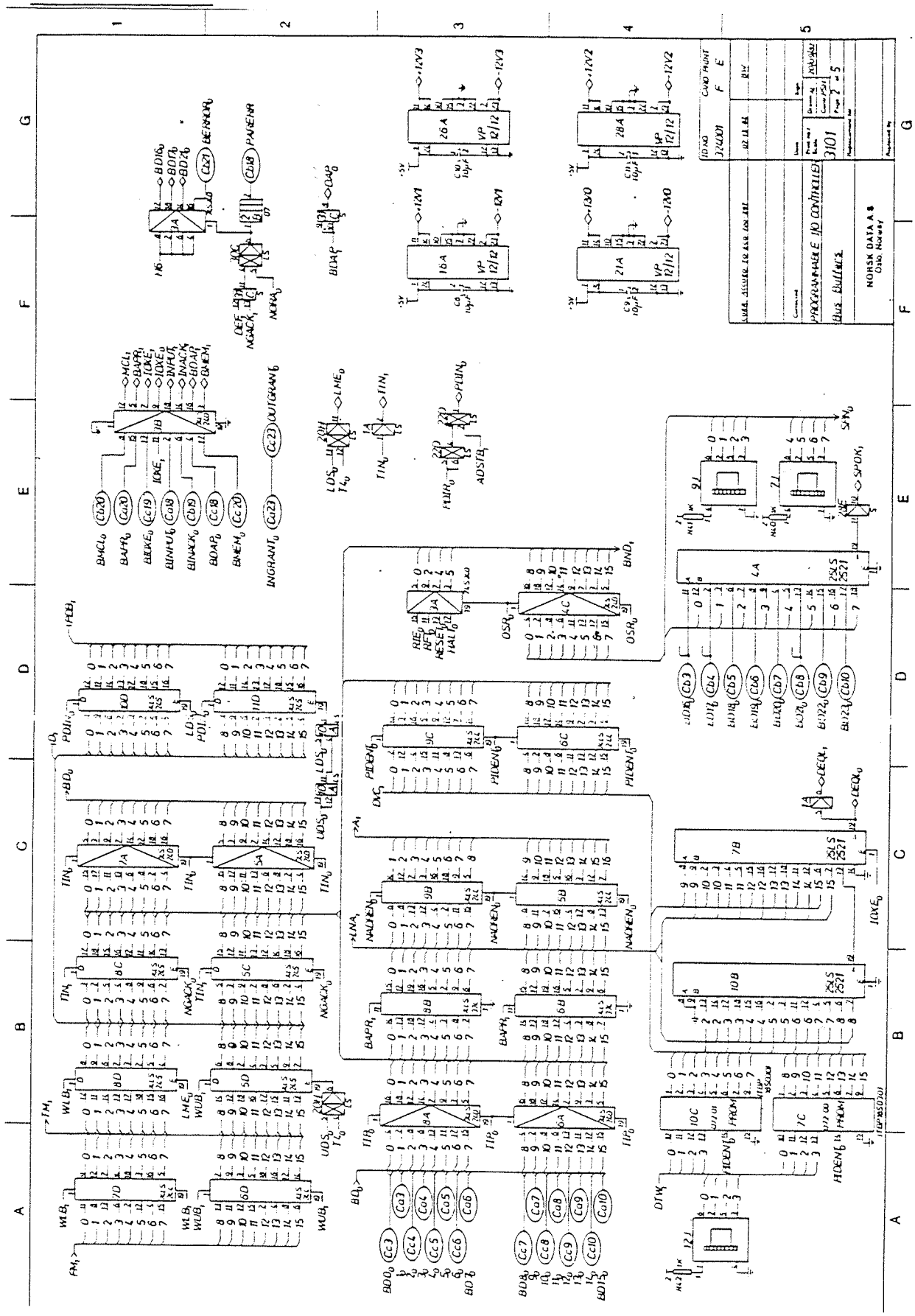
Fig. 5.5. Block Diagram of Serial Controllers and DMA Controllers
(X-21 Auxiliary Channels are dotted)

6 LOGIC DIAGRAMS

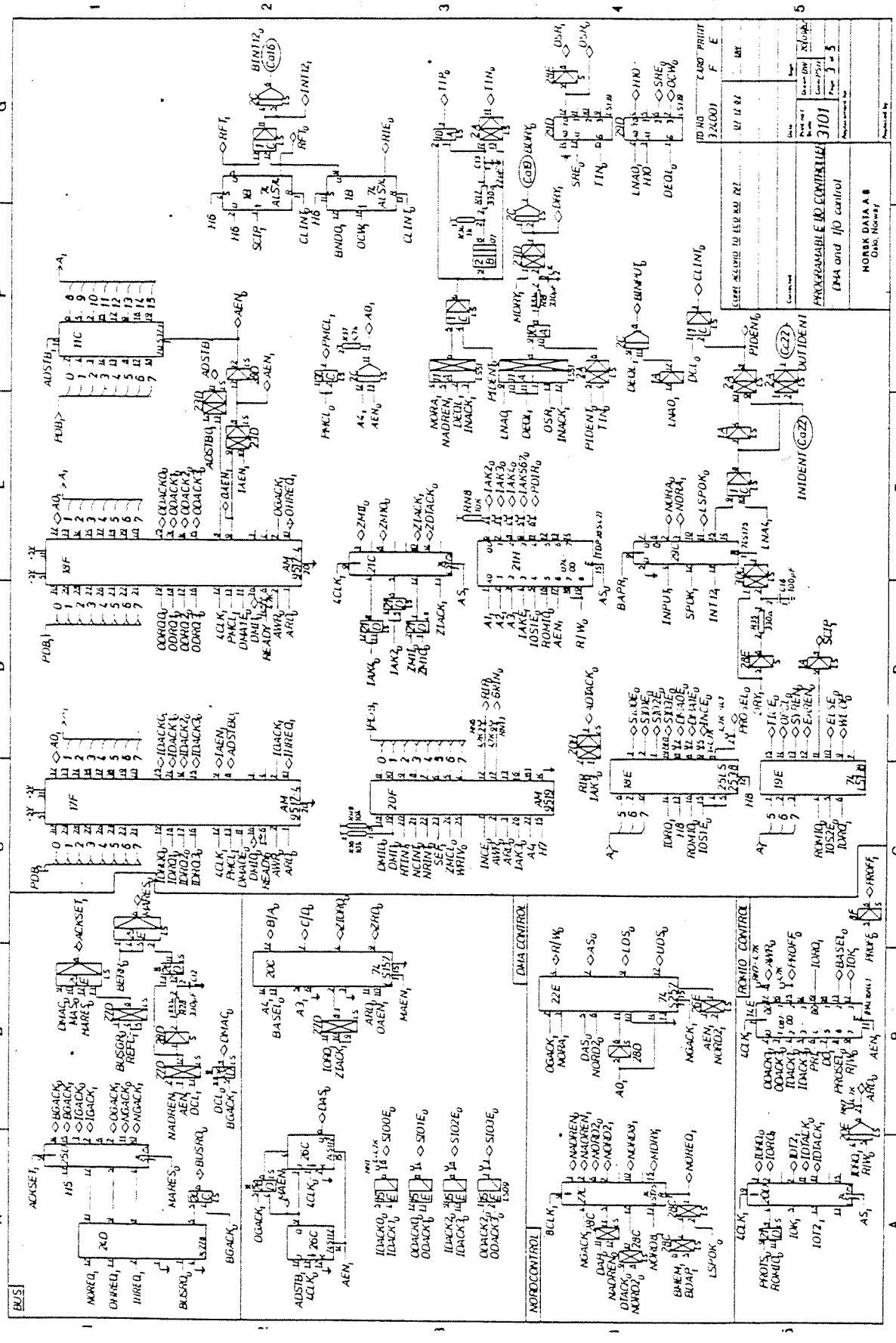
6.1 CPU and Memory Control



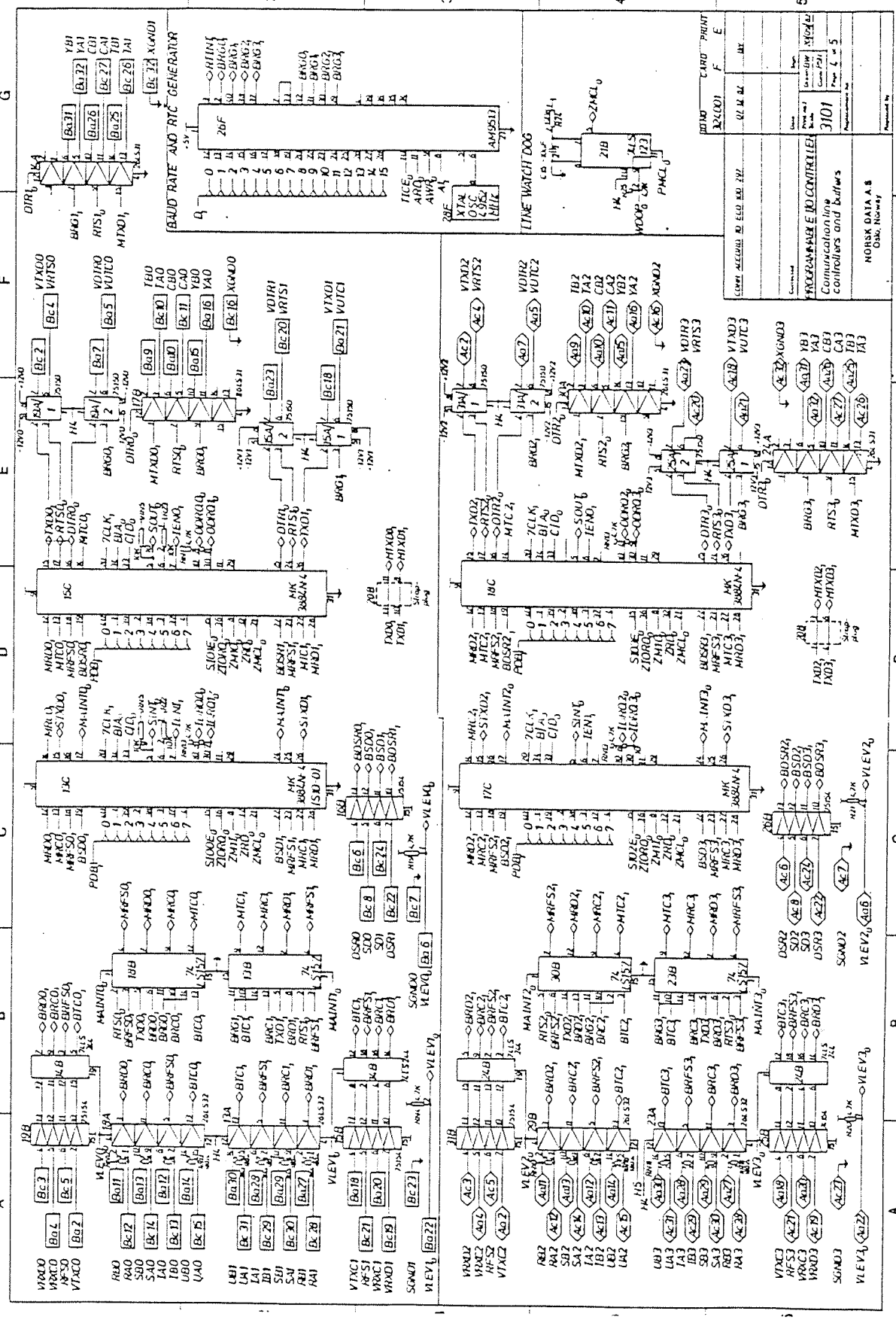
6.2 Bus Buffers



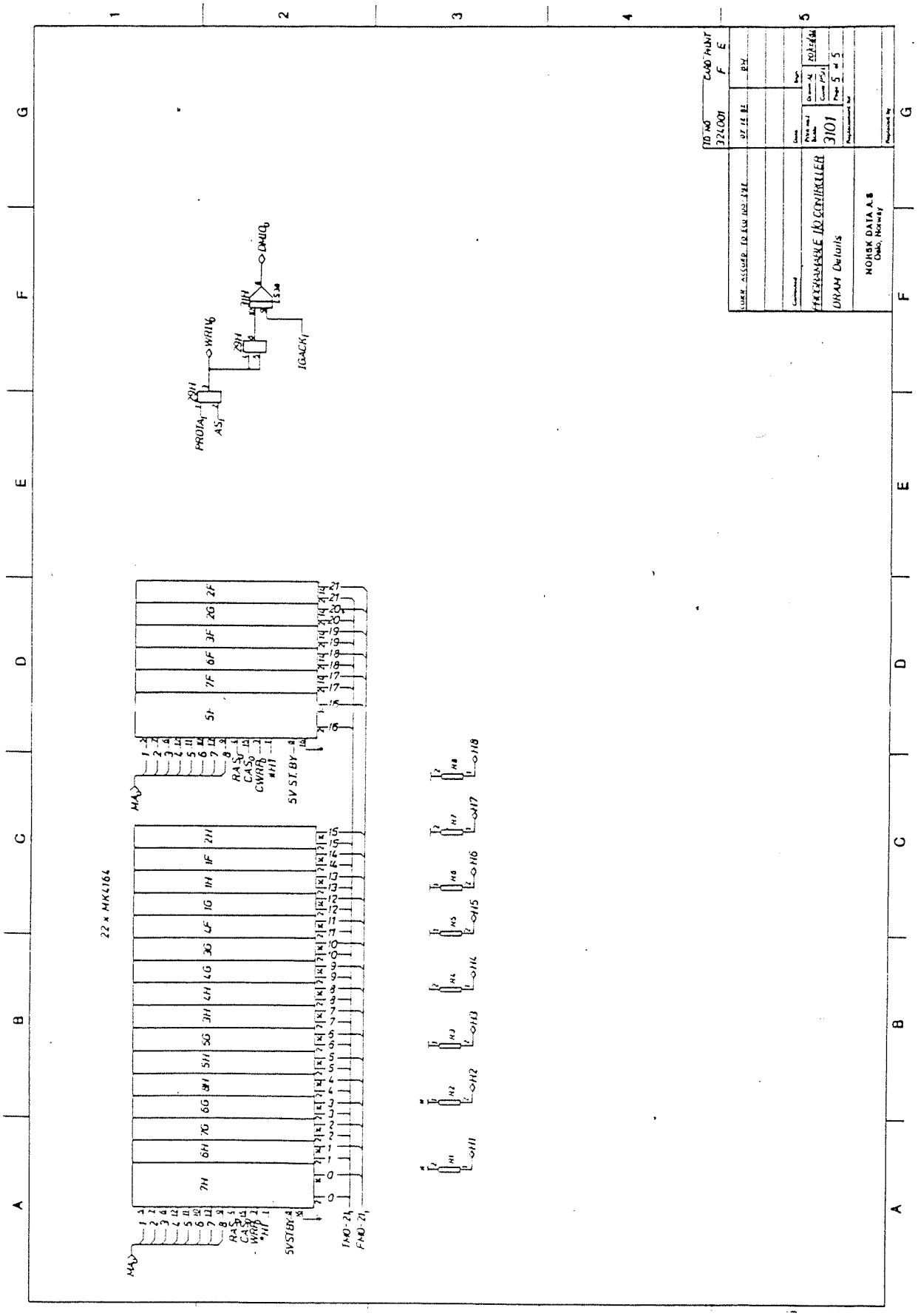
6.3 DMA and I/O Control



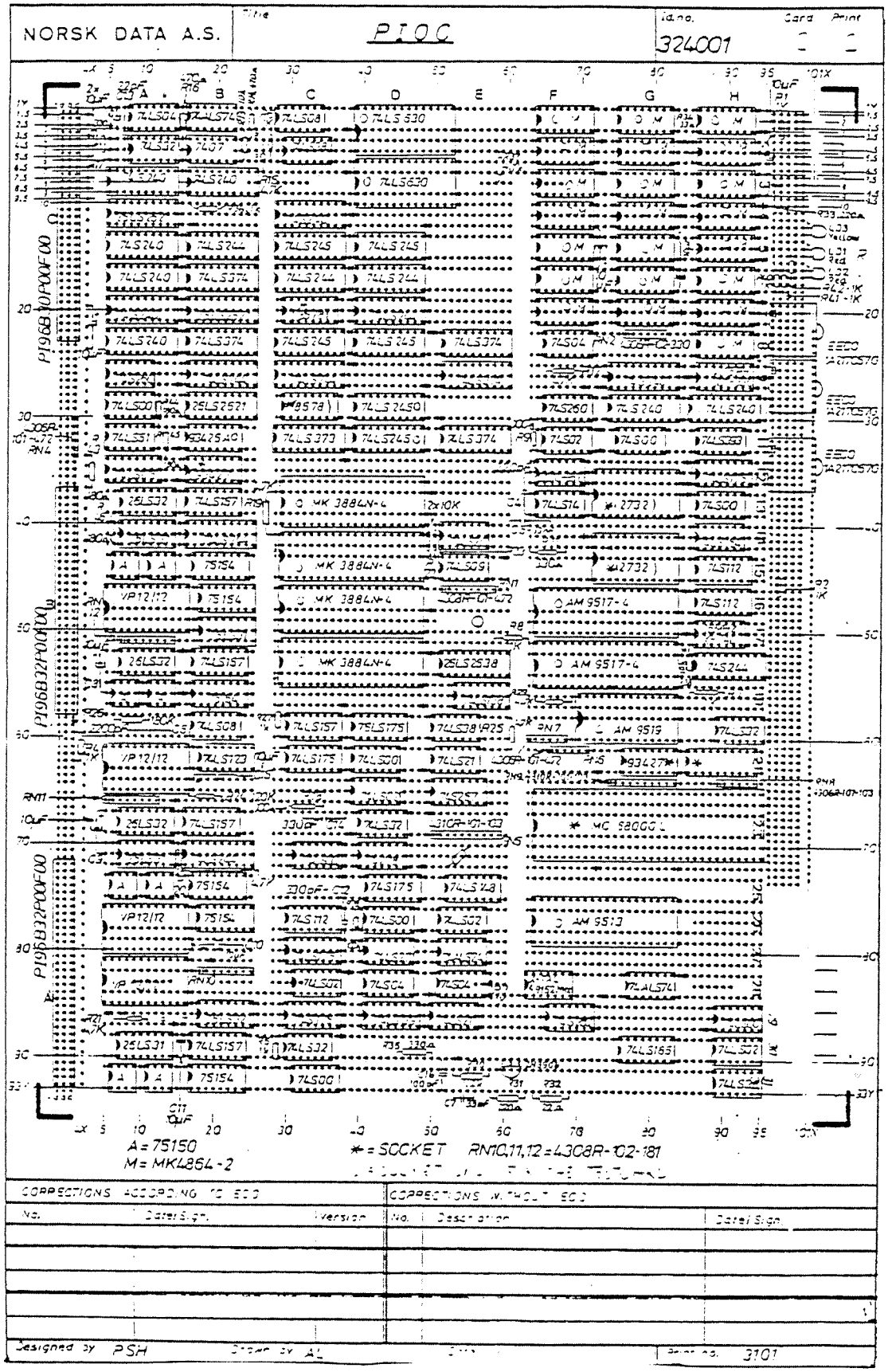
6.4 Communication Line Controllers and Buffers



6.5 DRAM Details



6.6 Layout of Components



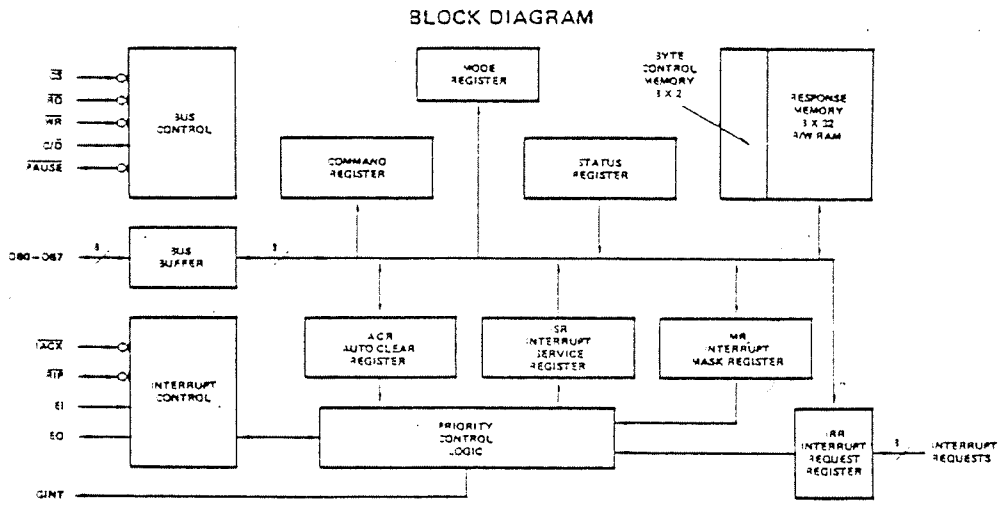
A P P E N D I X A

INTERRUPT CONTROLLER

The information in the appendices is taken from AM9500 Peripheral Products Interface Guide issued by Advanced Micro Devices and from the SIO manual issued by Zilog (UK) Limited.
Z80 is a Zilog trademark.

The AM 9519A Universal Interrupt Controller is a processor support circuit.
The circuit provides a powerful interrupt structure that increases the efficiency and the versatility of microcomputer based systems.

A single AM 9519A manages up to 8 maskable interrupt request inputs, resolves priorities and supplies up to 4 bytes of fully programmable response for each input.



AM 9519A Interrupt Controller Block Diagram

Usage in PIOC:

The interrupt controller must be initialized according to the PIOC hardware. This implies active low polarity for GINT and IREQ and single byte vector.
Correspondence between AM 9519A channels and PIOC interrupt levels is shown below.

PIOC Interrupt Level	AM 9519A Channel
3.7 Input DMA Error	0
3.6 Output DMA Finished	1
3.5 Real Time Clock	2
3.4 ND-100 Calling	3
3.3 ND-100 Ready	4
3.2 Single Memory Error	5
3.1 Comm. Controller Reset	6
3.0 Write Protect Violation	7

REGISTER DESCRIPTION

Interrupt Request Register (IRR): The 8-bit IRR is used to store pending interrupt requests. A bit in the IRR is set whenever the corresponding IREQ input goes active. Bits may also be set under program control from the CPU, thus permitting software generated interrupts. IRR bits may be cleared under program control. An IRR bit is automatically cleared when its interrupt is acknowledged. All IRR bits are cleared by a reset function.

Interrupt Service Register (ISR): The 8-bit ISR contains one bit for each IREQ input. It is used to indicate that a pending interrupt has been acknowledged and to mask all lower priority interrupts. When a bit is set by the acknowledge logic in the ISR, the corresponding IRR bit is cleared. If an acknowledged interrupt is not programmed to be automatically cleared, its ISR bit must be cleared by the CPU under program control when it is desired to permit interrupts from lower priority devices. When the interrupt is programmed for automatic clearing, the ISR bit is automatically reset during the acknowledge sequence. All ISR bits are cleared by a reset function.

Interrupt Mask Register (IMR): The 8-bit IMR is used to enable or disable the individual interrupt inputs. The IMR bits correspond to the IREQ inputs and all eight may be loaded, set or cleared in parallel under program control. In addition, individual IMR bits may be set or cleared by the CPU. A reset function will set all eight mask bits, disabling all requests. A mask bit that is set does not disable the IRR, and an IREQ that arrives while a corresponding mask bit is set will cause an interrupt later when the mask bit is cleared. Only unmasked interrupt inputs can generate a Group Interrupt output.

Response Memory: An 8 x 32 read/write response memory is included in the Am9519A. It is used to store up to four bytes of response information for each of the eight interrupt request inputs. All bits in the memory are programmable, allowing any desired vector, opcode, instruction or other data to be entered. The Am9519A transfers the interrupt response information for the highest priority unmasked interrupt from the memory to the data bus when the $\overline{IACK}$ input is active.

Auto Clear Register: The 8-bit Auto Clear register contains one bit for each IREQ input and specifies the operating mode for each of the ISR bits. When an auto clear bit is off, the corresponding ISR bit is set when that interrupt is acknowledged and is cleared by software command. When an auto clear bit is on, the corresponding ISR bit is cleared by the hardware at the end of the acknowledge sequence. A reset function clears all auto clear bits.

Status Register: The 8-bit Status register contains information concerning the internal state of the chip. It is especially useful when operating in the polled mode in order to identify interrupting devices. Figure 1 shows the status register bit assignments. The polarity of the GINT bit 7 is not affected by the GINT polarity control (Mode bit status register bit assignments. Bits S0-S2 are set asynchronously to a status register read operation. It is recommended to read the register twice and to compare the binary vectors for equality prior to the proceeding with device service in polled mode. The polarity of the GINT bit 7 is not affected by the GINT polarity control (Mode bit 3). The Status register is read by executing a read operation ($\overline{CS}$ = 0, RS = 0) with the control location selected (C/$\overline{D}$ = 1).

Mode Register: The 8-bit Mode register controls the operating options of the Am9519A. Figure 2 shows the bit assignments for the Mode register. The five low order mode bits (0 through 4) are loaded in parallel by command. Bits 5, 6 and 7 are controlled by separate commands. (See Figure 4.) The Mode register cannot be read out directly to the data bus, but Mode bits 0, 2 and 7 are available as part of the Status register.

Command Register: The 8-bit Command register stores the last command entered. Depending upon the command opcode, it may initiate internal actions or precondition the part for subsequent data bus transfers. The Command register is loaded by executing a write operation ($\overline{WR}$ = 0) with the control location selected (C/$\overline{D}$ = 1), as shown in Figure 3.

Byte Count Register: The length in bytes of the response associated with each interrupt is independently programmed so that different interrupts may have different length responses. The byte count for each response is stored in eight 2-bit Byte Count registers. For a given interrupt the Am9519A will expect to receive a number of $\overline{IACK}$ pulses that equals the corresponding byte count, and will hold $\overline{RIP}$ low until the count is satisfied.

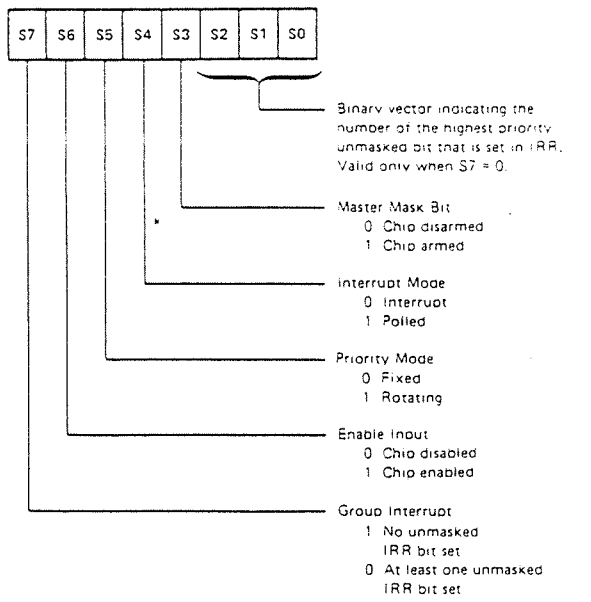


Figure 1. Status Register Bit Assignments. MOS-025

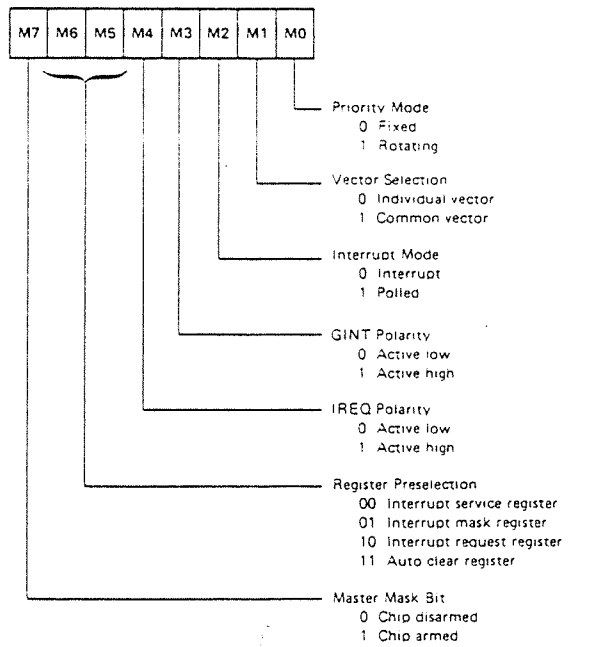


Figure 2. Mode Register Bit Assignments. MOS-026

FUNCTIONAL DESCRIPTION

Interrupts are used to improve system throughput and response time by eliminating heavy dependence on software polling procedures. Interrupts allow external devices to asynchronously modify the instruction sequence of a program being executed. In systems with multiple interrupts, vectoring can further improve performance by allowing direct identification of the interrupting device and its associated service routine. The Am9519A Universal Interrupt Controller contains, on one chip, all of the circuitry necessary to detect, prioritize and manage eight vectored interrupts. It includes many options and operating modes that permit the design of sophisticated interrupt systems.

Reset

The reset function is accomplished by software command or automatically during power-up. The reset command may be issued by the CPU at any time. Internal power up circuitry is triggered when VCC reaches a predetermined threshold, causing a brief internal reset pulse. In both cases, the resulting internal state of the machine is that all registers are cleared except the Mask register which is set. Thus no Group Interrupt will be generated and no interrupt requests will be recognized. The response memory and Byte Count registers are not affected by reset. Their contents after power-up are unpredictable and must be established by the host CPU during initialization.

Operating Sequence

A brief description of a typical sequence of events in an operating interrupt system will illustrate the general interactions among the host CPU, the interrupt controller and the interrupting peripheral.

1. The Am9519A controller is initialized by the CPU in order to customize its configuration and operation for the application at hand. Both the controller and the CPU are then enabled to accept interrupts.
2. One (or more) of the interrupt request inputs to the controller becomes active indicating that peripheral equipment is asking for service. The controller asynchronously accepts and latches the request(s).
3. If the request is masked, no further action takes place. If the request is not masked, a Group Interrupt output is generated by the controller.
4. The GINT signal is recognized by the CPU which normally will complete the execution of the current instruction, insert an interrupt acknowledge sequence into its instruction execution stream, and disable its internal interrupt structure. The controller expects to receive one or more $\overline{IACK}$ signals from the CPU during the acknowledge sequence.
5. When the controller receives the $\overline{IACK}$ signal, it brings $\overline{PAUSE}$ low and selects the highest priority unmasked pending request. When selection is complete, the $\overline{RIP}$ output is brought low and the first byte in the response memory associated with the selected request is output on the data bus. $\overline{PAUSE}$ stays low until RIP goes low. $\overline{RIP}$ stays low until the last byte of the response has been transferred.
6. During the acknowledge sequence, the IRR bit corresponding to the selected request is automatically cleared, and the corresponding ISR bit is set. When the ISR bit is set, the Group Interrupt output is disabled until a higher priority request arrives or the ISR bit is cleared. The ISR bit will be cleared by either hardware or software.
7. If a higher priority request arrives while the current request is being serviced, GINT will be output by the controller, but will be recognized and acknowledged only if the CPU has its interrupt input enabled. If acknowledged, the corresponding higher priority ISR bit will be set and the requests nested.

Information Transfers

Figure 3 shows the control signal configurations for all information transfer operations between the Am9519 and the data bus. The following conventions are assumed: $\overline{RD}$ and $\overline{WR}$ active are mutually exclusive; $\overline{RD}$, $\overline{WR}$ and C/$\overline{D}$ have no meaning unless $\overline{CS}$ is low; active $\overline{IACK}$ pulses occur only when $\overline{CS}$ is high.

For reading, the Status register is selected directly by the C/$\overline{D}$ control input. Other internal registers are read by preselecting the desired register with mode bits 5 and 6, and then executing a data read. The response memory can be read only with $\overline{IACK}$ pulses. For writing, the Command register is selected directly by the C/$\overline{D}$ control input. The Mask and Auto Clear registers are loaded following specific commands to that effect. To load each level of the response memory, the response preselect command is issued to select the desired level. An appropriate number of data write operations are then executed to load that level.

CONTROL INPUT					DATA BUS OPERATION
$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{IACK}$	
0	0	0	1	1	Transfer contents of preselected data register to data bus
0	0	1	0	1	Transfer contents of data bus to preselected data register
0	1	0	1	1	Transfer contents of status register to data bus
0	1	1	0	1	Transfer contents of data bus to command register
1	X	X	X	0	Transfer contents of selected response memory location to data bus
1	X	X	X	1	No information transferred

Figure 3. Summary of Data Bus Transfers.

The Pause output may be used by the host CPU to ensure that proper timing relationships are maintained with the Am9519A when $\overline{IACK}$ is active. The $\overline{IACK}$ pulse width required depends on several variables, including: operating temperature, internal logic delays, number of interrupt controllers chained together, and the priority level of the interrupt being acknowledged. When delays in these variables combine to delay selection of a request following the falling edge of the first $\overline{IACK}$, the Pause output may be used to extend the $\overline{IACK}$ pulse, if necessary. Pause will remain low until a request has been selected, as indicated by the falling edge of $\overline{RIP}$. Typically, the internal interrupt selection process is quite fast, especially for systems with a single Am9519A and Pause will consequently remain low for only a very brief interval and will not cause extension of the IACK timing.

Operating Options

The Mode register specifies the various combinations of operating options that may be selected by the CPU. It is cleared by power-up or by a reset command.

Mode bit 0 specifies the rotating/fixed priority mode (see Figure 2). In the fixed mode, priority is assigned to the request inputs based upon their physical location at the chip interface, with IREQ0 the highest and IREQ7 the lowest. In the rotating mode, relative priority is the same as for the fixed mode and the most recently serviced request is assigned the lowest priority. In the fixed mode, a lower priority request might never receive service if enough higher priority requests are active. In the rotating mode, any request will receive service within a maximum of seven other service cycles no matter what pattern the request inputs follow.

Mode bit 1 selects the individual/common vector option. Individual vectoring provides a unique location in the response memory for each interrupt request. The common vector option always supplies the response associated with IREQ0 no matter which request is being acknowledged.

Mode bit 2 specifies interrupt or polled operation. In the polled mode the Group Interrupt output is disabled. The CPU may read the Status register to determine if a request is pending. Since $\overline{IACK}$ pulses are not normally supplied in polled mode, the IRR bit is not automatically cleared, but may be cleared by command. With no $\overline{IACK}$ input the ISR and the response memory are not used. An Am9519A in the polled mode has EI connected to EO so that in multichip interrupt systems the polled chip is functionally removed from the priority hierarchy.

Mode bit 3 specifies the sense of the GINT output. When active high polarity is selected the output is a two-state configuration. For active low polarity, the output is open drain and requires an external pull-up resistor to provide the high logic level. The open drain output allows wired-or configurations with other similar output signals.

Mode bit 4 specifies the sense of the IREQ inputs. When active low polarity is selected, the IRR responds to falling edges on the request inputs. When active high is selected, the IRR responds to rising edges.

Mode bits 5 and 6 specify the register that will be read on subsequent data read operations (C/$\overline{D}$ = 0, $\overline{RD}$ = 0). This preselection remains valid until changed by a reset or a command.

Mode bit 7 is the master mask bit that disables all request inputs. It is used to disable all interrupts without modifying the IMR so that the previous IMR contents are valid when interrupts are re-enabled. When the master mask bit is low, it causes the EO line to remain disabled (low). Thus, for multiple-chip interrupt systems, one master mask bit can disable the whole interrupt structure. Alternatively, portions of the structure may be disabled. The state of the master mask bit is available as bit S3 of the Status register.

Programming

After reset, the Am9519A must be initialized by the CPU in order to perform useful work. At a minimum, the master mask bit and at least one of the IMR bits should be enabled. If vectoring is to be used, the response memory must be loaded; if not, the mode must be changed to a non-vectored configuration. Normally, the first step will be to modify the Mode register and the Auto clear register in order to establish the configuraton desired for the application. Then the response memory and byte count will be loaded for those request levels that will be in use. Finally, the master mask bit and at least portions of the IMR will be enabled to allow interrupt processing to proceed.

Interrupt Controller Programming Example:

```
MODULE PER-IN       % PERIPHERAL INITIALISATION

CONSTANT INCD                   = EF00C0H           % DATA ADDRESS
CONSTANT INCC                   = EF00D0H           % CONTROL ADDRESS
%
%         INTERRUPT CONTROLLER PARAMETERS
%
BYTES : INPART := (&
                  300B ,&       % AUTO CLEAR POINTER
                  376B ,&       % AUTO CLEAR DATA
                  340B ,&       % CHANNEL 0 VECTOR POINTER
                  107B ,&       % DMA INPUT VECTOR
                  341  ,&
                  106B ,&       % DMA OUTPUT VECTOR
                  342B ,&
                  105B ,&       % RTC VECTOR
                  343B ,&
                  104B ,&       % NORD CALLING VECTOR
                  344B ,&
                  103B ,&       % NORD RDY VECTOR
                  345B ,&
                  102B ,&       % SEF VECTOR
                  346B ,&
                  101B ,&       % ZMCL VECTOR


%         INTERRUPT CONTROLLER INITIALISATION:
%         THE INTERRUPT CONTROLLER (LEVEL 3) IS INITIALIZED,
%         AND LEVELS 1 TO 7 ARE ENABLED.

          ROUTINE SPECIAL VOID,VOID:INTC-IN

LABEL : INTC1

$*        MOVE.L #INCD,A0
$*        MOVE.L #INCC,A1
$*        MOVE.W #0FF00H,(A1)           % RESET
$*        MOVE.W #0007H,D0              % SET TABLE LENGTH
$*        LEA INPART,A4
$*        MOVE.B(A4)+,(A1)              % LOAD REG. POINTER
$*        MOVE.B(A4)+,(A0)              %LOAD DATA
$*        DBEQ D0,INTC1                 % NEXT REG.
$*        MOVE.W #0FF80H,(A1)           % SET MODE REG.
$*        MOVE.W #0FFA1H,(A1)           % ARM
$*        MOVE.W #0FFB0H,(A1)           % PRESELECT MASK REGISTER
$*        MOVE.W #0FF01H,(A0)           % SET MASK
$*        RTS

      ENDROUTINE
ENDMODULE
```

Commands

The host CPU configures, changes and inspects the internal condition of the Am9519A using the set of commands shown in Figure 4. An "X" entry in the table indicates a "don't care" state. All commands are entered by directly loading the Command register as shown in Figure 3 ($C/\overline{D}$ = 1, $\overline{WR}$ = 0). Figure 5 shows the coding assignments for the Byte Count registers. A detailed description of each command is contained in the Am9519A Application Note AMPUB-071.

BY1	BY0	COUNT
0	0	1
0	1	2
1	0	3
1	1	4

Figure 5. Byte Count Coding.

COMMAND CODE								COMMAND DESCRIPTION
7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	Reset
0	0	0	1	0	X	X	X	Clear all IRR and all IMR bits
0	0	0	1	1	B2	B1	B0	Clear IRR and IMR bit specified by B2, B1, B0
0	0	1	0	0	X	X	X	Clear all IMR bits
0	0	1	0	1	B2	B1	B0	Clear IMR bit specified by B2, B1, B0
0	0	1	1	0	X	X	X	Set all IMR bits
0	0	1	1	1	B2	B1	B0	Set IMR bit specified by B2, B1, B0
0	1	0	0	0	X	X	X	Clear all IRR bits
0	1	0	0	1	B2	B1	B0	Clear IRR bit specified by B2, B1, B0
0	1	0	1	0	X	X	X	Set all IRR bits
0	1	0	1	1	B2	B1	B0	Set IRR bit specified by B2, B1, B0
0	1	1	0	X	X	X	X	Clear highest priority ISR bit
0	1	1	1	0	X	X	X	Clear all ISR bits
0	1	1	1	1	B2	B1	B0	Clear ISR bit specified by B2, B1, B0
1	0	0	M4	M3	M2	M1	M0	Load Mode register bits 0–4 with specified pattern
1	0	1	0	M6	M5	0	0	Load Mode register bits 5, 6 with specified pattern
1	0	1	0	M6	M5	0	1	Load Mode register bits 5, 6 and set mode bit 7
1	0	1	0	M6	M5	1	0	Load Mode register bits 5, 6 and clear mode bit 7
1	0	1	1	X	X	X	X	Preselect IMR for subsequent loading from data bus
1	1	0	0	X	X	X	X	Preselect Auto Clear register for subsequent loading from data bus
1	1	1	BY1	BY0	L2	L1	L0	Load BY1, BY0 into byte count register and preselect response memory level specified by L2, L1, L0 for subsequent loading from data bus

Figure 4. Am9519A Command Summary.

A P P E N D I X B

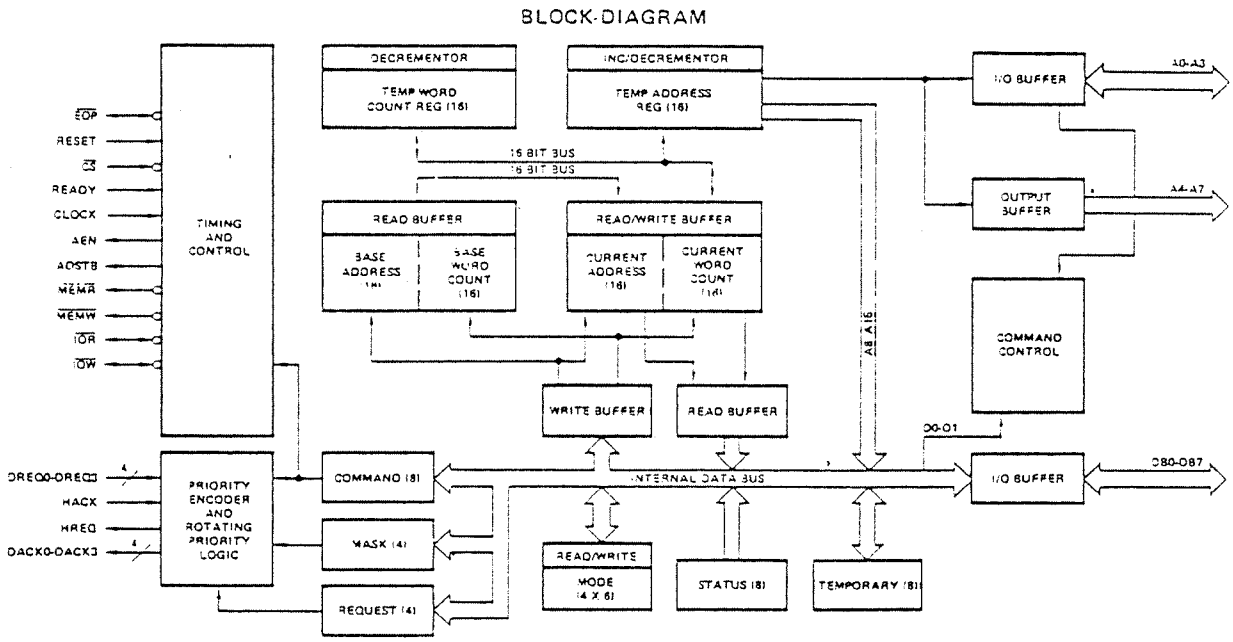
DMA CONTROLLERS

The information in the appendices is taken from AM9500 Peripheral Products Interface Guide issued by Advanced Micro Devices and from the SIO manual issued by Zilog (UK) Limited.
Z80 is a Zilog trademark.

The AM 9517 Multimode Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems.
It is designed to improve system performance by allowing external devices to transfer information directly to or from the system memory.
The AM 9517 contains 4 independant channels.

3 basic transfer modes may be program selected by the user. Each channel can be automatically programmed to Autoinitialize to its original condition following an End of Process (EOP).

Each channel has 64K address and byte count capability. An external EOP signal can terminate a DMA trasnsfer.



AM 9517 DMA Controllers Block Diagram

Usage in PIOC:

The DMA Controllers have to be initialized according to the PIOC hardware configuration. In addition to the fixed transfer direction for each device plus DREQ and DACK polarities, normal timing and extended write have to be selected.

Memory to memory cascade modes cannot be used. A programming example is shown at the end of this appendix.

FUNCTIONAL DESCRIPTION

The Am9517A block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The Am9517A contains 344 bits of internal memory in the form of registers. Figure 2 lists these registers by name and shows the size of each. A detailed description of the registers and their functions can be found under Register Description.

The Am9517A contains three basic blocks of control logic. The Timing Control block generates internal timing and external control signals for the Am9517A. The Program Command Control block decodes the various commands given to the Am9517A by the microprocessor prior to servicing a DMA Request. It also decodes each channel's Mode Control word. The Priority Encoder block resolves priority contention among DMA channels requesting service simultaneously.

The Timing Control block derives internal timing from the clock input. In Am9080A systems this input will usually be the $\phi 2$ TTL clock from an Am8224. However, any appropriate system clock will suffice.

DMA Operation

The Am9517A is designed to operate in two major cycles. These are called Idle and Active cycles. Each device cycle is made up of a number of states. The Am9517A can assume seven separate states, each composed of one full clock period. State 1 (S1) is the inactive state. It is entered when the Am9517A has no valid DMA requests pending. While in S1, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. State 0 (S0) is the first state of a DMA service. The Am9517A has requested a hold but the processor has not yet returned an acknowledge. An acknowledge from the CPU will signal that transfers may begin. S1, S2, S3 and S4 are the working states of the DMA service. If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted before S4 by the use of the Ready line on the Am9517A.

Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer. The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for each complete transfer. The first four states (S11, S12, S13, S14) are used for the read-from-memory half and the last four states (S21, S22, S23 and S24) for the write-to-memory half of the transfer. The Temporary Data register is used for intermediate storage of the memory byte.

IDLE Cycle

When no channel is requesting service, the Am9517A will enter the Idle cycle and perform "S1" states. In this cycle the Am9517A will sample the DREQ lines every clock cycle to determine if any channel is requesting a DMA service. The device will also sample $\overline{CS}$, looking for an attempt by the microprocessor to write or read the internal registers of the Am9517A. When $\overline{CS}$ is low and HACK is low the Am9517A enters the Program Condition. The CPU can now establish, change or inspect the internal definition of the part by reading from or writing to the internal registers. Address lines A0-A3 are inputs to the device and select which registers will be read or written. The $\overline{IOR}$ and $\overline{IOW}$ lines are used to select and time reads or writes. Due to the number and size of the internal registers, an internal flip/flop is used to generate an additional bit of address. This bit is used to determine the upper or lower byte of the 16-bit Address and Word Count registers. The flip/flop is reset by Master Clear or Reset. A separate software command can also reset this flip/flop.

Special software commands can be executed by the Am9517A in the Program Condition. These commands are decoded as sets of addresses when both $\overline{CS}$ and $\overline{IOW}$ are active and do not make use of the data bus. Functions include Clear First/Last Flip/Flop and Master Clear

Name	Size	Number
Base Address Registers	16 bits	4
Base Word Count Registers	16 bits	4
Current Address Registers	16 bits	4
Current Word Count Registers	16 bits	4
Temporary Address Register	16 bits	1
Temporary Word Count Register	16 bits	1
Status Register	8 bits	1
Command Register	8 bits	1
Temporary Register	8 bits	1
Mode Registers	6 bits	4
Mask Register	4 bits	1
Request Register	4 bits	1

Figure 2. Am9517A Internal Registers.

ACTIVE CYCLE

When the Am9517A is in the Idle cycle and a channel requests a DMA service, the device will output a HREQ to the microprocessor and enter the Active cycle. It is in this cycle that the DMA service will take place, in one of four modes:

Single Transfer Mode: In Single Transfer mode, the Am9517A will make a one-byte transfer during each HREQ/HACK handshake. When DREQ goes active, HREQ will go active. After the CPU responds by driving HACK active, a one-byte transfer will take place. Following the transfer, HREQ will go inactive, the word count will be decremented and the address will be either incremented or decremented. When the word count goes to zero a Terminal Count (TC) will cause an Autoinitialize if the channel has been programmed to do so.

To perform a single transfer, DREQ must be held active only until the corresponding DACK goes active. If DREQ is held continuously active, HREQ will go inactive following each transfer and then will go active again and a new one-byte transfer will be made following each rising edge of HACK. In 8080A/9080A systems this will ensure one full machine cycle of execution between DMA transfers. Details of timing between the Am9517A and other bus control protocols will depend upon the characteristics of the microprocessor involved.

Block Transfer Mode: In Block Transfer mode, the Am9517A will continue making transfers until a TC (caused by the word count going to zero) or an external End of Process ($\overline{EOP}$) is encountered. DREQ need be held active only until DACK becomes active. An autoinitialize will occur at the end of the service if the channel has been programmed for it.

Demand Transfer Mode: In Demand Transfer mode the device will continue making transfers until a TC or external $\overline{EOP}$ is encountered or until DREQ goes inactive. Thus, the device requesting service may discontinue transfers by bringing DREQ inactive. Service may be resumed by asserting an active DREQ once again. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count may be read from the Am9517A Current Address and Current Word Count registers. Autoinitialization will only occur following a TC or $\overline{EOP}$ at the end of service. Following Autoinitialization, an active-going DREQ edge is required to initiate a new DMA service.

TRANSFER TYPES

Each of the three active transfer modes can perform three different types of transfers. These are Read, Write and Verify. Write transfers move data from an I/O device to the memory by activating $\overline{IOR}$ and MEMW. Read transfers move data from memory to an I/O device by activating MEMR and $\overline{IOW}$. Verify transfers are pseudo transfers: the Am9517A operates as in Read or Write transfers generating addresses, responding to $\overline{EOP}$, etc., however the memory and I/O control lines remain inactive.

Memory-to-Memory: The Am9517A includes a block move capability that allows blocks of data to be moved from one memory address space to another. When Bit C0 in the Command register is set to a logical 1, channels 0 and 1 will operate as memory-to-memory transfer channels. Channel 0 forms the source address and channel 1 forms the destination address. The channel 1 word count is used. A memory-to-memory transfer is initiated by setting a software DMA request for channel 0. Block Transfer Mode should be used for memory-to-memory. When channel 0 is programmed for a fixed source address, a single source word may be written into a block of memory.

When setting up the Am9517A for memory-to-memory operation, it is suggested that both channels 0 and 1 be masked out. Further, the channel 0 word count should be initialized to the same value used in channel 1. No DACK outputs will be active during memory-to-memory transfers.

The Am9517A will respond to external $\overline{EOP}$ signals during memory-to-memory transfers. Data comparators in block search schemes may use this input to terminate the service when a match is found. The timing of memory-to-memory transfers may be found in Timing Diagram 4.

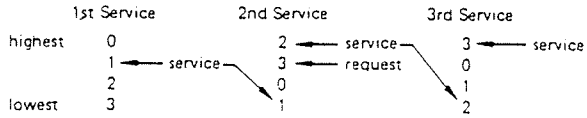
Autointialize: By programming a bit in the Mode register a channel may be set up for an Autoinitialize operation. During Autoinitialization, the original values of the Current Address and Current Word Count registers are automatically restored from the Base Address and Base Word Count registers of that channel following $\overline{EOP}$. The base registers are loaded simultaneously with the current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not set by $\overline{EOP}$ when the channel is in Autoinitialize. Following Autoinitialize the channel is ready to repeat its service without CPU intervention.

Cascade Mode: This mode is used to cascade more than one Am9517A together for simple system expansion. The HREQ and HACK signals from the additional Am9517A are connected to the DREQ and DACK signals of a channel of the initial Am9517A. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Since the cascade channel in the initial device is used only for prioritizing the additional device, it does not output any address or control signals of its own. These would conflict with the outputs of the active channel in the added device. The Am9517A will respond to DREQ with DACK but all other outputs except HREQ will be disabled.

Figure 3 shows two additional devices cascaded into an initial device using two of the previous channels. This forms a two level DMA system. More Am9517As could be added at the second level by using the remaining channels of the first level. Additional devices can also be added by cascading into the channels of the second level devices forming a third level.

Priority: The Am9517A has two types of priority encoding available as software selectable options. The first is Fixed Priority which fixes the channels in priority order based upon the descending value of their number. The channel with the lowest priority is 3 followed by 2, 1 and the highest priority channel, 0.

The second scheme is Rotating Priority. The last channel to get service becomes the lowest priority channel with the others rotating accordingly. With Rotating Priority in a single chip DMA system, any device requesting service is guaranteed to be recognized after no more than three higher priority services have occurred. This prevents any one channel from monopolizing the system.



The priority encoder selects the highest priority channel requesting service on each active-going HACK edge. Once a channel is started, its operation will not be suspended if a request is received by a higher priority channel. The high priority channel will only gain control after the lower priority channel releases HREQ. When control is passed from one channel to another, the CPU will always gain bus control. This ensures generation of rising HACK edge to be used to initiate selection of the new highest-priority requesting channel.

Compressed Timing: In order to achieve even greater throughput where system characteristics permit, the Am9517A can compress the transfer time to two clock cycles. From Timing Diagram 3 it can be seen that state S3 is used to extend the access time of the read pulse. By removing state S3 the read pulse width is made equal to the write pulse width and a transfer consists only of state S2 to change the address and state S4 to perform the read/write. S1 states will still occur when A8-A15 need updating (see Address Generation). Timing for compressed transfers is found in Timing Diagram 6.

Address Generation: In order to reduce pin count, the Am9517A multiplexes the eight higher order address bits on the data lines. State S1 is used to output the higher order address bits to an external latch from which they may be placed on the address bus. The falling edge of Address Strobe (ADSTB) is used to load these bits from the data lines to the latch. Address Enable (AEN) is used to enable the bits onto the address bus through a 3-state enable. The lower order address bits are output by the Am9517A directly. Lines A0-A7 should be connected to the address bus. Timing Diagram 3 shows the time relationships between CLK, AEN, ADSTB, DB0-DB7 and A0-A7.

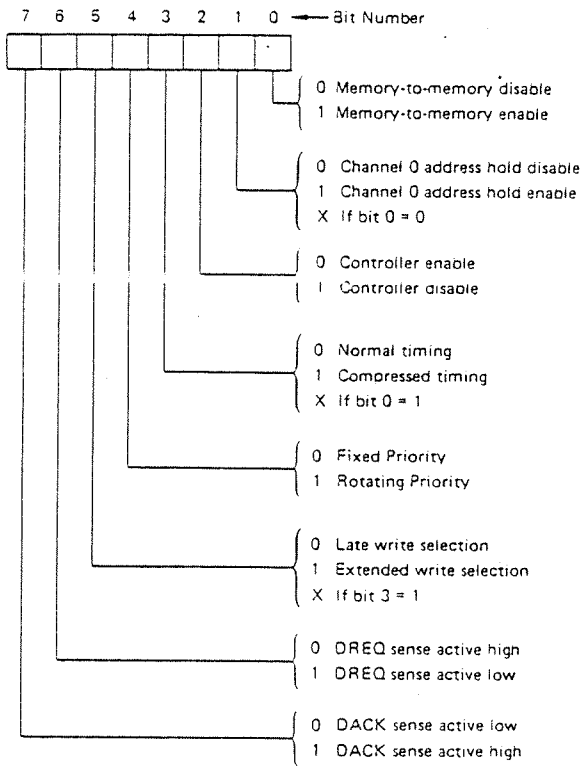
During Block and Demand Transfer mode services which include multiple transfers, the addresses generated will be sequential. For many transfers the data held in the external address latch will remain the same. This data need only change when a carry or borrow from A7 to A8 takes place in the normal sequence of addresses. To save time and speed transfers, the Am9517A executes S1 states only when updating of A8-A15 in the latch is necessary. This means for long services, S1 states may occur only once every 256 transfers, a savings of 255 clock cycles for each 256 transfers.

REGISTER DESCRIPTION

Current Address Register: Each channel has a 16-bit Current Address register. This register holds the value of the address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is written or read by the microprocessor in successive 8-bit bytes. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialization takes place only after an $\overline{EOP}$.

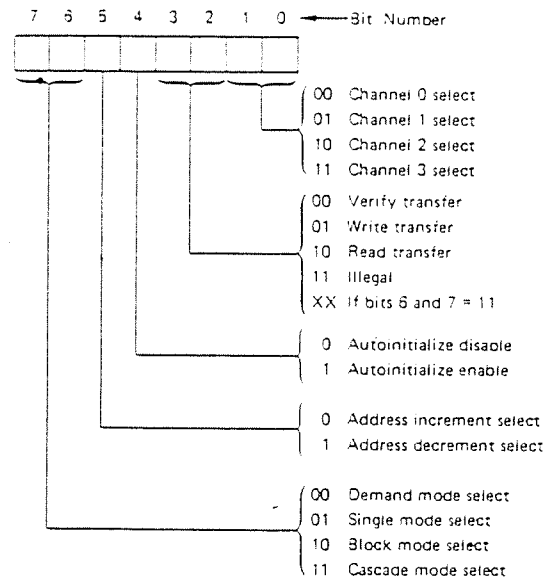
Current Word Count Register: Each channel has a 16-bit Current Word Count register. This register should be programmed with, and will return on a CPU read, a value one less than the number of words to be transferred. The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer. When the value in the register goes to zero, a TC will be generated. This register is loaded or read in successive 8-bit bytes by the microprocessor in the Program Condition. Following the end of a DMA service it may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize can occur only when an $\overline{EOP}$ occurs. Note that the contents of the Word Count register will be FFFF (hex) following on internally generated $\overline{EOP}$.

Command Register: This 8-bit register controls the operation of the Am9517A. It is programmed by the microprocessor in the Program Condition and is cleared by Reset. The following table lists the function of the command bits. See Figure 4 for address coding.

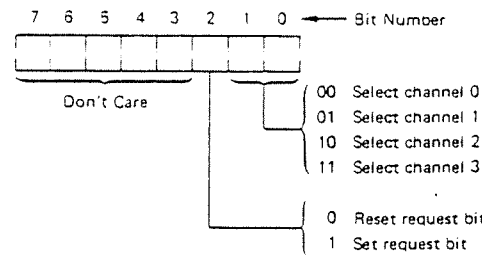


Base Address and Base Word Count Registers: Each channel has a pair of Base Address and Base Word Count registers. These 16-bit registers store the original values of their associated current registers. During Autoinitialize these values are used to restore the current registers to their original values. The base registers are written simultaneously with their corresponding current register in 8-bit bytes during DMA programming by the microprocessor. Accordingly, writing to these registers when intermediate values are in the Current registers will overwrite the intermediate values. The Base registers cannot be read by the microprocessor.

Mode Register: Each channel has a 6-bit Mode register associated with it. When the register is being written to by the microprocessor in the Program Condition, bits 0 and 1 determine which channel Mode register it to be written.

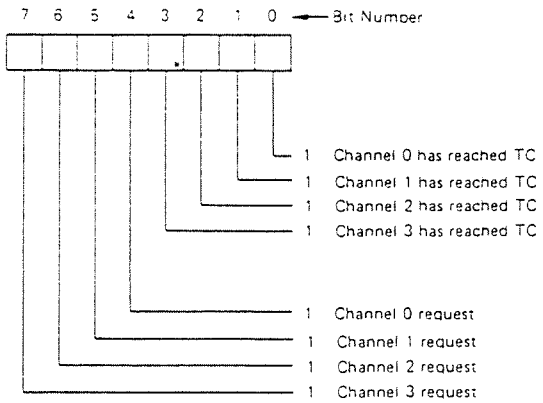


Request Register: The Am9517A can respond to requests for DMA service which are initiated by software as well as by a DREQ. Each channel has a request bit associated with it in the 4-bit Request register. These are nonmaskable and subject to prioritization by the Priority Encoder network. Each register bit is set or reset separately under software control or is cleared upon generation of a TC or external EOP. The entire register is cleared by a Reset. To set or reset a bit, the software loads the proper form of the data word. See Figure 4 for address coding.



Software requests will be serviced only if the channel is in Block mode. When initiating a memory-to-memory transfer, the software request for channel 0 should be set.

Status Register: The Status registers may be read out of the Am9517A by the microprocessor. It indicates which channels have reached a terminal count and which channels have pending DMA requests. Bits 0-3 are set each time a TC is reached by that channel, including after each Autoinitialization. These bits are cleared by Reset and each Status Read. Bits 4-7 are set whenever their corresponding channel is requesting service.



Temporary Register: The Temporary register is used to hold data during memory-to-memory transfers. Following the completion of the transfers, the last word moved can be read by the microprocessor in the Program Condition. The Temporary register always contains the last byte transferred in the previous memory-to-memory operation, unless cleared by a Reset.

Software Commands: There are two special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The two software commands are:

Clear First/Last Flip/Flop: This command may be issued prior to writing or reading Am9517A address or word count information. This initializes the flip/flop to a known state so that subsequent accesses to register contents by the microprocessor will address lower and upper bytes in the correct sequence.

Master Clear: This software instruction has the same effect as the hardware Reset. The Command, Status, Request, Temporary and Internal First/Last Flip/Flop registers are cleared and the Mask register is set. The Am9517A will enter the Idle cycle.

Figure 4 lists the address codes for the software commands.

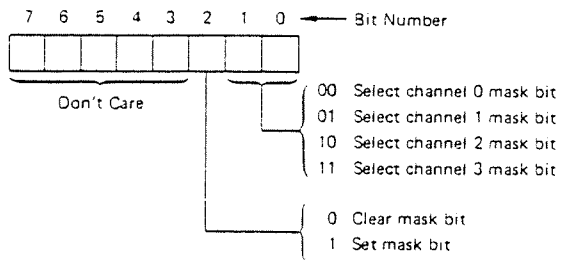
Interface Signals						
A3	A2	A1	A0	$\overline{IOR}$	$\overline{IOW}$	Operation
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip/Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Illegal
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

Figure 4. Register and Function Addressing.

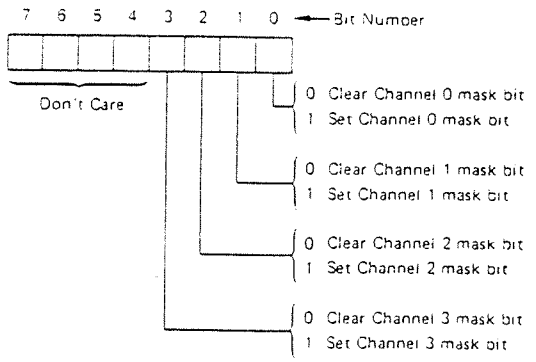
DMA Programming Example:

```
%       PROGRAM EXAMPLE SHOWING HOW INPUT DMA CHANNEL 0
%       CAN BE INITIALIZED TO TRANSFER 64 BYTES INTO
%       RAM, (FROM SERIAL CHANNEL 0) STARTING AT
%       ADDRESS 400B.

INTEGER  IBAL0 := 0          % BASE ADDRESS LOW BYTE
INTEGER  IBAH0 := 10         % BASE ADDRESS HIGH BYTE
INTEGER  IWCL0 := 100        % WORD COUNT LOW BYTE
INTEGER  IWCH0 := 0          % WORD COUNT HIGH BYTE
INTEGER  IMOD0 := 124        % MODE CHANNEL 0
INTEGER  ICOMM := 140        % COMMAND REGISTER DATA
INTEGER  IDMMR := 0          % CLEAR MASK CHANNEL 0

$*IDMIN: MOVE.B #0, 0EF0090H      % RESET
$*       MOVE.B IBAL0, 0EF0090H   % SET BASE ADDRESS L
$*       MOVE.B IBAH0, 0EF0090H   % SET BASE ADDRESS H
$*       MOVE.B IWCL0, 0EF0080H   % SET WORD COUNT L
$*       MOVE.B IWCH0, 0EF0080H   % SET WORD COUNT H
$*       MOVE.B IMOD0, 0EF008AH   % SET MODE REG. 0
$*       MOVE.B ICOMM, 0EF0098H   % Set COMMAND REG.
$*       MOVE.B IDMR0, 0EF009AH   % CLEAR MASK
$*       RTS
```

Mask Register: Each channel has associated with it a mask bit which can be set to disable the incoming DREQ. Each mask bit is set when its associated channel produces an EOP if the channel is not programmed for Autoinitialize. Each bit of the 4-bit Mask register may also be set or cleared separately under software control. The entire register is also set by a Reset. This disables all DMA requests until a clear Mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar in form to that used with the Request register. See Figure 4 for instruction addressing.



All four bits of the Mask Register may also be written with a single command.



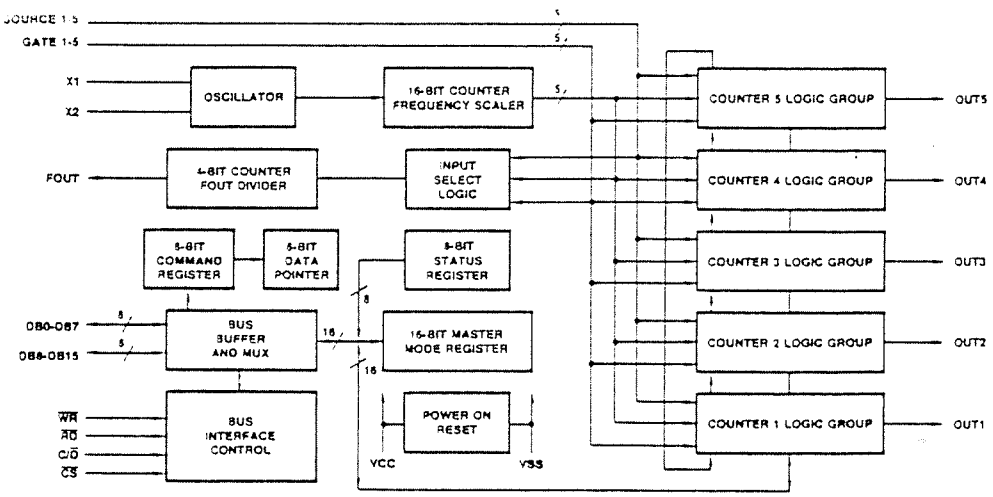
A P P E N D I X C

TIMING CONTROLLER

The information in the appendices is taken from AM9500 Peripheral Products Interface Guide issued by Advanced Micro Devices and from the SIO manual issued by Zilog (UK) Limited.
Z80 is a Zilog trademark.

The Am9513 System Timing Controller is an LSI circuit designed to service many types of courting, sequencing and timing applications. It provides the capability for programmable frequency synthesis, high resolution programmable duty cycle waveforms, retriggerable digital timing functions, time of day clocking, coincidence alarms, complex pulse generation, high resolution baud rate generation, frequency shift keying, stop-watch timing, even count accumulation, waveform analysis and many more.
A variety of programmable operating modes and control features allow the Am9513 to be personalized for particular applications as well as dynamically reconfigured under program control.

The STC includes five general-purpose 16-bit counters. A variety of internal frequency sources and external pins may be selected as inputs for individual counters with software selectable active-high or active-low input polarity. Both hardware and software gating of each counter is available.
Three-state outputs for each counter provide either pulses or levels. The counters can be programmed to count up or down in either binary or BCD. The accumulated count may be read without disturbing the counting process. Any of the counters may be internally concatenated to form an effective counter length of up to 80 bits.



Timing Controller Block Diagram

Usage in PIOC:

The STC must be programmed according to the PIOC hardware configuration. Channel 1 is connected to the interrupt controller, which will give an interrupt on level 3.5 at high to low transition. This channel should therefore be initialized to count down repeatedly with an active low output pulse.

STC channels 2 to 5 are used as baud rate generators for PIOC channels 0 to 3. They should have a duty cycle of 50% and are set to count down repeatedly at twice the desired frequency with the output flip-flop toggling each time zero count is reached.

The desired frequency is equal to the desired baud rate for synchronous mode.
When asynchronous mode, the frequency will be 16, 32 or 64 times the baud rate, depending on how the serial controller is set up.

FUNCTIONAL DESCRIPTION

The Am9513 block diagrams (Figures 2, 3 and 4) indicate the interface signals and the basic flow of information. Internal control lines and the internal data bus have been omitted. The control and data registers are all connected to a common internal 16-bit bus. The external bus may be 8 or 16 bits wide; in the 8-bit mode the internal 16-bit information is multiplexed to the low order data bus pins DB0 through DB7.

An internal oscillator provides a convenient source of frequencies for use as counter inputs. The oscillator's frequency is controlled at the X1 and X2 interface pins by an external reactive network such as a crystal. The oscillator output is divided by the Frequency Scaler to provide several sub-frequencies. One of the scaled frequencies (or one of ten input signals) may be selected as an input to the FOUT divider and then comes out of the chip at the FOUT interface pin.

The STC is addressed by the external system as two locations: a control port and a data port. The control port provides direct access to the Status and Command registers, as well as allowing the user to update the Data Pointer register. The data port is used to communicate with all other addressable internal locations. The Data Pointer register controls the data port addressing.

Among the registers accessible through the data port are the Master Mode register and five Counter Mode registers, one for each counter. The Master Mode register controls the programmable options that are not controlled by the Counter Mode registers.

Each of the five general-purpose counters is 16 bits long and is independently controlled by its Counter Mode register. Through this register, a user can software select one of 16 sources as the counter input, a variety of gating and repetition modes, up or down counting in binary or BCD and active-high or active-low input and output polarities.

Associated with each counter are a Load register and a Hold register, both accessible through the data port. The Load register is used to automatically reload the counter to any predefined value, thus controlling the effective count period. The Hold register is used to save count values without disturbing the count process, permitting the host processor to read intermediate counts. In addition, the Hold register may be used as a second Load register to generate a number of complex output waveforms.

All five counters have the same basic control logic and control registers. Counters 1 and 2 have additional Alarm registers and comparators associated with them, plus the extra logic necessary for operating in a 24-hour time-of-day mode. For real-time operation the time-of-day logic will accept 50Hz, 60Hz or 100Hz input frequencies.

Each general counter has a single dedicated output pin. It may be turned off when the output is not of interest or may be configured in a variety of ways to drive interrupt controllers, Darlington buffers, bus drivers, etc. The counter inputs, on the other hand, are specifically not dedicated to any given interface line. Considerable versatility is available for configuring both the input and the gating of individual counters. This not only permits dynamic reassignment of inputs under software control, but also allows multiple counters to use a single input, and allows a single gate pin to control more than one counter. Indeed, a single pin can be the gate for one counter and, at the same time, the count source for another.

A powerful command structure simplifies user interaction with the counters. A counter must be armed by one of the ARM commands before counting can commence. Once armed, the counting process may be further enabled or disabled using the hardware gating facilities. The ARM and DISARM commands permit software gating of the count process in some modes.

The LOAD command causes the counter to be reloaded with the value in either the associated Load register or the associated Hold register. It will often be used as a software retrigger or as counter initialization prior to active hardware gating.

The DISARM command disables further counting independent of any hardware gating. A disarmed counter may be reloaded using the LOAD command, may be incremented or decremented using the STEP command and may be read using the SAVE command. A count process may be resumed using an ARM command.

The SAVE command transfers the contents of a counter to its associated Hold register. This command will overwrite any previous Hold register contents. The SAVE command is designed to allow an accumulated count to be preserved so that it can be read by the host CPU at some later time.

Two combinations of the basic commands exist to either LOAD AND ARM or to DISARM AND SAVE any combination of counters. Additional commands are provided to: step an individual counter by one count; set and clear an output toggle; issue a software reset; clear and set special bits in the Master Mode register; and load the Data Pointer register.

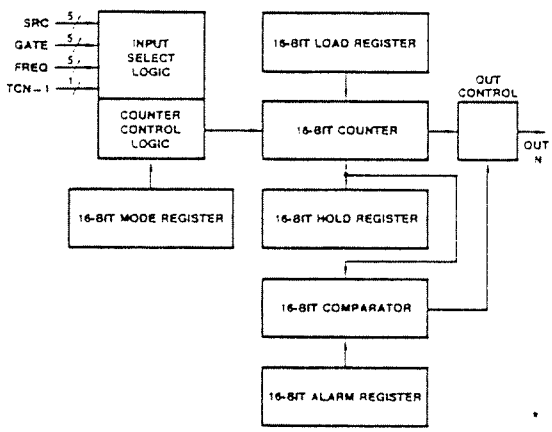


Figure 3. Counter Logic Groups 1 and 2.

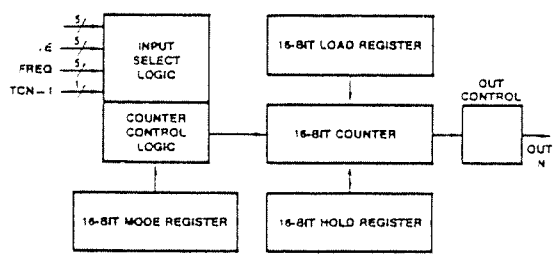


Figure 4. Counter Logic Groups 3, 4 and 5.

Command Code								Command Description
C7	C6	C5	C4	C3	C2	C1	C0	
0	0	0	E2	E1	G4	G2	G1	Load Data Pointer register with contents of E and G fields. (G ≠ 000, G ≠ 110)
0	0	1	S5	S4	S3	S2	S1	Arm counting for all selected counters
0	1	0	S5	S4	S3	S2	S1	Load contents of specified source into all selected counters
0	1	1	S5	S4	S3	S2	S1	Load and Arm all selected counters
1	0	0	S5	S4	S3	S2	S1	Disarm and Save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters in hold register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	1	N4	N2	N1	Set output bit N (001 ≤ N ≤ 101)
1	1	1	0	0	N4	N2	N1	Clear output bit N (001 ≤ N ≤ 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 ≤ N ≤ 101)
1	1	1	0	1	0	0	0	Set MM14 (Disable Data Pointer Sequencing)
1	1	1	0	1	1	1	0	Set MM12 (Gate off FOUT)
1	1	1	0	1	1	1	1	Set MM13 (Enter 16-bit bus mode)
1	1	1	0	0	0	0	0	Clear MM14 (Enable Data Pointer Sequencing)
1	1	1	0	0	1	1	0	Clear MM12 (Gate on FOUT)
1	1	1	0	0	1	1	1	Clear MM13 (Enter 8-bit bus mode)
1	1	1	1	1	1	1	1	Master reset

Figure 21. Am9513 Command Summary.

CONTROL PORT REGISTERS

The STC is addressed by the external system as only two locations: a Control port and a Data port. Transfers at the Control port (C/$\overline{D}$ = High) allow direct access to the command register when writing and the status register when reading. All other available internal locations are accessed for both reading and writing via the Data port (C/$\overline{D}$ = Low). Data port transfers are executed to and from the location currently addressed by the Data Pointer register. Options available in the Master Mode register and the Data Pointer control structure allow several types of transfer sequencing to be used. See Figure 7.

Transfers to and from the control port are always 8 bits wide. Each access to the Control port will transfer data between the Command register (writes) or Status register (reads) and Data Bus pins DB0-DB7, regardless of whether the Am9513 is in 8- or 16-bit bus mode. When the Am9513 is in 8-bit bus mode, Data Bus pins DB13-DB15 should be held at a logic high whenever $\overline{CS}$ and $\overline{WR}$ are both active.

Command Register

The Command register provides direct control over each of the five general counters and controls access through the Data port by allowing the user to update the Data Pointer register. The "Command Description" section of this data sheet explains the detailed operation of each command. A summary of all commands appears in Figure 21. Six of the command types are used for direct software control of the counting process. Each of these six commands contains a 5-bit S field. In a linear-select fashion, each bit in the S field corresponds to one of the five general counters (S1 = Counter 1, S2 = Counter 2, etc.). When an S bit is a one, the specified operation is performed on the counter so designated; when an S bit is a zero, no operation occurs for the corresponding counter.

Data Pointer Register

The 6-bit Data Pointer register is loaded by issuing the appropriate command through the control port to the Command register. As shown in Figure 7, the contents of the Data Pointer register are used to control the Data Port multiplexer, selecting which internal register is to be accessible through the Data Port.

The Data Pointer consists of a 3-bit Group Pointer, a 2-bit Element Pointer and a 1-bit Byte Pointer, depicted in Figure 8. The Byte Pointer bit indicates which byte of a 16-bit register is to be transferred on the next access through the data port. Whenever the Data Pointer is loaded, the Byte Pointer bit is set to one, indicating a least-significant byte is expected. The Byte Pointer toggles following each 8-bit data transfer with an 8-bit data bus (MM13 = 0), or it always remains set with the 16-bit data bus option (MM13 = 1). The Element and Group pointers are used to select which internal register is to be accessible through the Data Port. Although the contents of the Element and Group Pointer in the Data Pointer register cannot be read by the host processor, the Byte Pointer is available as a bit in the Status register.

Random access to any available internal data location can be accomplished by simply loading the Data Pointer using the command shown in Figure 9 and then initiating a data read or data write. This procedure can be used at any time, regardless of the setting of the Data Pointer Control bit (MM14). When the 8-bit data bus configuration is being used (MM13 = 0), two bytes of data would normally be transferred following the issuing of the "Load Data Pointer" command.

To permit the host processor to rapidly access the various internal registers, automatic sequencing of the Data Pointer is provided. Sequencing is enabled by clearing Master Mode bit 14 (MM14) to zero. As shown in Figure 10, several types of sequencing are available depending on the data bus width being used and the initial Data Pointer value entered by command.

When E1 = 0 or E2 = 0 and G4, G2, G1 point to a Counter Group, the Data Pointer will proceed through the Element cycle. The Element field will automatically sequence through the three values 00, 01 and 10 starting with the value entered. When the transition from 10 to 00 occurs, the Group field will also be incremented by one. Note that the Element field in this case does not sequence to a value of 11. The Group field circulates only within the five Counter Group codes.

If E2, E1 = 11 and a Counter Group is selected, then only the Group field is sequenced. This is the Hold cycle. It allows the Hold registers to be sequentially accessed while bypassing the Mode and Load registers. The third type of sequencing is the Control

cycle. If G4, G2, G1 = 111 and E2, E1 ≠ 11, the Element Pointer will be incremented through the values 00, 01 and 10, with no change to the Group Pointer.

When G4, G2, G1 = 111 and E2, E1 = 11, no incrementing takes place and only the Status register will be available through the data port. Note that the Status register can also always be read directly through the Control port.

For all of these auto-sequence modes, if an 8-bit data bus is used, the Byte pointer will toggle after every data transfer to allow the least and most significant bytes to be transferred before the Element or Group Fields are incremented.

Prefetch Circuit

In order to minimize the read access time to internal Am9513 registers, a prefetch circuit is used for all read operations through the Data Port. Following each read or write operation through the Data Port, the Data Pointer register is updated to point to the next register to be accessed. Immediately following this update, the new register data is transferred to a special prefetch latch at the interface pad logic. When the user performs a subsequent read of the Data Port, the data bus drivers are enabled, outputting the prefetched data on the bus. Since the internal data register is accessed prior to the start of the read operation, its access time is transparent to the user. In order to keep the prefetched data consistent with the data pointer, prefetches are also performed after each write to the Data Port and after execution at the "Load Data Pointer" command. The following rules should be kept in mind regarding Data Port Transfers.

1. The Data Pointer register should always be reloaded before reading from the Data Port if a command other than "Load Data Pointer" was issued to the Am9513 following the last Data Port read or write. The Data Pointer does not have to be loaded again if the first Data Port transaction after a command entry is a write, since the Data Port write will automatically cause a new prefetch to occur.
2. Operating modes N, O, Q and R allow the user to save the counter contents in the Hold register by applying an active-going gate edge. If the Data Pointer register had been pointing to the Hold register in question, the prefetched value will not correspond to the new value saved in the Hold register. To avoid reading an incorrect value, a new "Load Data Pointer" command should be issued before attempting to read the saved data. A Data Port write (to another register) will also initiate a prefetch; subsequent reads will access the recently saved Hold register data. Many systems will use the "saving" gate edge to interrupt the host CPU. In systems such as this the interrupt service routine should issue a "Load Data Pointer" command prior to reading the saved data.

Status Register

The 8-bit read-only Status register indicates the state of the Byte Pointer bit in the Data Pointer register and the state of the OUT signal for each of the general counters. See Figures 11 and 19. The OUT signals reported are those internal to the chip after the polarity-select logic and just before the 3-state interface buffer circuitry.

The Status register OUT bit reflects an active-high or active-low TC output, or a TC Toggled output, as programmed in the Output Control Field of the Counter Mode register. That is, it reflects the exact state of the OUT pin. When the Low Impedance to Ground Output option (CM2-CM0 = 000) is selected, the Status register will reflect an active-high TC Output. When a High Impedance Output option (CM2-CM0 = 100) is selected, the Status register will reflect an active-low TC output.

For Counters 1 and 2, the OUT pin will reflect the comparator output if the comparators are enabled. The Status register bit and OUT pin are active high if CM2 = 0 and active-low if CM2 = 1. When the High Impedance option is selected and the comparator is enabled, the status register bit will reflect an active-high comparator output. When the Low Impedance to Ground option is selected and the comparator is enabled, the status register bit will reflect an active-low comparator output.

The Status register is normally accessed by reading the control port (see Figure 7) but may also be read via the data port as part of the Control Group.

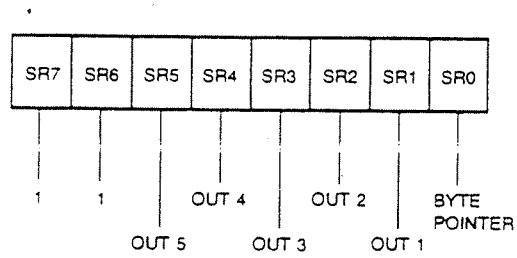


Figure 11. Status Register Bit Assignments.

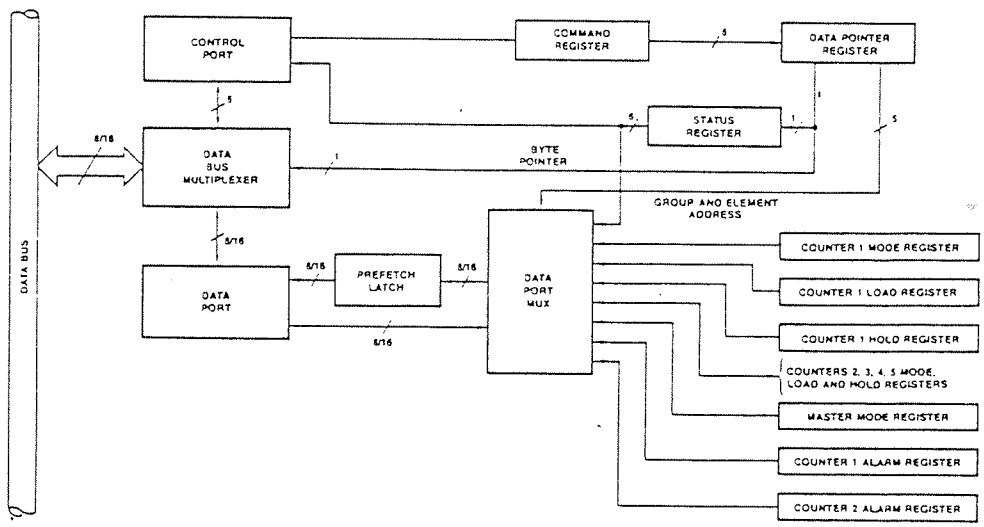


Figure 7. Am9513 Register Access.

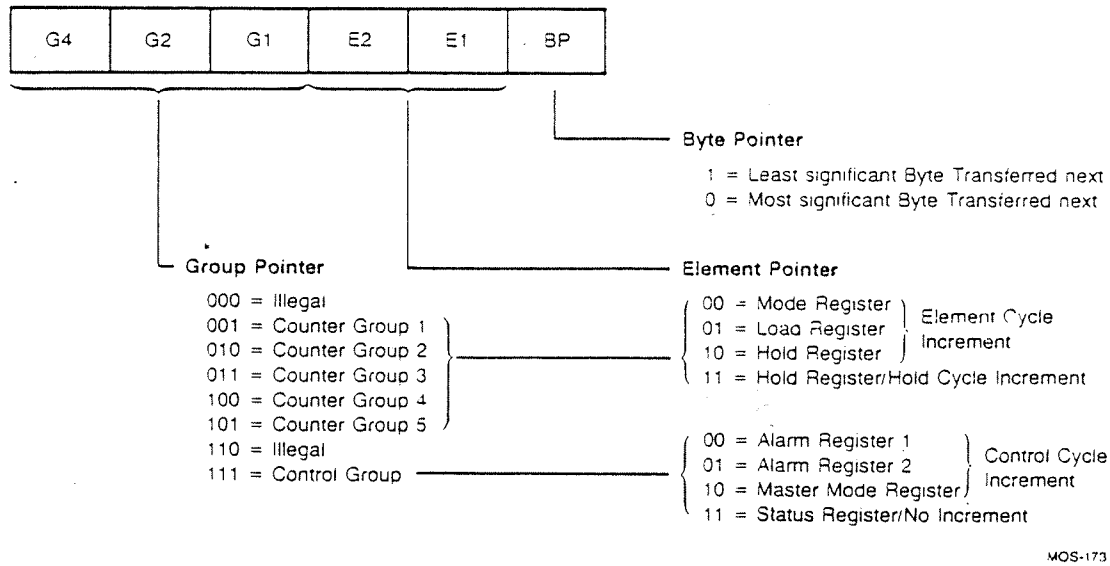


Figure 8. Data Pointer Register.

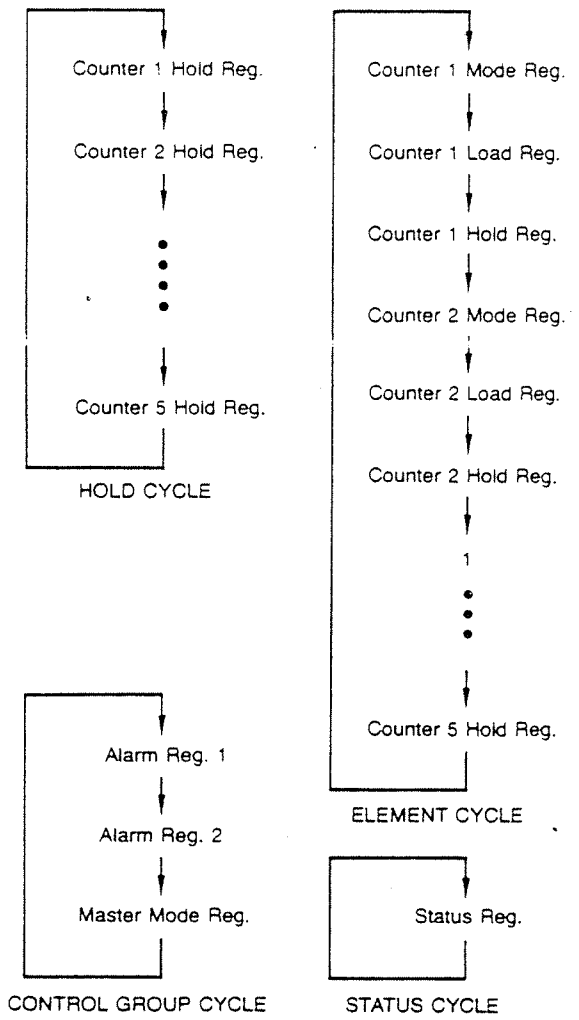


Figure 10. Data Pointer Sequencing.

	Element Cycle			Hold Cycle
	Mode Register	Load Register	Hold Register	Hold Register
Counter 1	FF01	FF09	FF11	FF19
Counter 2	FF02	FF0A	FF12	FF1A
Counter 3	FF03	FF0B	FF13	FF1B
Counter 4	FF04	FF0C	FF14	FF1C
Counter 5	FF05	FF0D	FF15	FF1D
Master Mode Register = FF17 Alarm 1 Register = FF07 Alarm 2 Register = FF0F				

Notes:
1. All codes are in hex.
2. When used with an 8-bit bus, only the two low order hex digits should be written to the command port; the 'FF' prefix should be used only for a 16-bit data bus interface.

Figure 9. Load Data Pointer Commands.

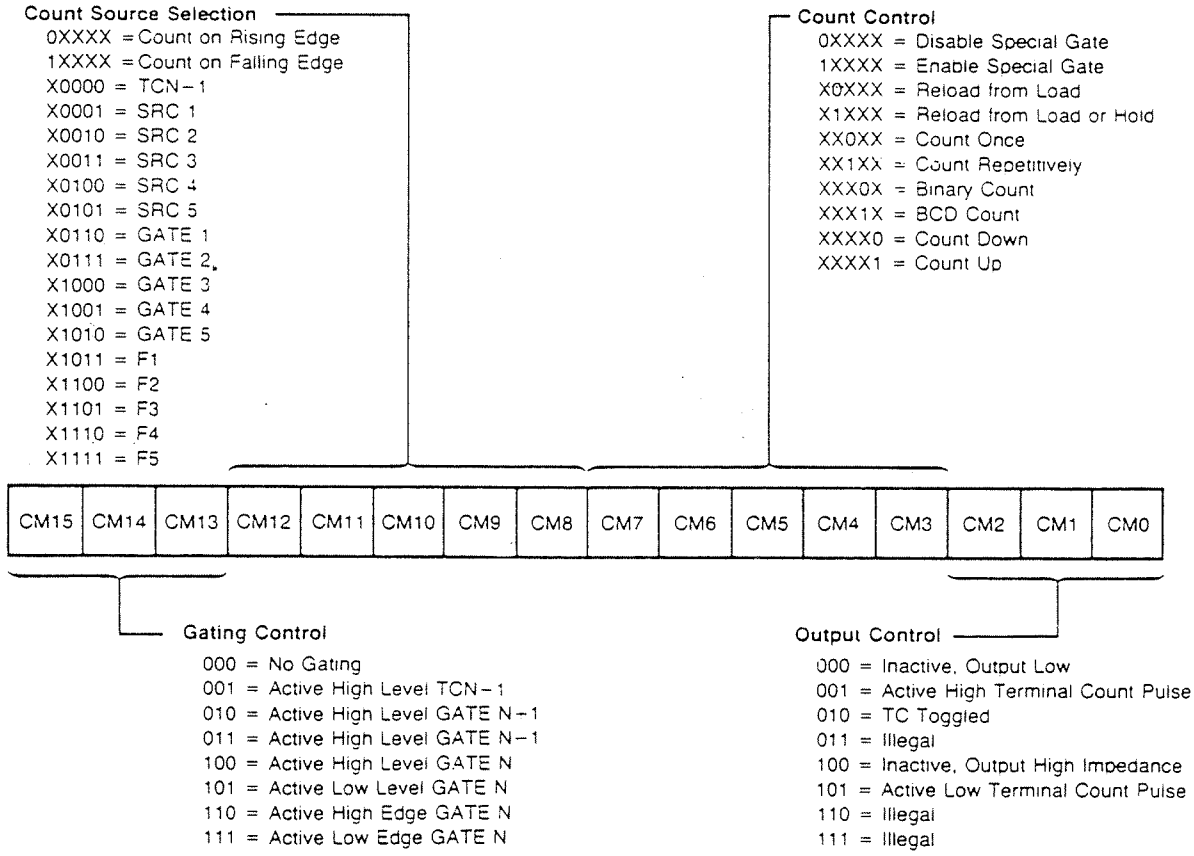


Figure 18. Counter Mode Register Bit Assignments.

DATA PORT REGISTERS

Counter Logic Groups

As shown in Figures 3 and 4, each of the five Counter Logic Groups consists of a 16-bit general counter with associated control and output logic, a 16-bit Load register, a 16-bit Hold register and a 16-bit Mode register. In addition, Counter Groups 1 and 2 also include 16-bit Comparators and 16-bit Alarm registers. The comparator/alarm functions are controlled by the Master Mode register. The operation of the Counter Mode registers is the same for all five counters. The host CPU has both read and write access to all registers in the Counter Logic Groups through the data port. The counter itself is never directly accessed.

The 16-bit read/write Load register is used to control the effective period of the general counter. Any 16-bit value may be written into the Load register. That value can then be transferred into the counter each time that Terminal Count (TC) occurs. "Terminal Count" is defined as that period of time when the counter contents would have been zero if an external value had not been transferred into the counter. Thus the terminal count frequency can be the input frequency divided by the value in the Load register. In all operating modes the contents of either Load or Hold register will be transferred into the counter when TC occurs. In cases where values are being accumulated in the counter, the Load register action can be transparent by filling the Load register with all zeros.

The 16-bit read/write Hold register is dual-purpose. It can be used in the same way as the Load register, thus offering an alternate source for modulo definition for the counter. The Hold register may also be used to store accumulated counter values for later transfer to the host processor. This allows the count to be sampled while the counting process proceeds. Transfer of the counter contents into the Hold register is accomplished by the hardware interface in some operating modes or by the software SAVE command at any time.

Counter Mode Register

The 16-bit read/write Counter Mode register controls the gating, counting, output and source select functions within each Counter Logic Group. The "Counter Mode Control Options" section of this data sheet describes the detailed control options available. Figure 18 shows the bit assignments for the Counter Mode registers.

Alarm Registers and Comparators

Added functions are available in the Counter Logic Groups for Counters 1 and 2 (see Figure 3). Each contains a 16-bit Alarm register and a 16-bit Comparator. When the value in the counter reaches the value in the Alarm register, the Comparator output will go true. The Master Mode register contains control bits to individually enable/disable the comparators. When enabled, the comparator output appears on the OUT pin of the associated counter in place of the normal counter output. The output will remain true as long as the comparison is true, that is, until the next input causes the count to change. The polarity of the Comparator output will be active-high if the Output Control field of the Counter Mode register is 001 or 010 and active-low if the Output Control field is 101.

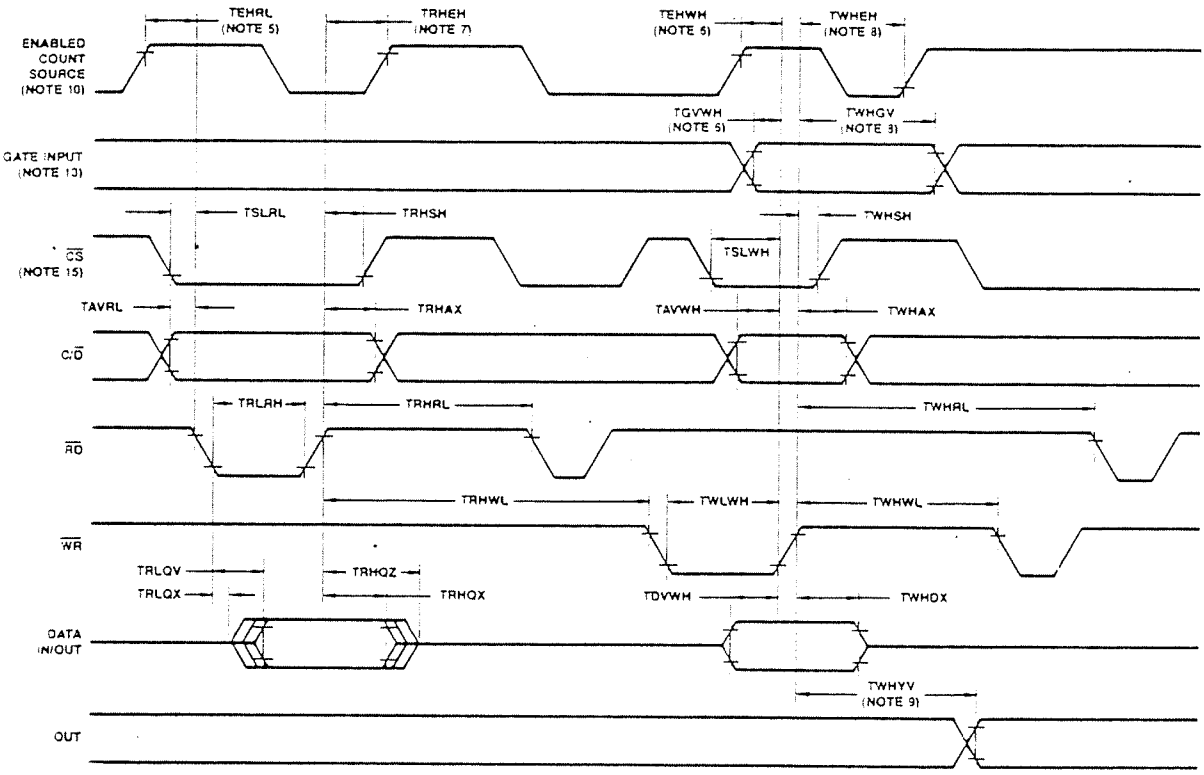


Figure 23. Bus Transfer Switching Waveforms.

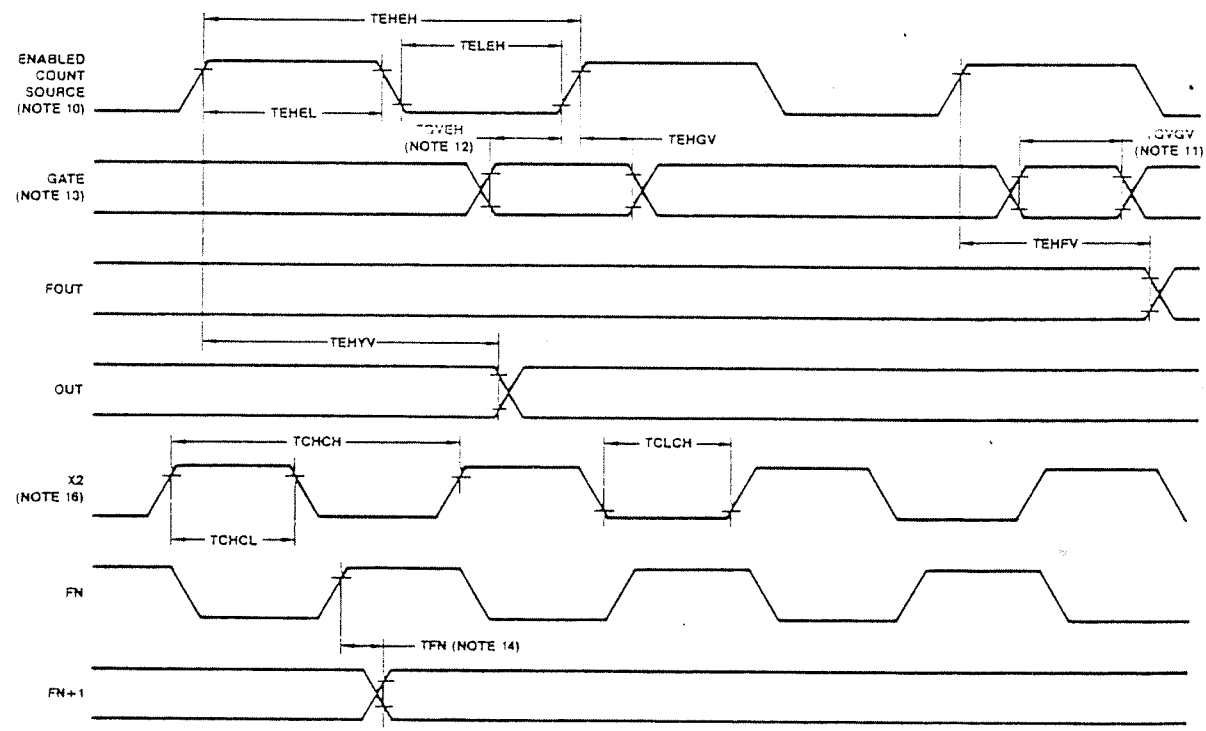


Figure 24. Counter Switching Waveforms.

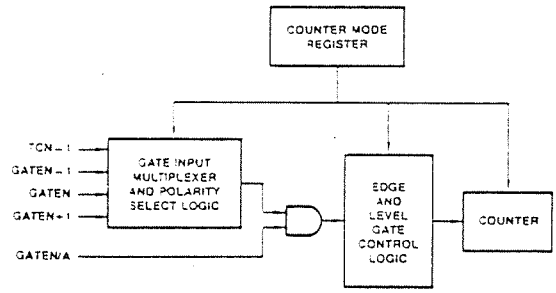


Figure 15. Gating Control.

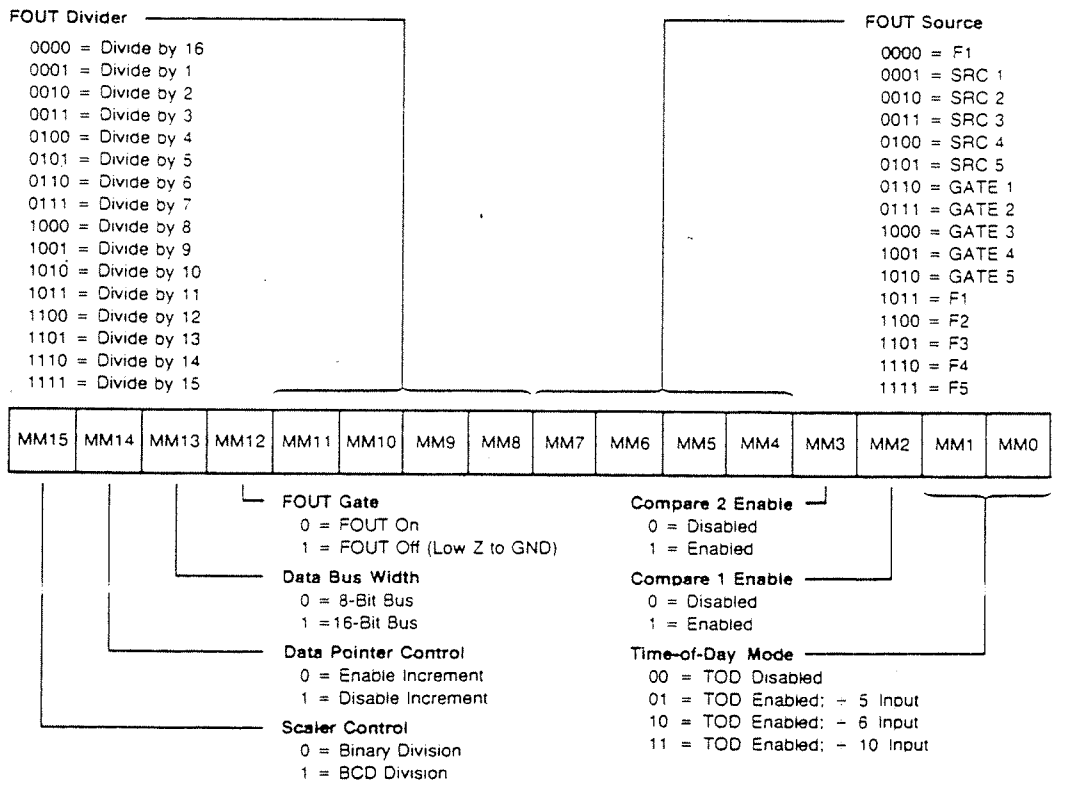


Figure 13. Master Mode Register Bit Assignments.

MASTER MODE CONTROL OPTIONS

The 16-bit Master Mode (MM) register is used to control those internal activities that are not controlled by the individual Counter Mode registers. This includes frequency control, time-of-day operation, comparator controls, data bus width and data pointer sequencing. Figure 13 shows the bit assignments for the Master Mode register. This section describes the use of each control field.

Master Mode register bits MM12, MM13 and MM14 can be individually set and reset using commands issued to the Command register. In addition they can all be changed by writing directly to the Master Mode register.

After power-on reset or a Master Reset command, the Master Mode register is cleared to an all zero condition. This results in the following configuration:

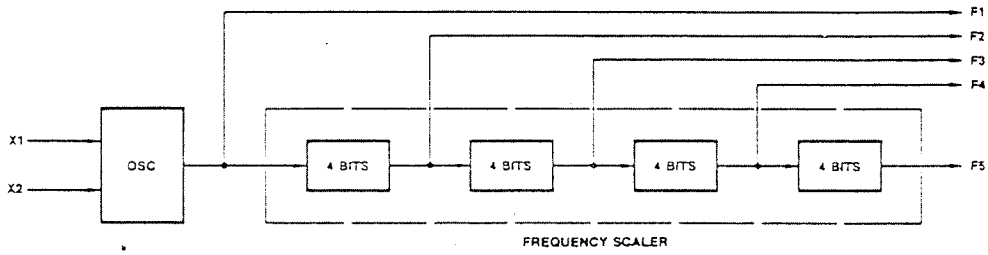
Time-of-day disabled
Both Comparators disabled
FOUT Source is frequency F1
FOUT Divider set for divide-by-16
FOUT gated on
Data Bus 8 bits wide
Data Pointer Sequencing enabled
Frequency Scaler divides in binary

Operating Mode	A	B	C	D	E	F	G	H	I	J	K	L
Special Gate (CM7)	0	0	0	0	0	0	0	0	0	0	0	0
Reload Source (CM6)	0	0	0	0	0	0	1	1	1	1	1	1
Repetition (CM5)	0	0	0	1	1	1	0	0	0	1	1	1
Gate Control (CM15-CM13)	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE
Count to TC once, then disarm	X	X	X									
Count to TC twice, then disarm							X	X	X			
Count to TC repeatedly				X	X	X				X	X	X
Gate input does not gate counter input	X			X			X			X		
Count only during active gate level		X			X			X			X	
Start count on active gate edge and stop count on next TC.			X			X						
Start count on active gate edge and stop count on second TC.									X			X
No hardware retriggering	X	X	X	X	X	X	X	X	X	X	X	X
Reload counter from Load Register on TC	X	X	X	X	X	X						
Reload counter on each TC, alternating reload source between Load and Hold Registers.							X	X	X	X	X	X
Transfer Load Register into counter on each TC that gate is LOW; transfer Hold Register into counter on each TC that gate is HIGH.												
On active gate edge transfer counter into Hold Register and then reload counter from Load Register.												

Operating Mode	M	N	O	P	Q	R	S	T	U	V	W	X
Special Gate (CM7)	1	1	1	1	1	1	1	1	1	1	1	1
Reload Source (CM6)	0	0	0	0	0	0	1	1	1	1	1	1
Repetition (CM5)	0	0	0	1	1	1	0	0	0	1	1	1
Gate Control (CM15-CM13)	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE
Count to TC once, then disarm		X	X									
Count to TC twice, then disarm							X					
Count to TC repeatedly					X	X				X		
Gate input does not gate counter input							X			X		
Count only during active gate level		X			X							
Start count on active gate edge and stop count on next TC.			X			X						
Start count on active gate edge and stop count on second TC.												
No hardware retriggering							X			X		
Reload counter from Load Register on TC		X	X		X	X						
Reload counter on each TC, alternating reload source between Load and Hold Registers.												
Transfer Load Register into counter on each TC that gate is LOW; transfer Hold Register into counter on each TC that gate is HIGH.							X			X		
On active gate edge transfer counter into Hold Register and then reload counter from Load Register.		X	X		X	X						

Note: Operating modes M, P, T, U, W and X are reserved and should not be used.

Figure 17. Counter Control Interaction.



Frequency	BCD Scaling MM15 = 1	Binary Scaling MM15 = 0
F1	OSC	OSC
F2	F1 ÷ 10	F1 ÷ 16
F3	F1 ÷ 100	F1 ÷ 256
F4	F1 ÷ 1,000	F1 ÷ 4,096
F5	F1 ÷ 10,000	F1 ÷ 65,536

Figure 16. Frequency Scaler Ratios.

Time-of-Day

Bits MM0 and MM1 of the Master Mode register specify the time-of-day (TOD) options. When MM0 = 0 and MM1 = 0, the special logic used to implement TOD is disabled and Counters 1 and 2 will operate in exactly the same way as Counters 3, 4 and 5. When MM0 = 1 or MM1 = 1, additional counter decoding and control logic is enabled on Counters 1 and 2 which causes their decades to turn over at the counts that generate appropriate 24-hour TOD accumulations.

Figure 14 shows the counter configurations for TOD operation. The two most significant decades of Counter 2 contain the "hours" digits and they can hold a maximum count of 23 hours. The two least significant decades of Counter 2 indicate "minutes" and will hold values up to 59. The three most significant decades of Counter 1 indicate "seconds" and will contain values up to 59.9. The least significant decade of Counter 1 is used to scale the input frequency in order to output tenth-of-second periods into the next decade. It can be set up to divide-by-five (MM0 = 1, MM1 = 0), divide-by-six (MM0 = 0, MM1 = 1), or divide-by-ten (MM0 = 1, MM1 = 1). The input frequency, therefore, for real-time clocking can be, respectively, 50Hz, 60Hz or 100Hz. With divide-by-ten specified and with 100Hz input, the least significant decade of Counter 1 accumulates time in hundredths of seconds (tens of milliseconds). For accelerated time applications other input frequencies may be useful.

The input for Counter 2 should be the TC output of Counter 1, connected either internally or externally, for TOD operation. Both counters should be set up for BCD counting. The Load registers should be used to initialize the counters to the proper time. Either count up or count down may be used.

To read the time, a SAVE command should be issued to Counters 1 and 2. Because counts ripple between the counters, the possibility exists of the SAVE command occurring after Counter 1 has counted but before Counter 2 has. This would result in an incorrectly saved time. To guard against this, Counter 2 should be resaved if Counter 1's saved value indicates a ripple carry/borrow may have been generated. In other words, Counter 2 should be resaved if the value saved from Counter 1 is 0 (up counting), 59.94 (down counting, MM1-MM0 = 01), 59.95 (down counting, MM1-MM0 = 10), or 59.99 (down counting, MM1-MM0 = 11). By the time this test is performed and Counter 2 is resaved, any rippling carry/borrow will have updated Counter 2.

Comparator Enable

Bits MM2 and MM3 control the Comparators associated with Counter 1 and 2. When a Comparator is enabled, its output is substituted for the normal counter output on the associated OUT1 or OUT2 pin. The comparator output will be active-high if the output control field of the Counter Mode register is 001 or 010 and active low for a code of 101. Once the compare output is true, it will remain so until the count changes and the comparison therefore goes false.

The two Comparators can always be used individually in any operating mode. One special case occurs when the time-of-day option is invoked and both Comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit compare must be true in order to generate a true signal on OUT2. OUT1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm 1 and Counter 1.

FOUT Source

Master Mode bits MM4 through MM7 specify the source input for the FOUT divider. Fifteen inputs are available for selection and they include the five Source pins, the five Gate pins and the five internal frequencies derived from the oscillator. The 16th combination of the four control bits (all zeros) is used to assure that an active frequency is available at the input to the FOUT divider following reset.

FOUT Divider

Bits MM8 throught MM11 specify the dividing ratio for the FOUT Divider. The FOUT source (selected by bits MM4 through MM7) is divided by an integer value between 1 and 16, inclusive, and is then passed to the FOUT output buffer. After power-on or reset, the FOUT divider is set to divide-by-16.

FOUT Gate

Master Mode bit MM12 provides a software gating capability for the FOUT signal. When MM12 = 1, FOUT is off and in a low impedance state to ground. MM12 may be set or cleared in conjunction with the loading of the other bits in the Master Mode register; alternatively, there are commands that allow MM12 to be individually set or cleared directly without changing any other Master Mode bits. After power-up or reset, FOUT is gated on. When changing the FOUT divider ratio or FOUT source, transient pulses as short as half the period of the FOUT source may appear on the FOUT pin. Turning the FOUT gate on or off can also generate a transient. This should be considered when using FOUT as a system clock source.

Timer Initialization Example:

```
%       TIMER INITIALISATION:
%       TIMCOM WILL INITIALIZE THE REAL TIME CLOCK
%       GENERATOR WITH A FREQUENCY OF 5 Hz, AND THE
%       BAUD RATE GENERATORS TO 16 * 9600 BAUD.

%       TIMER PARAMETERS
%
CONSTANT      STCD            = EF0100H
CONSTANT      STCC            = EF0102H
%
INTEGER ARRAY : TIPART:=(&
                  5445B    ,&   % MODE 1 (COUNTER 1 PARAMETERS)
                  170000B  ,&   % LOAD 1 ( 4915200/5*16)B
                  0        ,&   % HOLD 1 (DUMMY)
                  6047B    ,&   % MODE 2 (COUNTER 2 PARAMETERS)
                  20B      ,&   % LOAD 2 ( 4915200/16*9600)B
                  0        ,&   % HOLD 2
                  6047B    ,&   % MODE 3
                  20B      ,&   % HOLD 3
                  0        ,&   % HOLD 3
                  6047B    ,&   % MODE 4
                  20B      ,&   % LOAD 4
                  0        ,&   % HOLD 4
                  6047B    ,&   % MODE 5
                  20B      ,&   % LOAD 5
                  0        ,&   % HOLD 5
        ROUTINE SPECIAL VOID,VOID:TIMCOM

LABEL : TICO1

$*          MOVE.L #STCD,A0
$*          MOVE.L #STCC,A1
$*          MOVE.W #0FFFFH,(A1)      % RESET STC
$*          MOVE.W #0FFE7H,(A1)      % 16 BITS BUS
$*          MOVE.W #0FFF0H,(A1)      % AUTOMATIC SEQUENCING
$*          MOVE.W #0FFF1H,(A1)      % POINTER TO 1.DATA ELEMENT
$*          MOVE.W #0006H,D0         % SET TABLE LENGTH
$*          LEA TIPART,A4            % POINT TO TABLETOP
TICO1: $*   MOVE.W (A4)+,(A0)       % LOAD REGISTER CONTENTS
$*          DBEQ D0,TICO1            % NEXT REG.
$*          MOVE.W #0FF7FH,(A1)      % LOAD AND ARM COUNTERS
$*          RTS

        ENDROUTINE
```

A P P E N D I X D

SERIAL CONTROLLERS

The information in the appendices is taken from AM9500 Peripheral Products Interface Guide issued by Advanced Micro Devices and from the SIO manual issued by Zilog (UK) Limited.
Z80 is a Zilog trademark.

The Z80-SIO (Serial Input/Output) is a dual channel multi-function peripheral component designed to satisfy a wide variety of serial data communication requirements in microcomputer systems.
Its basic function is a serial to parallel, parallel to serial converter/controller, but within that role it is configurable by systems software so its "personality" can be optimized for a given serial data communications application.

The Z80-SIO is capable of handling synchronous and asynchronous byte oriented protocols such as IBM Bisync and synchronous bit oriented protocols such as HDLC and IBM SDLC.
This versatile device can also be used to support virtually any other serial protocol for applications other than data communications (e.g., cassette or floppy disk).

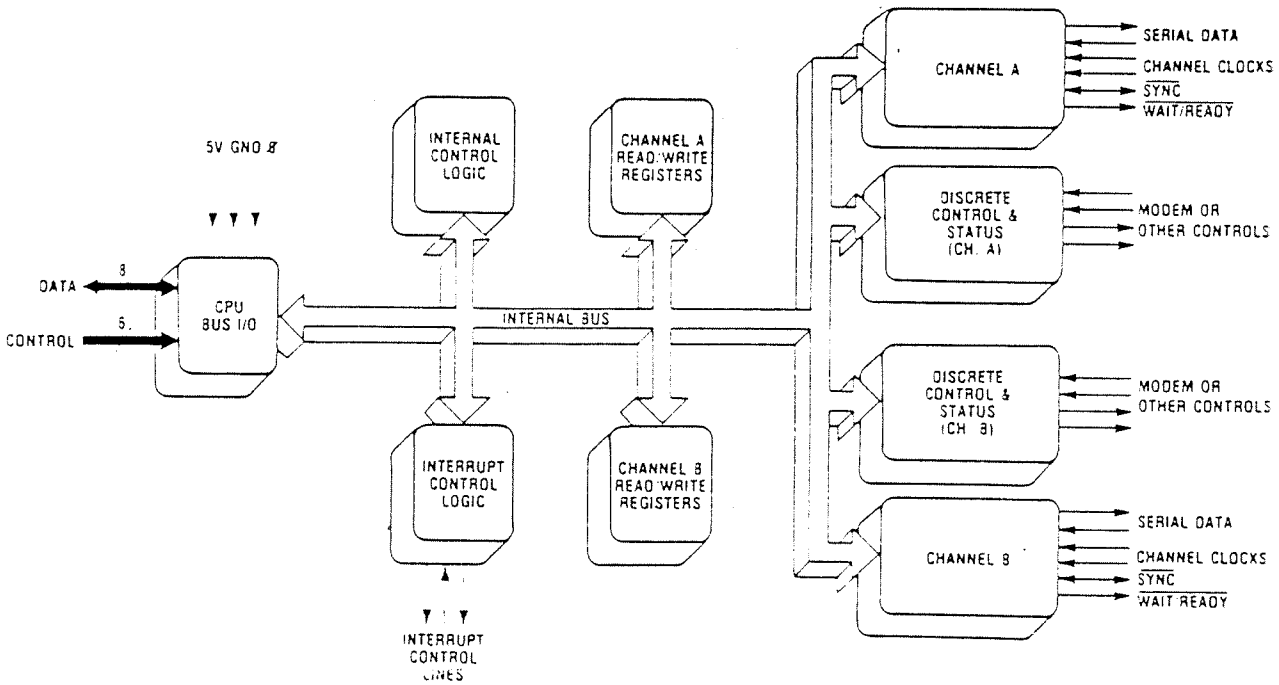
The Z80-SIO can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes.
The device also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general purpose I/O.

Structure:

* N-channel silicon gate depletion-load technology
* 40 pin DIP
* Single 5V power supply
* Single phase 5V clock
* All inputs and outputs TTL compatible

Features:

* Two independent full duplex channels
* Data rates in synchronous or asynchronous modes:
  - 0-550 Kbits/second with 2,5 MHz system clock
  - 0-880 Kbits/second with 4,0 MHz system clock rate.
* Receiver data registers quadruply buffered; transmitter doubly buffered
* Asynchronous features:
  - 5, 6, 7 or 8 bits/character
  - 1, 1 1/2 or 2 stop bits
  - Even, odd or no parity
  - X1, X16, X32 and X64 clock modes
  - Break generation and detection
  - Parity, overrun and framing error detection



Z80-SIO Block Diagram

Architecture

The device internal structure includes a Z80-CPU interface, internal control and interrupt logic, and two full-duplex channels. Associated with each channel are read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated in the text as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 illustrates the functions assigned to each read or write register.

WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and controls
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

(a) Write Register Functions

RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

(b) Read Register Functions

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$) are monitored by the discrete control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/ Status interrupts are prioritized in that order within each channel.

Data Path

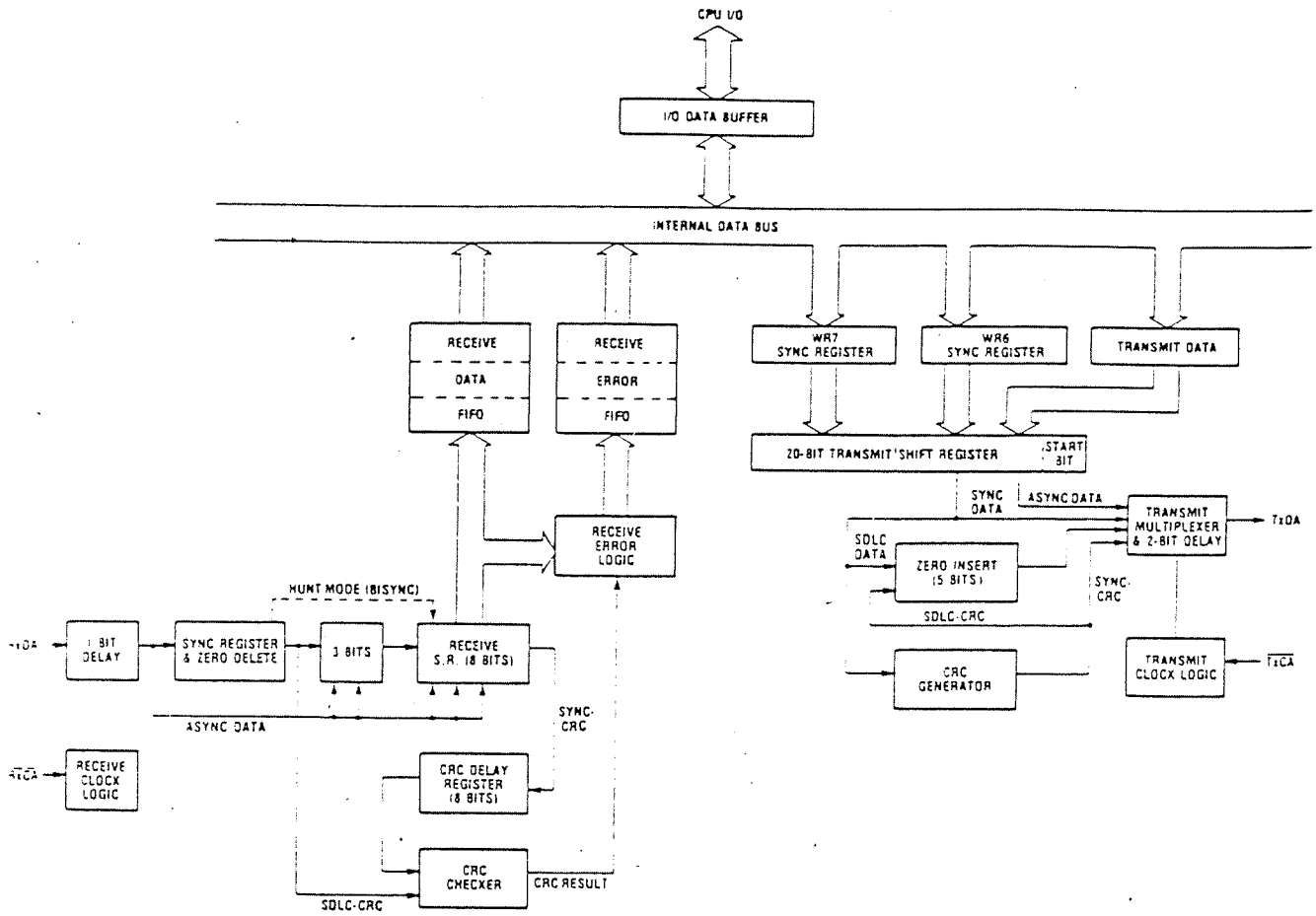
The transmit and receive data path for each channel is shown in Figure 4. The receiver has three 8-bit buffer registers in a FIFO arrangement (to provide a 3-byte delay) in addition to the 8-bit receive shift register. This arrangement creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. The receive error FIFO stores parity and framing errors and other types of status information for each of the three bytes in the receive data FIFO.

Incoming data is routed through one of several paths depending on the mode and character length. In the Asynchronous mode, serial data is entered in the 3-bit buffer if it has a character length of seven or eight bits, or is entered in the 3-bit receive shift register if it has a length of five or six bits.

In the Synchronous mode, however, the data path is determined by the phase of the receive process currently in operation. A Synchronous Receive operation begins with the receiver in the Hunt phase, during which the receiver searches the incoming data stream for a bit pattern that matches the preprogrammed sync characters (or flags in the SDLC mode). If the device is programmed for Monosync Hunt, a match is made with a single sync character stored in WR7. In Bisync Hunt, a match is made with dual sync characters stored in WR6 and WR7.

In either case the incoming data passes through the receive sync register, and is compared against the programmed sync character in WR6 or WR7. In the Monosync mode, a match between the sync character programmed into WR7 and the character assembled in the receive sync register establishes synchronization.

In the Bisync mode, however, incoming data is shifted to the receive shift register while the next eight bits of the message are assembled in the receive sync register. The match between the assembled character in the receive sync registers with the programmed sync character in WR6 and WR7 establishes synchronization. Once synchronization is established, incoming data by-



Transmit and Receive Data Path

passes the receive sync register and directly enters the 3-bit buffer.

In the SDLC mode, incoming data first passes through the receive sync register, which continuously monitors the receive data stream and performs zero deletion when indicated. Upon receiving five contiguous 1's, the sixth bit is inspected. If the sixth bit is a 0, it is deleted from the data stream. If the sixth bit is a 1, the seventh bit is inspected. If that bit is a 0, a Flag sequence has been received; if it is a 1, an Abort sequence has been received.

The reformatted data enters the 3-bit buffer and is transferred to the receive shift register. Note that the SDLC receive operation also begins in the Hunt phase, during which the Z80-SIO tries to match the assembled character in the receive shift register with the flag pattern in WR7. Once the first flag character is recognized, all subsequent data is routed through the same path, regardless of character length.

Although the same CRC checker is used for both SDLC and synchronous data, the data path taken for each mode is different. In Bisync protocol, a byte-oriented operation requires that the CPU decide to include the data character in CRC. To allow the CPU ample time to make this decision, the Z80-SIO provides an 8-bit delay for synchronous data. In the SDLC mode, no delay is provided since the Z80-SIO contains logic that determines the bytes on which CRC is calculated.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus and a 20-bit transmit shift register that can be loaded from WR6, WR7 and the transmit data register. WR6 and WR7 contain sync characters in the Monosync or Bisync modes, or address field (one character long) and flag respectively in the SDLC mode. During Synchronous modes, information contained in WR6 and WR7 is loaded into the transmit shift register at the beginning of the message and, as a time filler, in the middle of the message if a Transmit Underrun condition occurs. In the SDLC mode, the flags are loaded into the transmit shift register at the beginning and end of message.

Asynchronous data in the transmit shift register is formatted with start and stop bits and is shifted out to the transmit multiplexer at the selected clock rate. Synchronous (Monosync or Bisync) data is shifted out to the transmit multiplexer and also to the CRC generator at the x 1 clock rate.

SDLC/HDLC data is shifted out through the zero insertion logic, which is disabled while the flags are being sent. For all other fields (address, control and frame check) a 0 is inserted following five contiguous 1's in the data stream. The CRC generator result for SDLC data is also routed through the zero insertion logic.

Functional Description

The functional capabilities of the Z80-SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of various data communications protocols; as a Z80 family peripheral, it interacts with the Z80-CPU and other Z80 peripheral circuits, and shares their data, address and control busses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the Z80-SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capabilities.

The first part of the following functional description describes the interaction between the CPU and Z80-SIO; the second part introduces its data communications capabilities.

I/O CAPABILITIES

The Z80-SIO offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. The Polled mode avoids interrupts. Status registers RR0 and RR1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the Z80-SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a receive or transmit data transfer is needed. The status also indicates Error or other special status conditions (see "Z80-SIO Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 are accompanied by a Receive Character Available status in RR0.

Interrupts. The Z80-SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. As mentioned earlier, Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine, the Z80-SIO can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in WR2 is modified according to the assigned priority of the various interrupting conditions. The table in the Write Register 1 description (Z80-SIO Programming section) shows the modification details.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts (Figure 5). Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on first receive character
- Interrupt on all receive characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters has the option of modifying the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character or message basis (End Of Frame interrupt in SDLC, for example). The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort (SDLC mode) sequence in the data stream. The interrupt caused by the Break/Abort sequence has a special feature that allows the Z80-SIO to interrupt when the Break/Abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break/Abort condition in external logic.

CPU/DMA Block Transfer. The Z80-SIO provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers (Z80-DMA or other designs). The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{READY}}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{\text{WAIT/READY}}$ output can be defined under software control as a WAIT line in the CPU Block Transfer mode or as a READY line in the DMA Block Transfer mode.

To a DMA controller, the Z80-SIO READY output indicates that the Z80-SIO is ready to transfer data to or from memory. To the CPU, the WAIT output indicates that the Z80-SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle. The programming of bits 5, 6 and 7 of Write Register 1 and the logic states of the $\overline{\text{WAIT/READY}}$ line are defined in the Write Register 1 description (Z80-SIO Programming section.)

DATA COMMUNICATIONS CAPABILITIES

In addition to the I/O capabilities previously discussed, the Z80-SIO provides two independent full-duplex channels as well as Asynchronous, Synchronous and SDLC (HDLC) operational modes. These modes facilitate the implementation of commonly used data communications protocols.

The specific features of these modes are described in the following sections. To preserve the independence and completeness of each section, some information common to all modes is repeated.

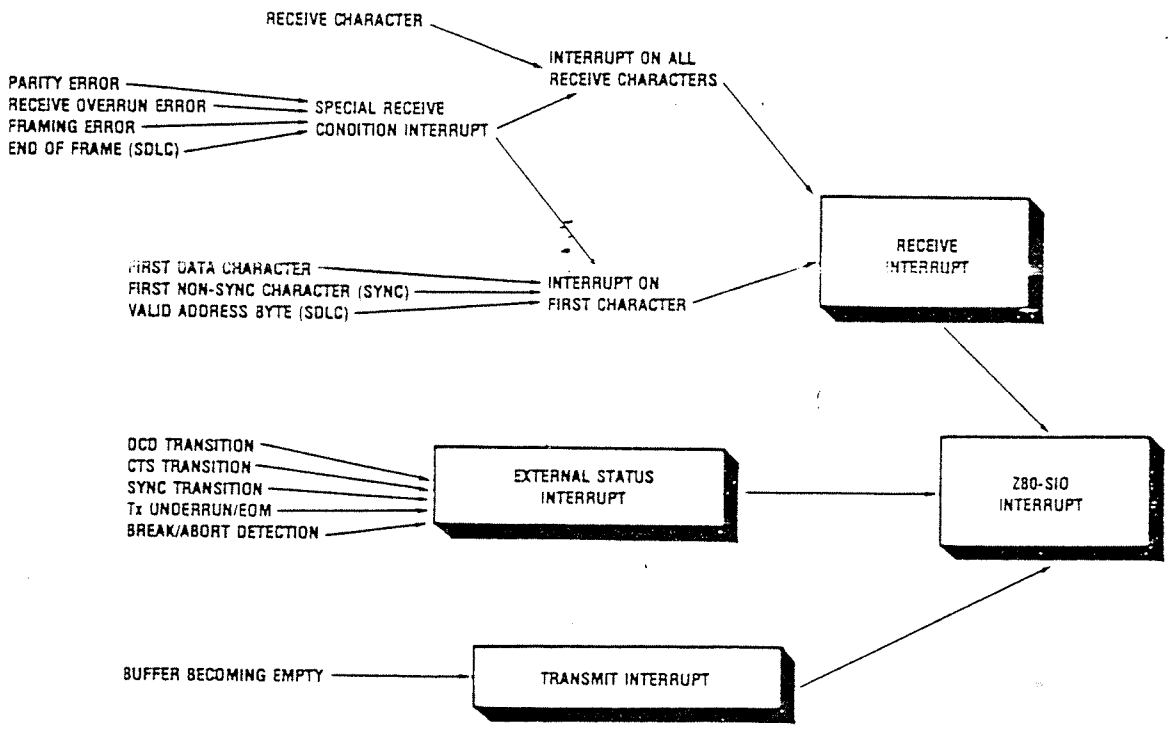


Figure 5. Interrupt Structure

Asynchronous Operation

To receive or transmit data in the Asynchronous mode, the Z80-SIO must be initialized with the following parameters: character length, clock rate, number of stop bits, even or odd parity, interrupt mode, and receiver or transmitter enable. The parameters are loaded into the appropriate write registers by the system program. WR4 parameters must be issued before WR1, WR3 and WR5 parameters or commands.

If the data is transmitted over a modem or RS232C interface, the REQUEST TO SEND (RTS) and DATA TERMINAL READY (DTR) outputs must be set along with the Transmit Enable bit. Transmission cannot begin until the Transmit Enable bit is set.

The Auto Enables feature allows the programmer to send the first data character of the message to the Z80-SIO without waiting for CTS. If the Auto Enables bit is set, the Z80-SIO will wait for the CTS pin to go Low before it begins data transmission. CTS, DCD and SYNC are general-purpose I/O lines that may be used for functions other than their labeled purposes. If CTS is used for another purpose, the Auto Enables Bit must be programmed to 0.

Figure 6 illustrates asynchronous message formats; Table 2 shows WR3, WR4 and WR5 with bits set to indicate the applicable modes, parameters and commands in asynchronous modes. WR2 (Channel B only) stores the interrupt vector; WR1 defines the interrupt modes and data transfer modes. WR6 and WR7 are not used in asynchronous modes. Table 3 shows the typical program steps that implement a full-duplex receive/transmit operation in either channel.

Asynchronous Transmit

The Transmit Data output (TxD) is held marking (High) when the transmitter has no data to send. Under program control, the Send Break (WR5, D_4) command can be issued to hold TxD spacing (Low) until the command is cleared.

The Z80-SIO automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits to the data character to be transmitted. When the character length is six or seven bits, the unused bits are automatically ignored by the Z80-SIO. If the character length is five bits or less, refer to the table in the Write Register 5 description (Z80-SIO Programming section) for the data format.

Serial data is shifted from TxD at a rate equal to 1, 1/16th, 1/32nd or 1/64th of the clock rate supplied to the Transmit Clock input (TxC). Serial data is shifted out on the falling edge of (TxC).

If set, the External/Status Interrupt mode monitors the status of DCD, CTS and SYNC throughout the transmission of the message. If these inputs change for a period of time greater than the minimum specified pulse width, the interrupt is generated. In a transmit operation, this feature is used to monitor the modem control signal CTS.

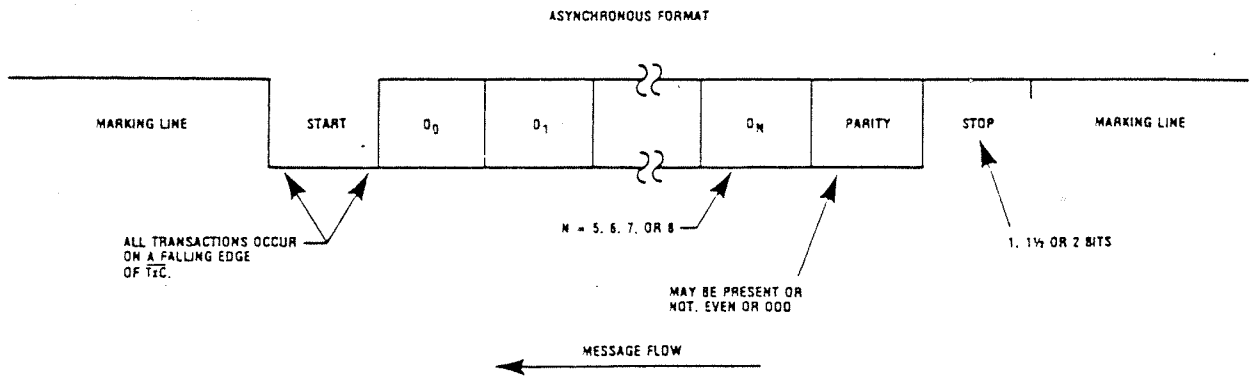


Figure 6. Asynchronous Message Format

Asynchronous Receive

An Asynchronous Receive operation begins when the Receive Enable bit is set. If the Auto Enables option is selected, $\overline{DCD}$ must be Low as well. A Low (spacing) condition on the Receive Data input (RxD) indicates a start bit. If this Low persists for at least one-half of a bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on an otherwise marking line.

If the ×1 clock mode is selected, bit synchronization must be accomplished externally. Receive data is sampled on the rising edge of RxC. The receiver inserts 1's when a character length of other than eight bits is used. If parity is enabled, the parity bit is not stripped from the assembled character for character lengths other than eight bits. For lengths other than eight bits, the receiver assembles a character length of the required number of data bits, plus a parity bit and 1's for any unused bits. For example, the receiver assembles a 5-bit character with the following format: 1 1 P D_4 D_3 D_2 D_1 D_0.

Since the receiver is buffered by three 8-bit registers in addition to the receive shift register, the CPU has enough time to service an interrupt and to accept the data character assembled by the Z80-SIO. The receiver also has three buffers that store error flags for each data character in the receive buffer. These error flags are loaded at the same time as the data characters.

After a character is received, it is checked for the following error conditions:

- When parity is enabled, the Parity Error bit (RR1, D_4) is set whenever the parity bit of the character does not match with the programmed parity. Once this bit is set, it remains set until the Error Reset Command (WR0) is given.
- The Framing Error bit (RR1, D_6) is set if the character is assembled without any stop bits (that is, a Low level detected for a stop bit). Unlike the Parity Error bit, this bit is set (and not latched) only for the character on which it occurred. Detection of framing error adds an additional one-half of a bit time to the character time so the framing error is not interpreted as a new start bit.
- If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1, D_5) is set. When this occurs, the fourth character assembled replaces the third character in the receive buffers. With this arrangement, only the character that has been written over is flagged with the Receive Overrun Error bit. Like Parity Error, this bit can only be reset by the Error Reset command from the CPU. Both the Framing Error and Receive Overrun Error cause an interrupt with the interrupt vector indicating a Special Receive condition (if Status Affects Vector is selected).

Since the Parity Error and Receive Overrun Error flags are latched, the error status that is read reflects an error in the current word in the receive buffer plus any Parity or Overrun Errors received since the last Error Reset command. To keep correspondence between the state of the error buffers and the contents of the receive data buffers, the error status register must be read before the data. This is easily accomplished if vectored

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	0	0	0	0	Rx ENABLE
WR4	00 = ×1 CLOCK MODE 01 = ×16 CLOCK MODE 10 = ×32 CLOCK MODE 11 = ×64 CLOCK MODE		0	0	00 = NOT USED 01 = 1 STOP BIT/CHAR 10 = 1½ STOP BITS/CHAR 11 = 2 STOP BITS/CHAR		EVEN/$\overline{ODD}$ PARITY	PARITY ENABLE
WR5	DTR	00 = Tx 5 BITS (OR LESS)/CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR	SEND BREAK	Tx ENABLE	0		RTS	0

Table 2. Contents of Write Registers 3, 4 and 5 in Asynchronous Modes

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	*INFORMATION LOADED:*	
	WR0	CHANNEL RESET	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4, RESET EXTERNAL/STATUS INTERRUPT	
	WR4	ASYNCHRONOUS MODE, PARITY INFORMATION, STOP BITS INFORMATION, CLOCK RATE INFORMATION	Issue parameters
INITIALIZE	WR0	POINTER 3	
	WR3	RECEIVE ENABLE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	
	WR0	POINTER 5	
	WR5	REQUEST TO SEND, TRANSMIT ENABLE, TRANSMIT CHARACTER LENGTH, DATA TERMINAL READY	Receive and Transmit both fully initialized. Auto Enables will enable Transmitter if $\overline{CTS}$ is active and Receiver if $\overline{DCD}$ is active.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	TRANSMIT INTERRUPT ENABLE, STATUS AFFECTS VECTOR, INTERRUPT ON ALL RECEIVE CHARACTERS, DISABLE WAIT/READY FUNCTION, EXTERNAL INTERRUPT ENABLE	Transmit/Receive interrupt mode selected. External Interrupt monitors the status of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ inputs and detects the Break sequence. Status Affects Vector in Channel B only.
	TRANSFER FIRST DATA BYTE TO SIO		This data byte must be transferred or no transmit interrupts will occur.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Program is waiting for an interrupt from the SIO.
DATA TRANSFER AND ERROR MONITORING	Z80 INTERRUPT ACKNOWLEDGE CYCLE TRANSFERS RR2 TO CPU		When the interrupt occurs, the interrupt vector is modified by: 1. Receive Character Available; 2. Transmit Buffer Empty; 3. External/Status change; and 4. Special Receive condition.
	IF A CHARACTER IS RECEIVED: • TRANSFER DATA CHARACTER TO CPU • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		Program control is transferred to one of the eight interrupt service routines.
	IF TRANSMITTER BUFFER IS EMPTY: • TRANSFER DATA CHARACTER TO SIO • UPDATE POINTERS AND PARAMETERS • RETURN FROM INTERRUPT		
	IF EXTERNAL STATUS CHANGES: • TRANSFER RR0 TO CPU • PERFORM ERROR ROUTINES (INCLUDE BREAK DETECTION) • RETURN FROM INTERRUPT		If used with processors other than the Z80, the modified interrupt vector (RR2) should be returned to the CPU in the Interrupt Acknowledge sequence.
	IF SPECIAL RECEIVE CONDITION OCCURS: • TRANSFER RR1 TO CPU • DO SPECIAL ERROR (E.G. FRAMING ERROR) ROUTINE • RETURN FROM INTERRUPT		
TERMINATION	REDEFINE RECEIVE/TRANSMIT INTERRUPT MODES		When transmit or receive data transfer is complete.
	DISABLE TRANSMIT/RECEIVE MODES		
	UPDATE MODEM CONTROL OUTPUTS (E.G. RTS OFF)		In Transmit, the All Sent status bit indicates transmission is complete.

Table 3. Asynchronous Mode

interrupts are used, because a special interrupt vector is generated for these conditions.

While the External/Status interrupt is enabled, break detection causes an interrupt and the Break Detected status bit (RR0, D_7) is set. The Break Detected interrupt should be handled by issuing the Reset External/Status Interrupt command to the Z80-SIO in response to the first Break Detected interrupt that has a Break status of 1 (RR0, D_7). The Z80-SIO monitors the Receive Data input and waits for the Break sequence to terminate, at which point the Z80-SIO interrupts the CPU with the Break status set to 0. The CPU must again issue the Reset External/Status Interrupt command in its interrupt service routine to reinitialize the break detection logic.

The External/Status interrupt also monitors the status of $\overline{DCD}$. If the $\overline{DCD}$ pin becomes inactive for a period greater than the minimum specified pulse width, an interrupt is generated with the DCD status bit (RR0, D_3) set to 1. Note that the $\overline{DCD}$ input is inverted in the RR0 status register.

If the status is read after the data, the error data for the next word is also included if it has been stacked in the buffer. If operations are performed rapidly enough so the next character is not yet received, the status register remains valid. An exception occurs when the Interrupt On First Character Only mode is selected. A special interrupt in this mode holds the error data and the character itself (even if read from the buffer) until the Error Reset command is issued. This prevents further data from becoming available in the receiver until the Reset command is issued, and allows CPU intervention on the character with the error even if DMA or block transfer techniques are being used.

If Interrupt On Every Character is selected, the interrupt vector is different if there is an error status in RR1. If a Receiver Overrun occurs, the most recent character received is loaded into the buffer; the character preceding it is lost. When the character that has been written over the other characters is read, the Receive Overrun bit is set and the Special Receive Condition vector is returned if Status Affects Vector is enabled.

In a polled environment, the Receive Character Available bit (RR0, D_0) must be monitored so the Z80-CPU can know when to read a character. This bit is automatically reset when the receive buffers are read. To prevent overwriting data in polled operations, the transmit buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit is set to 1 whenever the transmit buffer is empty.

Synchronous Operation

Before describing synchronous transmission and reception, the three types of character synchronization—Monosync, Bisync and External Sync—require some explanation. These modes use the ×1 clock for both Transmit and Receive operations. Data is sampled on the rising edge of the Receive Clock input ($\overline{RxC}$). Transmitter data transitions occur on the falling edge of the Transmit Clock input ($\overline{TxC}$).

The differences between Monosync, Bisync and External Sync are in the manner in which initial character synchronization is achieved. The mode of operation must be selected before sync characters are loaded, because the registers are used differently in the various modes. Figure 7 shows the formats for all three of these synchronous modes.

Monosync. In a Receive operation, matching a single sync character (8-bit sync mode) with the programmed sync character stored in WR7 implies character synchronization and enables data transfer.

Bisync. Matching two contiguous sync characters (16-bit sync mode) with the programmed sync characters stored in WR6 and WR7 implies character synchronization. In both the Monosync and Bisync modes, $\overline{SYNC}$ is used as an output, and is active for the part of the receive clock that detects the sync character.

External Sync. In this mode, character synchronization is established externally; $\overline{SYNC}$ is an input that indicates external character synchronization has been achieved. After the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. The SYNC input must be held Low until character synchronization is lost. Character assembly begins on the rising edge of $\overline{RxC}$ that precedes the falling edge of $\overline{SYNC}$.

In all cases after a reset, the receiver is in the Hunt phase, during which the Z80-SIO looks for character synchronization. The hunt can begin only when the receiver is enabled, and data transfer can begin only when character synchronization has been achieved. If character synchronization is lost, the Hunt phase can be re-entered by writing a control word with the Enter Hunt Phase bit set (WR3, D_4). In the Transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16). In the Monosync mode, the transmitter transmits from WR6; the receiver compares against WR7.

In the Monosync, Bisync and External Sync modes, assembly of received data continues until the Z80-SIO is reset, or until the receiver is disabled (by command or by $\overline{DCD}$ in the Auto Enables mode), or until the CPU sets the Enter Hunt Phase bit.

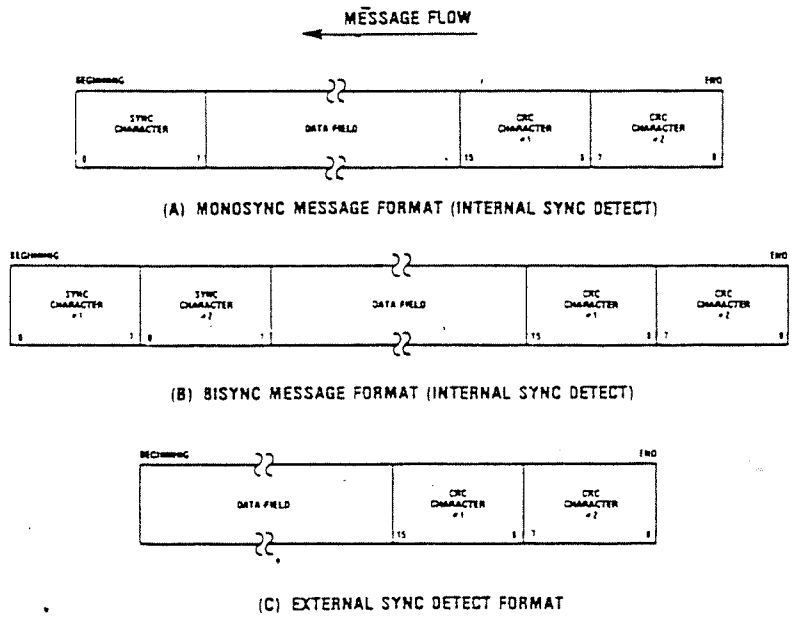


Figure 7. Synchronous Formats

After initial synchronization has been achieved, the operation of the Monosync, Bisync and External Sync modes is quite similar. Any differences are specified in the following text.

Table 4 shows how WR3, WR4 and WR5 are used in synchronous receive and transmit operations. WR0 points to other registers and issues various commands, WR1 defines the interrupt modes, WR2 stores the interrupt vector, and WR6 and WR7 store sync characters. Table 5 illustrates the typical program steps that implement a half-duplex Bisync transmit operation.

Synchronous Transmit

INITIALIZATION

The system program must initialize the transmitter with the following parameters: odd or even parity, × 1 clock mode, 8- or 16-bit sync character(s), CRC polynomial, Transmitter Enables, Request To Send, Data Terminal Ready, interrupt modes and transmit character length. WR4 parameters must be issued before WR1, WR3, WR5, WR6 and WR7 parameters or commands.

One of two polynomials—CRC-16 ($X^{16}+X^{15}+X^2+1$) or SDLC ($X^{16}+X^{12}+X^5+1$)—may be used with synchronous modes. In either case (SDLC mode not selected), the CRC generator and checker are reset to all 0's. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command bits (WR0). Both the transmitter and the receiver use the same polynomial.

Transmit Interrupt Enable or Wait/Ready Enable can be selected to transfer the data. The External/Status interrupt mode is used to monitor the status of the $\overline{\text{CLEAR TO SEND}}$ input as well as the Transmit Underrun/EOM latch. Optionally, the Auto Enables feature can be used to enable the transmitter when $\overline{\text{CTS}}$ is active. The first data transfer to the Z80-SIO can begin when the External/Status interrupt occurs (CTS status bit set) or immediately following the Transmit Enable command (if the Auto Enables modes is set).

Transmit data is held marking after reset or if the transmitter is not enabled. Break may be programmed to generate a spacing line that begins as soon as the Send Break bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

DATA TRANSFER AND STATUS MONITORING

In this phase, there are several combinations of interrupts and Wait/Ready.

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit (WR1, D_1) is set, an interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied either by writing another character into the transmitter or by resetting the Transmitter Interrupt Pending latch with a Reset Transmitter Pending command (WR0, CMD_5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupts, because it is the process of the buffer becoming empty that causes the interrupts and the buffer cannot become empty when it is already empty. This situation does cause a Transmit Underrun condition, which is explained in the "Bisync Transmit Underrun" section.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	0	SYNC CHAR LOAD INHIBIT	RX ENABLE
WR4	0	0	00 = 8-BIT SYNC CHAR 01 = 16-BIT SYNC CHAR 10 = SDLC MODE 11 = EXT SYNC MODE		0	0 SELECTS SYNC MODES	EVEN/$\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00 = Tx 5 BITS (OR LESS)/CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		SEND BREAK	Tx ENABLE	1 SELECTS CRC-16	RTS	Tx CRC ENABLE

Table 4. Contents of Write Registers 3, 4 and 5 in Synchronous Modes

Data Transfer Using WAIT/READY. To the CPU, the activation of $\overline{\text{WAIT}}$ indicates that the Z80-SIO is not ready to accept data and that the CPU must extend the output cycle. To a DMA controller, READY indicates that the transmit buffer is empty and that the Z80-SIO is ready to accept the next data character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition.

Bisync Transmit Underrun. In Bisync protocol, filler characters are inserted to maintain synchronization when the transmitter has no data to send (Transmit Underrun condition). The Z80-SIO has two programmable options for solving this situation: it can insert sync characters, or it can send the CRC characters generated so far, followed by sync characters.

These options are under the control of the Reset Transmit Underrun/EOM command in WR0. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RR0, D_6) is in a set condition and allows the insertion of sync characters when there is no data to send. CRC is not calculated on the automatically inserted sync characters. When the CPU detects the end of message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data. In this case, the Z80-SIO sends CRC, followed by sync characters, to terminate the message.

There is no restriction as to when in the message the Transmit Underrun/EOM bit can be reset. If Reset is issued after the first data character has been loaded the 16-bit CRC is sent and followed by sync characters the first time the transmitter has no data to send. Because of the Transmit Underrun condition, an External/Status interrupt is generated whenever the Transmit Underrun/EOM bit becomes set.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded. The status indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded). If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter (WR5, D_3).

Pad characters may be sent by setting the Z80-SIO to 8 bits/transmit character and writing FF to the transmitter while CRC is being sent. Alternatively, the sync characters can be redefined as pad characters during this time. The following example is included to clarify this point.

The Z80-SIO interrupts with the Transmit Buffer Empty bit set.

The CPU recognizes that the last character (ETX) of the message has already been sent to the Z80-SIO by examining the internal program status.

To force the Z80-SIO to send CRC, the CPU issues the Reset Transmit Underrun/EOM Latch command (WR0) and satisfies the interrupt with the Reset Transmit Interrupt Pending command. (This command prevents the Z80-SIO from requesting more data.) Because of the transmit underrun caused by this command, the Z80-SIO starts sending CRC. The Z80-SIO also causes an External/Status interrupt with the Transmit Underrun/EOM latch set.

The CPU satisfies this interrupt by loading pad characters into the transmit buffer and issuing the Reset External/Status Interrupt command.

With this sequence, CRC is followed by a pad character instead of a sync character. Note that the Z80-SIO will interrupt with a Transmit Buffer Empty interrupt when CRC is completely sent and that the pad character is loaded into the transmit shift register.

From this point on the CPU can send more pad characters or sync characters.

Bisync CRC Generation. Setting the Transmit CRC enable bit (WR5, D_0) initiates CRC accumulation when the program sends the first data character to the Z80-SIO. Although the Z80-SIO automatically transmits up to two sync characters (16-bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded from the transmit data buffer into the transmit shift register. To ensure this bit is in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the Z80-SIO.

Transmit Transparent Mode. Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16-bit sync characters. Exclusion of DLE characters from CRC calculation can be achieved by disabling CRC calculation immediately preceding the DLE character transfer to the Z80-SIO.

In the case of a Transmit Underrun condition in the Transparent mode, a pair of DLE-SYN characters are sent. The Z80-SIO can be programmed to send the DLE-SYN sequence by loading a DLE character into WR6 and a sync character into WR7.

Transmit Termination. The Z80-SIO is equipped with a special termination feature that maintains data integrity and validity. If the transmitter is disabled while a data or sync character is being sent, that character is sent as usual, but is followed by a marking line rather than CRC or sync characters. When the transmitter is disabled, a

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
	REGISTER	*INFORMATION LOADED*	
INITIALIZE	WR0	CHANNEL RESET, RESET TRANSMIT CRC GENERATOR	Reset SIO, initilize CRC generator.
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 3	
	WR3	AUTO ENABLES	Transmission begins only after $\overline{CTS}$ is detected.
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODES INFORMATION, x1 CLOCK MODE	Issue transmit parameters.
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	POINTER 7, RESET EXTERNAL/STATUS INTERRUPTS	
	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, TRANSMIT INTERRUPT ENABLE OR WAIT/READY MODE ENABLE	External Interrupt mode monitors the status of $\overline{CTS}$ and $\overline{DCD}$ input pins as well as the status of Tx Underrun/EOM latch. Transmit Interrupt Enable interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data using DMA or CPU Block Transfer.
	WR0	POINTER 5	Status Affects Vector (Channel B only).
	WR5	REQUEST TO SEND, TRANSMIT ENABLE, BISYNC CRC, TRANSMIT CHARACTER LENGTH	Transmit CRC Enable should be set when first non-sync data is sent to Z80-SIO.
	FIRST SYNC BYTE TO SIO		Need several sync characters in the beginning of message. Transmitter is fully initialized.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Waiting for interrupt or Wait/Ready output to transfer data.
DATA TRANSFER AND STATUS MONITORING	*WHEN INTERRUPT (WAIT/READY) OCCURS:* • INCLUDE/EXECLUDE DATA BYTE FROM CRC ACCUMULATION (IN SIO). • TRANSFER DATA BYTE FROM CPU (OR MEMORY) TO SIO. • DETECT AND SET APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU). • RESET Tx UNDERRUN/EOM LATCH (WR0) IF LAST CHARACTER OF MESSAGE IS DETECTED. • UPDATE POINTERS AND PARAMETERS (CPU). • RETURN FROM INTERRUPT		Interrupt occurs (Wait/Ready becomes active) when first data byte is being sent. Wait mode allows CPU block transfer from memory to SIO; Ready mode allows DMA block transfer from memory to SIO. The DMA chip can be programmed to capture special control characters (by examining only the bits that specify ASCII or EBCDIC control characters), and interrupt CPU.
	IF ERROR CONDITION OR STATUS CHANGE OCCURS. • TRANSFER RR0 TO CPU. • EXECUTE ERROR ROUTINE. • RETURN FROM INTERRUPT.		Tx Underrun/EOM indicates either transmit underrun (sync character being sent) or end of message (CRC-16 being sent).
TERMINATION	REDEFINE INTERRUPT MODES.		
	UPDATE MODEM CONTROL OUTPUTS (E.G., TURN OFF RTS).		Program should gracefully terminate message.
	DISABLE TRANSMIT MODE		

Table 5. Bisync Transmit Mode

character in the buffer remains in the buffer. If the transmitter is disabled while CRC is being sent, the 16-bit transmission is completed, but sync is sent instead of CRC.

A programmed break is effective as soon as it is written into the control register; characters in the transmit buffer and shift register are lost.

In all modes, characters are sent with the least significant bits first. This requires right-hand justification of transmitted data if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 discussion (Z80-SIO Programming section) must be used for the data format. The states of any unused bits in a data character are irrelevant, except when in the Five Bits Or Less mode.

If the External/Status Interrupt Enable bit is set, transmitter conditions such as "starting to send CRC characters," "starting to send sync characters," and $\overline{CTS}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. This interrupt mode may be used during block transfers.

All interrupts may be disabled for operation in a Polled mode, or to avoid interrupts at inappropriate times during the execution of a program.

Synchronous Receive

INITIALIZATION

The system program initiates the Synchronous Receive operation with the following parameters: odd or even parity, 8- or 16-bit sync characters, ×1 clock mode, CRC polynomial, receive character length, etc. Sync characters must be loaded into registers WR6 and WR7. The receivers can be enabled only after all receive parameters are set. WR4 parameters must be issued before WR1, WR3, WR5, WR6 and WR7 parameters or commands.

After this is done, the receiver is in the Hunt phase. It remains in this phase until character synchronization is achieved. Note that, under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit bit in WR3.

DATA TRANSFER AND STATUS MONITORING

After character synchronization is achieved, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer the data and its associated status to the CPU.

No Interrupts Enabled. This mode is used for a purely polled operation or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a polling loop or a Block Transfer instruction using $\overline{WAIT/READY}$ to synchronize the CPU or the DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter interrupts only if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt On Next Receive Character command to allow the next character received to generate an interrupt. Parity errors do not cause interrupts in this mode, but End Of Frame (SDLC mode) and Receive Overrun do.

If External/Status interrupts are enabled, they may interrupt any time $\overline{DCD}$ changes state.

Interrupt On Every Character. Whenever a character enters the receive buffer, an interrupt is generated. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected. Optionally, a Parity Error may be directed not to generate the special interrupt vector.

Special Receive Condition Interrupts. The Special Receive Condition interrupt can occur only if either the Receive Interrupt On First Character Only or Interrupt On Every Receive Character modes is also set. The Special Receive Condition interrupt is caused by the Receive Overrun error condition. Since the Receive Overrun and Parity error status bits are latched, the error status—when read—reflects an error in the current word in the receive buffer in addition to any Parity or Overrun errors received since the last Error Reset command. These status bits can only be reset by the Error reset command.

CRC Error Checking and Termination. A CRC error check on the receive message can be performed on a per character basis under program control. The Receive CRC Enable bit (WR3, D_3) must be set/reset by the program before the next character is transferred from the receive shift register into the receive buffer register. This ensures proper inclusion or exclusion of data characters in the CRC check.

To allow the CPU ample time to enable or disable the CRC check on a particular character, the Z80-SIO calculates CRC eight bit times after the character has been transferred to the receive buffer. If CRC is enabled before the next character is transferred, CRC is calculated on the transferred character. If CRC is disabled before the time of the next transfer, calculation proceeds on the word in progress, but the word just transferred to the buffer is not included. When these requirements are satisfied, the 3-byte receive data buffer is, in effect, unusable in Bisync operation. CRC may be enabled and disabled as many times as necessary for a given calculation.

In the Monosync, Bisync and External Sync modes, the CRC/Framing Error bit (RR1, D_6) contains the comparison result of the CRC checker 16 bit times (eight bits delay and eight shifts for CRC) after the character has been transferred from the receive shift register to the buffer. The result should be zero, indicating an error-

free transmission. (Note that the result is valid only at the end of CRC calculation. If the result is examined before this time, it usually indicates an error.) The comparison is made with each transfer and is valid only as long as the character remains in the receive FIFO.

Following is an example of the CRC checking operation when four characters (A, B, C and D) are received in that order.

Character A loaded into buffer
Character B loaded into buffer

If CRC is disabled before C is in the buffer, CRC is not calculated on B.

Character C loaded into buffer

After C is loaded, the CRC/Framing Error bit shows the result of the comparison through character A.

Character D loaded into buffer

After D is in the buffer, the CRC Error bit shows the result of the comparison through character B whether or not B was included in the CRC calculations.

Due to the serial nature of CRC calculation, the Receive Clock ($\overline{RxC}$) must cycle 16 times (8-bit delay plus 8-bit CRC shift) after the second CRC character has been loaded into the receive buffer, or 20 times (the previous 16 plus 3-bit buffer delay and 1-bit input delay) after the last bit is at the RxD input, before CRC calculation is complete. A faster external clock can be gated into the Receive Clock input to supply the required 16 cycles. The Transmit and Receive Data Path diagram (Figure 4) illustrates the various points of delay in the CRC path.

The typical program steps that implement a half-duplex Bisync Receive mode are illustrated in Table 6. The complete set of command and status bit definitions are explained under "Z80-SIO Programming."

FUNCTION		TYPICAL PROGRAM STEPS	COMMENTS
	REGISTER	*INFORMATION LOADED*	
	WR0	CHANNEL RESET, RESET RECEIVE CRC CHECKER	Reset SIO; initialize Receive CRC checker
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODES INFORMATION, x1 CLOCK MODE	Issue receive parameters.
	WR0	POINTER 5, RESET EXTERNAL STATUS INTERRUPT	
	WR5	BISYNC CRC-16, DATA TERMINAL READY	
	WR0	POINTER 3	
INITIALIZE	WR3	SYNC CHARACTER LOAD INHIBIT, RECEIVE CRC ENABLE; ENTER HUNT MODE, AUTO ENABLES, RECEIVE CHARACTER LENGTH	Sync character load inhibit strips all the leading sync characters at the beginning of the message. Auto Enables enables the receiver to accept data only after the $\overline{DCD}$ input is active.
	WR0	POINTER 6	
	WR6	SYNC CHARACTER 1	
	WR0	POINTER 7	
	WR7	SYNC CHARACTER 2	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPT	
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY	In this interrupt mode, only the first non-sync data character is transferred to the CPU. All subsequent data is transferred on a DMA basis; however Special Receive Condition interrupts will interrupt the CPU. Status Affects Vector used in Channel B only.

Table 6. Bisync Receive Mode

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
INITIALIZE (CONTINUED)	WR0	POINTER 3, ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Resetting this interrupt mode provides simple program loopback entry for the next transaction.
	WR3	RECEIVE ENABLE, SYNC CHARACTER LOAD INHIBIT, ENTER HUNT MODE, AUTO ENABLE, RECEIVE WORD LENGTH	WR3 is reissued to enable receiver. Receive CRC Enable must be set after receiving SOH or STX character.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Receive mode is fully initialized and the system is waiting for interrupt on first character.
DATA TRANSFER AND STATUS MONITORING	*WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING:* • TRANSFERS DATA BYTE TO CPU • DETECTS AND SETS APPROPRIATE FLAGS FOR CONTROL CHARACTERS (IN CPU) • INCLUDES/EXCLUDES DATA BYTE IN CRC CHECKER • UPDATES POINTERS AND OTHER PARAMETERS • ENABLES WAIT/READY FOR DMA OPERATION • ENABLES DMA CONTROLLER • RETURNS FROM INTERRUPT		During the Hunt mode, the SIO detects two contiguous characters to establish synchronization. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller. The controller is also programmed to capture special characters (by examining only the bits that specify ASCII or EBCDIC control characters) and interrupt the CPU upon detection. In response, the CPU examines the status or control characters and takes appropriate action (e.g. CRC Enable Update).
	WHEN WAIT/READY BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING: • TRANSFERS DATA BYTE TO MEMORY • INTERRUPTS CPU IF A SPECIAL CHARACTER IS CAPTURED BY THE DMA CONTROLLER • INTERRUPTS THE CPU IF THE LAST CHARACTER OF THE MESSAGE IS DETECTED		
	FOR MESSAGE TERMINATION, THE CPU DOES THE FOLLOWING: • TRANSFERS RR1 TO THE CPU • SETS ACK/NAK REPLY FLAG BASED ON CRC RESULT • UPDATES POINTERS AND PARAMETERS • RETURNS FROM INTERRUPT		The SIO interrupts the CPU for error condition, and the error routine aborts the present message, clears the error condition, and repeats the operation.
TERMINATION	REDEFINE INTERRUPT MODES AND SYNC MODES UPDATE MODEM CONTROLS DISABLES RECEIVE MODE		

Table 6. Bisync Receive Mode (Continued)

SDLC (HDLC) Operation

The Z80-SIO is capable of handling both High-level Synchronous Data Link Control (HDLC) and IBM Synchronous Data Link Control (SDLC) protocols. In the following text, only SDLC is referred to because of the high degree of similarity between SDLC and HDLC.

The SDLC mode is considerably different than Synchronous Bisync protocol because it is bit oriented rather than character oriented and, therefore, can naturally handle transparent operation. Bit orientation makes SDLC a flexible protocol in terms of message length and bit patterns. The Z80-SIO has several built-in features to handle variable message length. Detailed information concerning SDLC protocol can be found in literature published on this subject, such as IBM document GA27-3093.

The SDLC message, called the frame (Figure 8), is opened and closed by flags that are similar to the sync characters in Bisync protocol. The Z80-SIO handles the transmission and recognition of the flag characters that mark the beginning and end of the frame. Note that the Z80-SIO can receive shared-zero flags, but cannot transmit them. The 8-bit address field of an SDLC frame contains the secondary station address. The Z80-SIO has an Address Search mode that recognizes the secondary station address so it can accept or reject the frame.

Since the control field of the SDLC frame is transparent to the Z80-SIO, it is simply transferred to the CPU. The Z80-SIO handles the Frame Check sequence in a manner that simplifies the program by incorporating features such as initializing the CRC generator to all 1's, resetting the CRC checker when the opening flag is detected in the Receive mode, and sending the Frame Check/Flag sequence in the Transmit mode. Controller hardware is simplified by automatic zero insertion and deletion logic contained in the Z80-SIO.

Table 7 shows the contents of WR3, WR4 and WR5 during SDLC Receive and Transmit modes. WR0 points to other registers and issues various commands. WR1 defines the interrupt modes. WR2 stores the interrupt vector. WR7 stores the flag character and WR6 the secondary address.

SDLC Transmit

INITIALIZATION

Like Synchronous operation, the SDLC Transmit mode must be initialized with the following parameters: SDLC mode, SDLC polynomial, Request To Send, Data Terminal Ready, transmit character length, transmit interrupt modes (or Wait/Ready function), Transmit Enable, Auto Enables and External/Status interrupt.

Selecting the SDLC mode and the SDLC polynomial enables the Z80-SIO to initialize the CRC Generator to all 1's. This is accomplished by issuing the Reset Transmit CRC Generator command (WR0). Refer to the Synchronous Operation section for more details on the interrupt modes.

After reset, or when the transmitter is not enabled, the Transmit Data output is held marking. Break may be programmed to generate a spacing line. With the transmitter fully initialized and enabled, continuous flags are transmitted on the Transmit Data output.

An abort sequence may be sent by issuing the Send Abort command (WR0, CMD_1). This causes at least eight, but less than fourteen, 1's to be sent before the line reverts to continuous flags. It is possible that the Abort sequence (eight 1's) could follow up to five continuous 1 bits (allowed by the zero insertion logic) and thus cause up to thirteen 1's to be sent. Any data being transmitted and any data in the transmit buffer is lost when an abort is issued.

When required, an extra 0 is automatically inserted when there are five contiguous 1's in the data stream. This does not apply to flags or aborts.

DATA TRANSFER AND STATUS MONITORING

There are several combinations of interrupts and the Wait/Ready function in the SLDC mode.

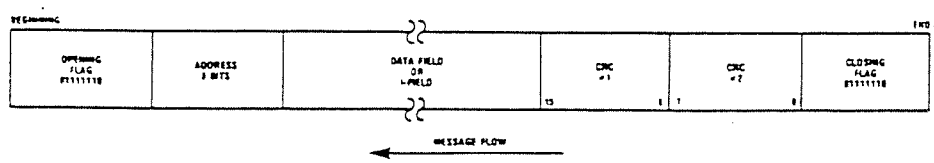


Figure 8. Transmit/Receive SDLC/HDLC Message Format

Data Transfer Using Interrupts. If the Transmit Interrupt Enable bit is set, an interrupt is generated each time the buffer becomes empty. The interrupt may be satisfied either by writing another character into the transmitter or by resetting the Transmit Interrupt Pending latch with a Reset Transmitter Pending command (WR0, CMD_5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there are no further transmitter interrupts. The result is a Transmit Underrun condition. When another character is written and sent out, the transmitter can again become empty and interrupt the CPU. Following the flags in an SDLC operation, the 8-bit address field, control field and information field may be sent to the Z80-SIO using the Transmit Interrupt mode. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

When the transmitter is first enabled, it is already empty and obviously cannot then become empty. Therefore, no Transmit Buffer Empty interrupts can occur until after the first data character is written.

When the transmitter is first enabled, it is already empty and cannot then become empty. Therefore, no Transmit Buffer Empty interrupts can occur until after the first data character is written.

Data Transfer Using Wait/Ready. If the Wait/Ready function has been selected, WAIT indicates to the CPU that the Z80-SIO is not ready to accept the data and the CPU must extend the I/O cycle. To a DMA controller, READY indicates that the transmitter buffer is empty and that the Z80-SIO is ready to accept the next character. If the data character is not loaded into the Z80-SIO by the time the transmit shift register is empty, the Z80-SIO enters the Transmit Underrun condition. Address, control and information fields may be transferred to the Z80-SIO with this mode using the Wait/Ready function. The Z80-SIO transmits the Frame Check sequence using the Transmit Underrun feature.

SDLC Transmit Underrun/End Of Message. SDLC-like protocols do not have provisions for fill characters within a message. The Z80-SIO therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by first sending the two bytes of CRC and following these with one or more flags. This technique allows very high-speed transmissions under DMA or CPU block I/O control without requiring the CPU to respond quickly to the end of message situation.

The action that the Z80-SIO takes in the underrun situation depends on the state of the Transmit Underrun/EOM command. Following a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Consequently, flag characters are sent. The Z80-SIO begins to send the frame as data is written into the transmit buffer. Between the time the first data byte is written and the end of the message, the Reset Transmit Underrun/EOM command must be issued. Thus the Transmit Underrun/EOM status bit is in the reset state at the end of the message (when underrun occurs), which automatically sends the CRC characters. The sending of CRC again sets the Transmit/Underrun/EOM status bit.

Although there is no restriction as to when the Transmit Underrun/EOM bit can be reset within a message, it is usually reset after the first data character (secondary address) is sent to the Z80-SIO. Resetting this bit allows CRC and flags to be sent when there is no data to send which gives additional time to the CPU for recognizing the fault and responding with an abort command. By resetting it early in the message, the entire message has the maximum amount of CPU response time in an unintentional transmit underrun situation.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
WR3	00 = Rx 5 BITS/CHAR 10 = Rx 6 BITS/CHAR 01 = Rx 7 BITS/CHAR 11 = Rx 8 BITS/CHAR		AUTO ENABLES	ENTER HUNT MODE (IF INCOMING DATA NOT NEEDED)	Rx CRC ENABLE	ADDRESS SEARCH MODE	0	Rx ENABLE
WR4	0	0	1 0 SELECTS SDLC MODE		0	0	0	0
WR5	DTR	00 = Tx 5 BITS (OR LESS)/CHAR 10 = Tx 6 BITS/CHAR 01 = Tx 7 BITS/CHAR 11 = Tx 8 BITS/CHAR		0	Tx ENABLE	0 SELECTS SDLC CRC	RTS	Tx CRC ENABLE

Table 7. Contents of Write Registers 3, 4 and 5 in SDLC Modes

When the External/Status interrupt is set and while CRC is being sent, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset to indicate that the transmit register is full of CRC data. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin. This interrupt occurs because CRC has been sent and the flag has been loaded. If no more messages are to be sent, the program can terminate transmission by resetting $\overline{RTS}$, and disabling the transmitter.

In the SDLC mode, it is good practice to reset the Transmit Underrun/EOM status bit immediately after the first character is sent to the Z80-SIO. When the Transmit Underrun is detected, this ensures that the transmission time is filled by CRC characters, giving the CPU enough time to issue the Send Abort command. This also stops the flags from going on the line prematurely and eliminates the possibility of the receiver accepting the frame as valid data. The situation can happen because it is possible that—at the receiving end—the data pattern immediately preceding the automatic flag insertion could match the CRC checker, giving a false CRC check result. The External/Status interrupt is generated whenever the Transmit Underrun/EOM bit is set because of the Transmit Underrun condition.

The transmit underrun logic provides additional protection against premature flag insertion if the proper response is given to the Z80-SIO by the CPU interrupt service routine. The following example is given to clarify this point:

The Z80-SIO raises an interrupt with the Transmit Buffer Empty status bit set.

The CPU does not respond in time and causes a Transmit Underrun condition.

The Z80-SIO starts sending CRC characters (two bytes).

The CPU eventually satisfies the Transmit Buffer Empty interrupt with a data character that follows the CRC character being transmitted.

The Z80-SIO sets the External/Status interrupt with the Transmit Underrun/EOM status bit set.

The CPU recognizes the Transmit Underrun/EOM status and determines from its internal program status that the interrupt is not for "end of message".

The CPU immediately issues a Send Abort Command (WR0) to the Z80-SIO.

The Z80-SIO sends the Abort sequence by destroying whatever data (CRC, data or flag) is being sent.

This sequence illustrates that the CPU has a protection of 22 minimum and 30 maximum transmit clock cycles.

SDLC CRC Generation. The CRC generator must be reset to all 1's at the beginning of each frame before CRC accumulation can begin. Actual accumulation begins when the program sends the address field (eight bits) to the Z80-SIO. Although the Z80-SIO automatically transmits one flag character following the Transmit Enable, it may be wise to send a few more flag characters ahead of the message to ensure character synchronization at the receiving end. This can be done by externally timing out after enabling the transmitter and before loading the first character.

The Transmit CRC Enable (WR5, D_0) should be enabled prior to sending the address field. In the SDLC mode all the characters between the opening and closing flags are included in CRC accumulation, and the CRC generated in the Z80-SIO transmitter is inverted before it is sent on the line.

Transmit Termination. If the transmitter is disabled while a character is being sent, that character (data or flag) is sent in the normal fashion, but is followed by a marking line rather than CRC or flag characters.

A character in the buffer when the transmitter is disabled remains in the buffer; however, a programmed Abort sequence is effective as soon as it is written into the control register. Characters being transmitted, if any, are lost. In the case of CRC, the 16-bit transmission is completed if the transmitter is disabled; however, flags are sent in place of CRC.

In all modes, characters are sent with the least-significant bits first. This requires right-hand justification of data to be transmitted if the word length is less than eight bits. If the word length is five bits or less, the special technique described in the Write Register 5 section ("Z80-SIO Programming" chapter; "Write Registers" section) must be used.

Since the number of bits/character can be changed on the fly, the data field can be filled with any number of bits. When used in conjunction with the Receiver Residue codes, the Z80-SIO can receive a message that has a variable I-field and retransmit it exactly as received with no previous information about the character structure of the I-field (if any). A change in the number of bits does not affect the character in the process of being shifted out. Characters are sent with the number of bits programmed at the time that the character is loaded from the transmit buffer to the transmitter.

If the External/Status Interrupt Enable is set, transmitter conditions such as "starting to send CRC characters," "starting to send flag characters," and $\overline{CTS}$ changing state cause interrupts that have a unique vector if Status Affects Vector is set. All interrupts can be disabled for operation in a polled mode.

Table 8 shows the typical program steps that implement the half-duplex SDLC Transmit mode.

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER	*INFORMATION LOADED*	
INITIALIZE	WR0	CHANNEL RESET	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 3	
	WR3	AUTO ENABLES	Transmitter sends data only after $\overline{CTS}$ is detected.
	WR0	POINTER 4, RESET EXTERNAL/STATUS INTERRUPTS	
	WR4	PARITY INFORMATION, SDLC MODE, x1 CLOCK MODE	
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	
	WR1	EXTERNAL INTERRUPT ENABLE, STATUS AFFECTS VECTOR, TRANSMIT INTERRUPT ENABLE OR WAIT/READY MODE ENABLE	The External Interrupt mode monitors the status of the $\overline{CTS}$ and $\overline{DCD}$ inputs, as well as the status of Tx Underrun/EOM latch. Transmit Interrupt interrupts when the Transmit buffer becomes empty; the Wait/Ready mode can be used to transfer data on a DMA or Block Transfer basis. The first interrupt occurs when $\overline{CTS}$ becomes active, at which point flags are transmitted by the Z80-SIO. The first data byte (address field) can be loaded in the Z80-SIO after this interrupt. Flags cannot be sent to the Z80-SIO as data. Status Affects Vector used in Channel B only.
	WR0	POINTER 5	
	WR5	TRANSMIT CRC ENABLE, REQUEST TO SEND, SDLC-CRC, TRANSMIT ENABLE, TRANSMIT WORD LENGTH, DATA TERMINAL READY	SDLC-CRC mode must be defined before initializing transmit CRC generator.
	WR0	RESET TRANSMIT CRC GENERATOR	Initialize CRC generator to all 1's.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		Waiting for Interrupt or Wait/Ready output to transfer data.
DATA TRANSFER AND STATUS MONITORING	*WHEN INTERRUPT (WAIT/READY) OCCURS, THE CPU DOES THE FOLLOWING:* • CHANGES TRANSMIT WORD LENGTH (IF NECESSARY) • TRANSFERS DATA BYTE FROM CPU (MEMORY) TO SIO • RESETS Tx UNDERRUN/EOM LATCH (WR0)		Flags are transmitted by the SIO as soon as Transmit Enable is set and $\overline{CTS}$ becomes active. The $\overline{CTS}$ status change is the first interrupt that occurs and is followed by transmit buffer empty for subsequent transfers.
	IF LAST CHARACTER OF THE I-FIELD IS SENT, THE SIO DOES THE FOLLOWING: • SENDS CRC • SENDS CLOSING FLAG • INTERRUPTS CPU WITH BUFFER EMPTY STATUS *CPU DOES THE FOLLOWING:* • ISSUES RESET Tx INTERRUPT PENDING COMMAND TO THE Z80-SIO • UPDATES NS COUNT • REPEATS THE PROCESS FOR NEXT MESSAGE, ETC. *IF THE VECTOR INDICATES AN ERROR, THE CPU DOES THE FOLLOWING:* • SENDS ABORT • EXECUTES ERROR ROUTINE • UPDATES PARAMETERS, MODES, ETC. • RETURNS FROM INTERRUPT		Word length can be changed "on the fly" for variable I-field length. The data byte can contain address, control, or I-field information (never a flag). It is a good practice to reset Tx Underrun/EOM latch in the beginning of the message to avoid a false end-of-frame detection at the receiving end. This ensures that, when underrun occurs, CRC is transmitted and underrun interrupt (Tx Underrun/EOM latch active) occurs. Note that "Send Abort" can be issued to the SIO in response to any interrupting continuing to abort the transmission.
TERMINATION	REDEFINE INTERRUPT MODES UPDATE MODEM CONTROL OUTPUTS DISABLE TRANSMIT MODE		Terminate gracefully.

Table 8. SDLC Transmit Mode

SDLC Receive

INITIALIZATION

The SDLC Receive mode is initialized by the system with the following parameters: SDLC mode, x 1 clock mode, SDLC polynomial, receive word length, etc. The flag characters must also be loaded in WR7 and the secondary address field loaded in WR6. The receiver is enabled only after all the receive parameters have been set. After all this has been done, the receiver is in the Hunt phase and remains in this phase until the first flag is received. While in the SDLC mode, the receiver never re-enters the Hunt phase, unless specifically instructed to do so by the program. The WR4 parameters must be issued prior to the WR1, WR3, WR5, WR6 and WR7 parameters.

Under program control, the receiver can enter the Address Search mode. If the Address Search bit (WR3, D_2) is set, a character following the flag (first non-flag character) is compared against the programmed address in WR6 and the hardwired global address (11111111). If the SDLC frame address field matches either address, data transfer begins.

Since the Z80-SIO is capable of matching only one address character, extended address field recognition must be done by the CPU. In this case, the Z80-SIO simply transfers the additional address bytes to the CPU as if they were data characters. If the CPU determines that the frame does not have the correct address field, it can set the Hunt bit, and the Z80-SIO suspends reception and searches for a new message headed by a flag. Since the control field of the frame is transparent to the Z80-SIO, it is transferred to the CPU as a data character. Extra zeros inserted in the data stream are automatically deleted: flags are not transferred to the CPU.

DATA TRANSFER AND STATUS MONITORING

After receipt of a valid flag, the assembled characters are transferred to the receive data FIFO. The following four interrupt modes are available to transfer this data and its associated status.

No Interrupts Enabled. This mode is used for purely polled operations or for off-line conditions.

Interrupt On First Character Only. This mode is normally used to start a software polling loop or a Block Transfer instruction using $\overline{\text{WAIT/READY}}$ to synchronize the CPU or DMA device to the incoming data rate. In this mode, the Z80-SIO interrupts on the first character and thereafter only interrupts if Special Receive conditions are detected. The mode is reinitialized with the Enable Interrupt On Next Receive Character Command.

The first character received after this command is issued causes an interrupt. If External/Status interrupts are enabled, they may interrupt any time the $\overline{\text{DCD}}$ input changes state. Special Receive conditions such as End Of Frame and Receiver Overrun also cause interrupts. The End Of Frame interrupt can be used to exit the Block Transfer mode.

Interrupt On Every Character. An interrupt is generated whenever the receive FIFO contains a character. Error and Special Receive conditions generate a special vector if Status Affects Vector is selected.

Special Receive Condition Interrupts. The Special Receive Condition interrupt is not, as such, a separate interrupt mode. Before the Special Receive condition can cause an interrupt, either Interrupt On First Receive Character Only or Interrupt On Every Character must be selected. The Special Receive Condition interrupt is caused by a Receive Overrun or End Of Frame detection. Since the Receive Overrun status bit is latched, the error status read reflects an error in the current word in the receive buffer in addition to any errors received since the last Error Reset command. The Receive Overrun status bit can only be reset by the Error Reset command. The End Of Frame status bit indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid.

Character length may be changed on the fly. If the address and control bytes are processed as 8-bit characters, the receiver may be switched to a shorter character length during the time that the first information character is being assembled. This change must be made fast enough so it is effective before the number of bits specified for the character length have been assembled. For example, if the change is to be from the 8-bit control field to a 7-bit information field, the change must be made *before* the first seven bits of the I-field are assembled.

SDLC Receive CRC Checking. Control of the receive CRC checker is automatic. It is reset by the leading flag and CRC is calculated up to the final flag. The byte that has the End Of Frame bit set is the byte that contains the result of the CRC check. If the CRC/Framing Error bit is not set, the CRC indicates a valid message. A special check sequence is used for the SDLC check because the transmitted CRC check is inverted. The final check must be 0001110100001111. The 2-byte CRC check characters must be read by the CPU and discarded because the Z80-SIO, while using them for CRC checking, treats them as ordinary data.

SDLC Receive Termination. If enabled, a special vector is generated when the closing flag is received. This signals that the byte with the End Of Frame bit set has been received. In addition to the results of the CRC check, RR1 has three bits of Residue code valid at this time. For those cases in which the number of bits in the I-field is not an integral multiple of the character length used, these bits indicate the boundary between the CRC check bits and the I-field bits. For a detailed description of the meaning of these bits, see the description of the residue codes in RR1 under "Z80-SIO Programming."

Any frame can be prematurely aborted by an Abort sequence. Aborts are detected if seven or more 1's occur

and cause an External/Status interrupt (if enabled) with the Break/Abort bit in RR0 set. After the Reset External/Status interrupts command has been issued a second interrupt occurs when the continuous 1's condition has been cleared. This can be used to distinguish between the Abort and Idle line conditions.

Unlike the synchronous mode, CRC calculation in SDLC does not have an 8-bit delay since all the characters are included in CRC calculation. When the second CRC character is loaded into the receive buffer, CRC calculation is complete.

Table 9 shows the typical steps required to implement a half-duplex SDLC receive mode. The complete set of command and status bit definitions is found in the next section.

FUNCTION	TYPICAL PROGRAM STEPS		COMMENTS
	REGISTER:	*INFORMATION LOADED:*	
	WR0	CHANNEL 2	Reset SIO
	WR0	POINTER 2	
	WR2	INTERRUPT VECTOR	Channel B only
	WR0	POINTER 4	
	WR4	PARITY INFORMATION, SYNC MODE, SDLC MODE, x1 CLOCK MODE	
	WR0	POINTER 5, RESET EXTERNAL/STATUS INTERRUPTS	
	WR5	SDLC-CRC, DATA TERMINAL READY	
	WR0	POINTER 3	
	WR3	RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES, RECEIVE CHARACTER LENGTH, ADDRESS SEARCH MODE	'Auto Enables' enables the receiver to accept data only after $\overline{DCD}$ becomes active. Address Search Mode enables SIO to match the message address with the programmed address or the global address.
	WR0	POINTER 6	
INITIALIZE	WR6	SECONDARY ADDRESS FIELD	This address is matched against the message address in an SDLC poll operation.
	WR0	POINTER 7	
	WR7	SDLC FLAG 01111110	This flag detects the start and end of frame in an SDLC operation.
	WR0	POINTER 1, RESET EXTERNAL/STATUS INTERRUPTS	In this interrupt mode, only the Address Field (1 character only) is transferred to the CPU. All subsequent fields (Control, Information, etc.) are transferred on a DMA basis. Status Affects Vector in Channel B only.
	WR1	STATUS AFFECTS VECTOR, EXTERNAL INTERRUPT ENABLE, RECEIVE INTERRUPT ON FIRST CHARACTER ONLY.	
	WR0	POINTER 3, ENABLE INTERRUPT ON NEXT RECEIVE CHARACTER	Used to provide simple loop-back entry point for next transaction.
	WR3	RECEIVE ENABLE, RECEIVE CRC ENABLE, ENTER HUNT MODE, AUTO ENABLES, RECEIVER CHARACTER LENGTH, ADDRESS SEARCH MODE	WR3 reissued to enable receiver.
IDLE MODE	EXECUTE HALT INSTRUCTION OR SOME OTHER PROGRAM		SDLC Receive Mode is fully initialized and SIO is waiting for the opening flag followed by a matching address field to interrupt the CPU.

Table 9. SDLC Receive Mode

FUNCTION	TYPICAL PROGRAM STEPS	COMMENTS
	WHEN INTERRUPT ON FIRST CHARACTER OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS DATA BYTE (ADDRESS BYTE) TO CPU • DETECTS AND SETS APPROPRIATE FLAG FOR EXTENDED ADDRESS FIELD • UPDATES POINTERS AND PARAMETERS • ENABLES DMA CONTROLLER • ENABLES WAIT/READY FUNCTION IN SIO • RETURNS FROM INTERRUPT	During the Hunt phase, the SIO interrupts when the programmed address matches the message address. The CPU establishes the DMA mode and all subsequent data characters are transferred by the DMA controller to memory.
	WHEN THE READY OUTPUT BECOMES ACTIVE, THE DMA CONTROLLER DOES THE FOLLOWING: • TRANSFERS THE DATA BYTE TO MEMORY • UPDATES THE POINTERS	During the DMA operation, the SIO monitors the $\overline{DCD}$ input and the Abort sequence in the data stream to interrupt the CPU with External Status error. The Special Receive condition interrupt is caused by Receive Overrun error.
DATA TRANSFER AND STATUS MONITORING	*WHEN END OF FRAME INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING:* • EXITS DMA MODE (DISABLES WAIT/READY) • TRANSFERS RR1 TO THE CPU • CHECKS THE CRC ERROR BIT STATUS AND RESIDUE CODES • UPDATES NR COUNT • ISSUES 'ERROR RESET' COMMAND TO SIO	Detection of End of Frame (Flag) causes interrupt and deactivates the Wait/Ready function. Residue codes indicate the structure of the last two bytes of the message, which were transferred to memory under DMA. 'Error Reset' is issued to clear the special condition.
	WHEN 'ABORT SEQUENCE DETECTED' INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • TRANSFERS RR0 TO THE CPU • EXITS DMA MODE • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO • ENTERS THE IDLE MODE	Abort sequence is detected when seven or more 1's are found in the data stream. CPU is waiting for Abort Sequence to terminate. Termination clears the Break/Abort status bit and causes interrupt.
	WHEN THE SECOND ABORT SEQUENCE INTERRUPT OCCURS, THE CPU DOES THE FOLLOWING: • ISSUES THE RESET EXTERNAL STATUS INTERRUPT COMMAND TO THE SIO	At this point, the program proceeds to terminate this message.
TERMINATION	REDEFINE INTERRUPT MODES, SYNC MODE AND SDLC MODES DISABLE RECEIVE MODE	

Table 9. SDLC Receive Mode (Continued)

Z80-SIO Programming

To program the Z80-SIO, the system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity are first set, then the interrupt mode and, finally, receiver or transmitter enable. The WR4 parameters must be issued before any other parameters are issued in the initialization routine.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/$\overline{A}$) and the Control/Data input (C/$\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus. Figure 14 illustrates the timing relationships for programming the write registers, and transferring data and status.

C/$\overline{D}$	B/$\overline{A}$	Function
0	0	Channel A Data
0	1	Channel B Data
1	0	Channel A Commands/Status
1	1	Channel B Commands/Status

Write Registers

The Z80-SIO contains eight registers (WR0–WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0–D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z80-SIO.

Note that the programmer has complete freedom, after pointing to the selected register, of either reading to test the read register or writing to initialize the write register. By designing software to initialize the Z80-SIO in a modular and structured fashion, the programmer can use powerful block I/O instructions.

WR0 is a special case in that all the basic commands (CMD_0–CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0–D_2 to point to WR0.

The basic commands (CMD_0–CMD_2) and the CRC controls (CRC_0, CRC_1) are contained in the first byte of any write register access. This maintains maximum flexibility and system control. Each channel contains the following control registers. These registers are addressed as commands (not data).

WRITE REGISTER 0

WR0 is the command register; however, it is also used for CRC reset codes and to point to the other registers.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code 1	CRC Reset Code 0	CMD 2	CMD 1	CMD 0	PTR 2	PTR 1	PTR 0

Pointer Bits (D_0–D_2). Bits D_0–D_2 are pointer bits that determine which other write register the next byte is to be written into or which read register the next byte is to be read from. The first byte written into each channel after a reset (either by a Reset command or by the external reset input) goes into WR0. Following a read or write to any register (except WR0), the pointer will point to WR0.

Command Bits (D_3–D_5). Three bits, D_3–D_5, are encoded to issue the seven basic Z80-SIO commands.

Command	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no effect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Enable Interrupt on next Rx Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A)

Command 0 (Null). The Null command has no effect. Its normal use is to cause the Z80-SIO to do nothing while the pointers are set for the following byte.

Command 1 (Send Abort). This command is used only with the SDLC mode to generate a sequence of eight to thirteen 1's.

Command 2 (Reset External/Status Interrupts). After an External/Status interrupt (a change on a modem line or a break condition, for example), the status bits of RR0 are latched. This command re-enables them and allows interrupts to occur again. Latching the status bits captures short pulses until the CPU has time to read the change.

Command 3 (Channel Reset). This command performs the same function as an External Reset, but only on a single channel. Channel A Reset also resets the interrupt prioritization logic. All control registers for the channel must be rewritten after a Channel Reset command.

After a Channel Reset, four extra system clock cycles should be allowed for Z80-SIO reset time before any additional commands or controls are written into that channel. This can normally be the time used by the CPU to fetch the next op code.

Command 4 (Enable Interrupt On Next Receive Character). If the Interrupt On First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the Z80-SIO for the next message.

Command 5 (Reset Transmitter Interrupt Pending). The transmitter interrupts when the transmit buffer becomes empty if the Transmit Interrupt Enable mode is selected. In those cases where there are no more characters to be sent (at the end of message, for example), issuing this command prevents further transmitter interrupts until after the next character has been loaded into the transmit buffer or until CRC has been completely sent.

Command 6 (Error Reset). This command resets the error latches. Parity and Overrun errors are latched in RR1 until they are reset with this command. With this scheme, parity errors occurring in block transfers can be examined at the end of the block.

Command 7 (Return From Interrupt). This command must be issued in Channel A and is interpreted by the Z80-SIO in exactly the same way it would interpret an RETI command on the data bus. It resets the interrupt-under-service latch of the highest-priority internal device under service and thus allows lower priority devices to interrupt via the daisy chain. This command allows use of the internal daisy chain even in systems with no external daisy chain or RETI command.

CRC Reset Codes 0 and 1 (D_6 and D_7). Together, these bits select one of the three following reset commands:

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (no affect)
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Tx Underrun/End Of Message latch

The Reset Transmit CRC Generator command normally initializes the CRC generator to all 0's. If the SDLC mode is selected, this command initializes the CRC generator to all 1's. The Receive CRC checker is also initialized to all 1's for the SDLC mode.

WRITE REGISTER 1

WR1 contains the control bits for the various interrupt and Wait/Ready modes.

D_7	D_6	D_5	D_4
Wait/Ready Enable	$\overline{\text{Wait}}$ Or Ready Function	Wait/Ready On Receive/ Transmit	Receive Interrupt Mode 1
D_3	D_2	D_1	D_0
Receive Interrupt Mode 0	Status Affects Vector	Transmit Interrupt Enable	External Interrupts Enable

External/Status Interrupt Enable (D_0). The External/Status Interrupt Enable allows interrupts to occur as a result of transitions on the $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ or $\overline{\text{SYNC}}$ inputs, as a result of a Break/Abort detection and termination, or at the beginning of CRC or sync character transmission when the Transmit Underrun/EOM latch becomes set.

Transmitter Interrupt Enable (D_1). If enabled, interrupts occur whenever the transmitter buffer becomes empty.

Status Affects Vector (D_2). This bit is active in Channel B only. If this bit is not set, the fixed vector programmed in WR2 is returned from an interrupt acknowledge sequence. If this bit is set, the vector returned from an interrupt acknowledge is variable according to the following interrupt conditions:

	V_3	V_2	V_1	
Ch B	0	0	0	Ch B Transmit Buffer Empty
	0	0	1	Ch B External/Status Change
	0	1	0	Ch B Receive Character Available
	0	1	1	Ch B Special Receive Condition*
Ch A	1	0	0	Ch A Transmit Buffer Empty
	1	0	1	Ch A External/Status Change
	1	1	0	Ch A Receive Character Available
	1	1	1	Ch A Special Receive Condition*

*Special Receive Conditions: Parity Error, Rx Overrun Error, Framing Error, End Of Frame (SDLC).

Receive Interrupt Modes 0 and 1 (D_3 and D_4). Together these two bits specify the various character-available conditions. In Receive Interrupt modes 1, 2 and 3, a Special Receive Condition can cause an interrupt and modify the interrupt vector.

D_4 Receive Interrupt Mode 1	D_3 Receive Interrupt Mode 0	
0	0	0. Receive Interrupts Disabled
0	1	1. Receive Interrupt On First Character Only
1	0	2. Interrupt On All Receive Characters—parity error is a Special Receive condition
1	1	3. Interrupt On All Receive Characters—parity error is not a Special Receive condition

Wait/Ready Function Selection (D_5–D_7). The Wait and Ready functions are selected by controlling D_5, D_6, and D_7. Wait/Ready function is enabled by setting Wait/Ready Enable (WR1, D_7) to 1. The Ready function is selected by setting D_6 (Wait/Ready function) to 1. If this bit is 1, the $\overline{\text{WAIT/READY}}$ output switches from High to Low when the Z80-SIO is ready to transfer data. The Wait function is selected by setting D_6 to 0. If this bit is

0, the $\overline{\text{WAIT/READY}}$ output is in the open-drain state and goes Low when active.

Both the Wait and Ready functions can be used in either the Transmit or Receive modes, but not both simultaneously. If D_5 (Wait/Ready on Receive/Transmit) is set to 1, the Wait/Ready function responds to the condition of the receive buffer (empty or full). If D_5 is set to 0, the Wait/Ready function responds to the condition of the transmit buffer (empty or full).

The logic states of the $\overline{\text{WAIT/READY}}$ output when active or inactive depend on the combination of modes selected. Following is a summary of these combinations:

If $D_7 = 0$	
And $D_5 = 1$	And $D_6 = 0$
$\overline{\text{READY}}$ is High	$\overline{\text{WAIT}}$ is floating

If $D_7 = 1$			
And $D_5 = 0$		And $D_5 = 1$	
$\overline{\text{READY}}$	Is High when transmit buffer is full.	$\overline{\text{READY}}$	Is High when receive buffer is empty.
$\overline{\text{WAIT}}$	Is Low when transmit buffer is full and an SIO data port is selected.	$\overline{\text{WAIT}}$	Is Low when receive buffer is empty and an SIO data port is selected.
$\overline{\text{READY}}$	Is Low when transmit buffer is empty.	$\overline{\text{READY}}$	Is Low when receive buffer is full.
$\overline{\text{WAIT}}$	Is floating when transmit buffer is empty.	$\overline{\text{WAIT}}$	Is floating when receive buffer is full.

The $\overline{\text{WAIT}}$ output High-to-Low transition occurs with the delay time $t_{DIC(WR)}$ after the I/O request. The Low-to-High transition occurs with the delay $t_{DHC(WR)}$ from the falling edge of ϕ. The $\overline{\text{READY}}$ output High-to-Low transition occurs with the delay $t_{DLC(WR)}$ from the rising edge of ϕ. The $\overline{\text{READY}}$ output Low-to-High transition occurs with the delay $t_{DIC(WR)}$ after $\overline{\text{IORQ}}$ falls.

The Ready function can occur any time the Z80-SIO is not selected. When the $\overline{\text{READY}}$ output becomes active (Low), the DMA controller issues $\overline{\text{IORQ}}$ and the corresponding B/$\overline{\text{A}}$ and C/$\overline{\text{D}}$ inputs to the Z80-SIO to transfer data. The $\overline{\text{READY}}$ output becomes inactive as soon as $\overline{\text{IORQ}}$ and $\overline{\text{CS}}$ become active. Since the Ready function can occur internally in the Z80-SIO whether it is addressed or not, the $\overline{\text{READY}}$ output becomes inactive when any CPU data or command transfer takes place. This does not cause problems because the DMA controller is not enabled when the CPU transfer takes place.

The Wait function—on the other hand—is active only if the CPU attempts to read Z80-SIO data that has not yet been received, which occurs frequently when block transfer instructions are used. The Wait function can also become active (under program control) if the CPU tries to write data while the transmit buffer is still full. The fact that the $\overline{\text{WAIT}}$ output for either channel can become active when the opposite channel is addressed (because the Z80-SIO is addressed) does not affect operation of software loops or block move instructions.

WRITE REGISTER 2

WR2 is the interrupt vector register; it exists in Channel B only. V_4–V_7 and V_0 are always returned exactly as written; V_1–V_3 are returned as written if the Status Affects Vector (WR1, D_2) control bit is 0. If this bit is 1, they are modified as explained in the previous section.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

WRITE REGISTER 3

WR3 contains receiver logic control bits and parameters.

D_7	D_6	D_5	D_4
Receiver Bits/ Char 1	Receiver Bits/ Char 0	Auto Enables	Enter Hunt Phase

D_3	D_2	D_1	D_0
Receiver CRC Enable	Address Search Mode	Sync Char Load Inhibit	Receiver Enable

Receiver Enable (D_0). A 1 programmed into this bit allows receive operations to begin. This bit should be set only after all other receive parameters are set and receiver is completely initialized.

Sync Character Load Inhibit (D_1). Sync characters preceding the message (leading sync characters) are not loaded into the receive buffers if this option is selected. Because CRC calculations are not stopped by sync character stripping, this feature should be enabled only at the beginning of the message.

Address Search Mode (D_2). If SDLC is selected, setting this mode causes messages with addresses not matching the programmed address in WR6 or the global (11111111) address to be rejected. In other words, no receive interrupts can occur in the Address Search mode unless there is an address match.

Receiver CRC Enable (D_3). If this bit is set, CRC calculation starts (or restarts) at the beginning of the last character transferred from the receive shift register to the buffer stack, regardless of the number of characters in the stack. See "SDLC Receive CRC Checking" (SDLC Receive section) and "CRC Error Checking" (Synchronous Receive section) for details regarding when this bit should be set.

Enter Hunt Phase (D_4). The Z80-SIO automatically enters the Hunt phase after a reset; however, it can be re-entered if character synchronization is lost for any reason (Synchronous mode) or if the contents of an incoming message are not needed (SDLC mode). The Hunt phase is re-entered by writing a 1 into bit D_4. This sets the Sync/Hunt bit (D_4) in RR0.

Auto Enables (D_5). If this mode is selected, $\overline{DCD}$ and $\overline{CTS}$ become the receiver and transmitter enables, respectively. If this bit is not set, $\overline{DCD}$ and $\overline{CTS}$ are simply inputs to their corresponding status bits in RR0.

Receiver Bits/Characters 1 and 0 (D_7 and D_6). Together, these bits determine the number of serial receive bits assembled to form a character. Both bits may be changed during the time that a character is being assembled, but they must be changed before the number of bits currently programmed is reached.

D_7	D_6	Bits/Character
0	0	5
0	1	7
1	0	6
1	1	8

WRITE REGISTER 4

WR4 contains the control bits that affect both the receiver and transmitter. In the transmit and receive initialization routine, these bits should be set before issuing WR1, WR3, WR5, WR6, and WR7.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate 1	Clock Rate 0	Sync Modes 1	Sync Modes 0	Stop Bits 1	Stop Bits 0	Parity Even/ $\overline{Odd}$	Parity

Parity (D_0). If this bit is set, an additional bit position (in addition to those specified in the bits/character control) is added to transmitted data and is expected in receive data. In the Receive mode, the parity bit received is transferred to the CPU as part of the character, unless 8 bits/character is selected.

Parity Even/$\overline{Odd}$ (D_1). If parity is specified, this bit determines whether it is sent and checked as even or odd (1 = even).

Stop Bits 0 and 1 (D_2 and D_3). These bits determine the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit. A special mode (00) signifies that a synchronous mode is to be selected.

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync modes
0	1	1 stop bit per character
1	0	1½ stop bits per character
1	1	2 stop bits per character

Sync Modes 0 and 1 (D_4 and D_5). These bits select the various options for character synchronization.

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDLC mode (01111110 flag pattern)
1	1	External Sync mode

Clock Rate 0 and 1 (D_6 and D_7). These bits specify the multiplier between the clock ($\overline{TxC}$ and $\overline{RxC}$) and data rates. For synchronous modes, the × 1 clock rate must be specified. Any rate may be specified for asynchronous modes; however, the same rate must be used for both the receiver and transmitter. The system clock in all modes must be at least 4.5 times the data rate. If the × 1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate × 1 = Clock Rate
0	1	Data Rate × 16 = Clock Rate
1	0	Data Rate × 32 = Clock Rate
1	1	Data Rate × 64 = Clock Rate

WRITE REGISTER 5

WR5 contains control bits that affect the operation of transmitter, with the exception of D_2, which affects the transmitter and receiver.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Tx Bits/ Char 1	Tx Bits/ Char 0	Send Break	Tx Enable	CRC-16/ $\overline{SDLC}$	RTS	Tx CRC Enable

Transmit CRC Enable (D_0). This bit determines if CRC is calculated on a particular transmit character. If it is set at the time the character is loaded from the transmit buffer into the transmit shift register, CRC is calculated on the character. CRC is not automatically sent unless this bit is set when the Transmit Underrun condition exists.

Request To Send (D_1). This is the control bit for the $\overline{RTS}$ pin. When the $\overline{RTS}$ bit is set, the $\overline{RTS}$ pin goes Low; when reset, $\overline{RTS}$ goes High. In the Asynchronous mode, $\overline{RTS}$ goes High only after all the bits of the character are transmitted and the transmitter buffer is empty. In Synchronous modes, the pin directly follows the state of the bit.

CRC-16/$\overline{SDLC}$ (D_2). This bit selects the CRC polynomial used by both the transmitter and receiver. When set, the CRC-16 polynomial ($X^{16} + X^{15} + X^2 + 1$) is used; when reset the SDLC polynomial ($X^{16} + X^{12} + X^5 + 1$) is used. If the SDLC mode is selected, the CRC generator and checker are preset to all 1's and a special check sequence is used. The SDLC CRC polnomial must be selected when the SDLC mode is selected. If the SDLC mode is not selected, the CRC generator and checker are preset to all 0's (for both polynomials).

Transmit Enable (D_3). Data is not transmitted until this bit is set, and the Transmit Data output is held marking. Data or sync characters in the process of being transmitted are completely sent if this bit is reset after transmission has started. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC.

Send Break (D_4). When set, this bit immediately forces the Transmit Data output to the spacing condition, regardless of any data being transmitted. When reset, TxD returns to marking.

Transmit Bits/Characters 0 and 1 (D_5 and D_6). Together, D_6 and D_5 control the number of bits in each byte transferred to the transmit buffer.

D_6 Transmit Bits/ Character 1	D_5 Transmit Bits/ Character 0	Bits/Character
0	0	Five or less
0	1	7
1	0	6
1	1	8

Bits to be sent must be right justified, least-significant bits first. The Five Or Less mode allows transmission of one to five bits per character; however, the CPU should format the data character as shown in the following table.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	Sends one data bit
1	1	1	0	0	0	D	D	Sends two data bits
1	1	0	0	0	D	D	D	Sends three data bits
1	0	0	0	D	D	D	D	Sends four data bits
0	0	0	D	D	D	D	D	Sends five data bits

Data Terminal Ready (D_7). This is the control bit for the $\overline{DTR}$ pin. When set, $\overline{DTR}$ is active (Low); when reset, $\overline{DTR}$ is inactive (High).

WRITE REGISTER 6

This register is programmed to contain the transmit sync character in the Monosync mode, the first eight bits of a 16-bit sync character in the Bisync mode, or a transmit sync character in the External Sync mode. In the SDLC mode, it is programmed to contain the secondary address field used to compare against the address field of the SDLC frame.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 7	Sync 6	Sync 5	Sync 4	Sync 3	Sync 2	Sync 1	Sync 0

WRITE REGISTER 7

This register is programmed to contain the receive sync character in the Monosync mode, a second byte (last eight bits) of a 16-bit sync character in the Bisync mode, or a flag character (01111110) in the SDLC mode. WR7 is not used in the External Sync mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Sync 15	Sync 14	Sync 13	Sync 12	Sync 11	Sync 10	Sync 9	Sync 8

Read Registers

The Z80-SIO contains three registers, RR0–RR2 (Figure 10), that can be read to obtain the status information for each channel (except for RR2—Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

READ REGISTER 0

This register contains the status of the receive and transmit buffers; the $\overline{DCD}$, $\overline{CTS}$ and $\overline{SYNC}$ inputs; the Transmit Underrun/EOM latch; and the Break/Abort latch.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Transmit Underrun/ EOM	CTS	Sync/ Hunt	DCD	Transmit Buffer Empty	Interrupt Pending (Ch. A only)	Receive Character Available

Receive Character Available (D_0). This bit is set when at least one character is available in the receive buffer; it is reset when the receive FIFO is completely empty.

Interrupt Pending (D_1). Any interrupting condition in the Z80-SIO causes this bit to be set; however, it is readable only in Channel A. This bit is mainly used in applications that do not have vectored interrupts available. During the interrupt service routine in these applications, this bit indicates if any interrupt conditions are present in the Z80-SIO. This eliminates the need for analyzing all the bits of RR0 in both Channels A and B. Bit D_1 is reset when all the interrupting conditions are satisfied. This bit is always 0 in Channel B.

Transmit Buffer Empty (D_2). This bit is set whenever the transmit buffer becomes empty, except when a CRC character is being sent in a synchronous or SDLC mode. The bit is reset when a character is loaded into the transmit buffer. This bit is in the set condition after a reset.

Data Carrier Detect (D_3). The DCD bit shows the state of the $\overline{DCD}$ input at the time of the last change of any of the five External/Status bits (DCD, CTS, Sync/Hunt, Break/Abort or Transmit Underrun/EOM). Any transition of the $\overline{DCD}$ input causes the DCD bit to be latched

and causes an External/Status interrupt. To read the current state of the DCD bit, this bit must be read immediately following a Reset External/Status Interrupt command.

Sync/Hunt (D_4). Since this bit is controlled differently in the Asynchronous, Synchronous and SDLC modes, its operation is somewhat more complex than that of the other bits and therefore requires more explanation.

In asynchronous modes, the operation of this bit is similar to the DCD status bit, except that Sync/Hunt shows the state of the $\overline{SYNC}$ input. Any High-to-Low transition on the $\overline{SYNC}$ pin sets this bit and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the $\overline{SYNC}$ pin at the time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the $\overline{SYNC}$ input.

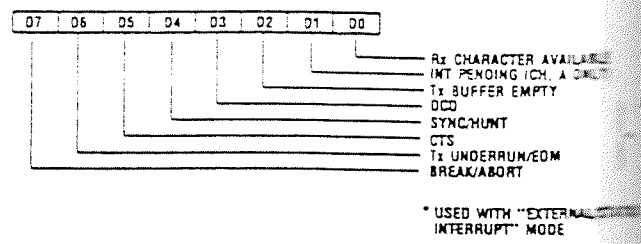
In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the $\overline{SYNC}$ input must be held High by the external logic until external character synchronization is achieved. A High at the $\overline{SYNC}$ input holds the Sync/Hunt status bit in the reset condition.

When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. Once $\overline{SYNC}$ is forced Low, it is a good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Refer to Figure 18 for timing details. The High-to-Low transition of the $\overline{SYNC}$ input sets the Sync/Hunt bit, which—in turn—sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt command.

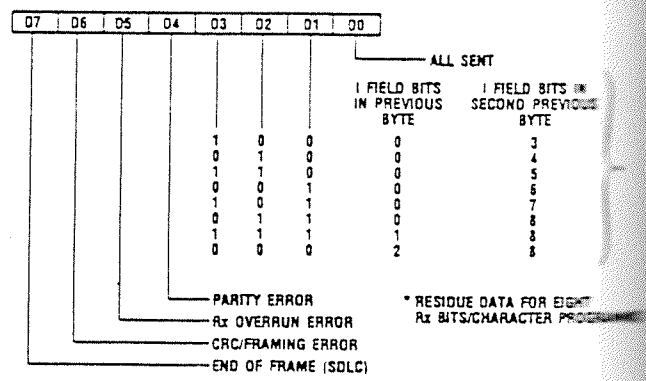
When the $\overline{SYNC}$ input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the Z80-SIO again looks for a High-to-Low transition on the $\overline{SYNC}$ input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the Z80-SIO is waiting for $\overline{SYNC}$ to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the Z80-SIO establishes character synchronization. The

READ REGISTER 0



READ REGISTER 1 †



READ REGISTER 2

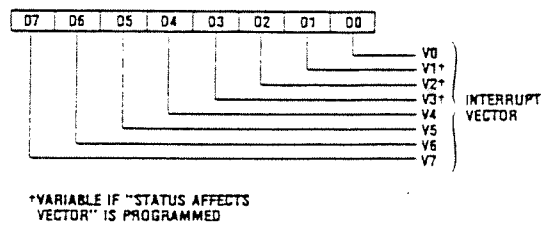


Figure 10. Read Register Bit Functions

High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the Z80-SIO to detect the next transition of other External/Status bits.

When the CPU detects the end of message or that character synchronization is lost, it sets the Enter Hunt Mode control bit, which—in turn—sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status interrupt, which must also be cleared by the Reset External/Status Interrupt command. Note that the $\overline{SYNC}$ pin acts as an output in this mode and goes Low every time a sync pattern is detected in the data stream.

In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit, or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the Z80-SIO. The External/Status interrupt is also generated, and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The Z80-SIO automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the receiver.

Clear To Send (D_5). This bit is similar to the DCD bit, except that it shows the inverted state of the $\overline{CTS}$ pin.

Transmit Underrun/End Of Message (D_6). This bit is in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D_5 and D_7). When the Transmit Underrun condition occurs, this bit is set; its becoming set causes the External/Status interrupt, which must be reset by issuing the Reset External/Status Interrupt command bits (WR0). This status bit plays an important role in conjunction with other control bits in controlling a transmit operation. Refer to "Bisync Transmit Underrun" and "SDLC Transmit Underrun" for additional details.

Break/Abort (D_7). In the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when Break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, CMD_2) to the break detection logic so the Break sequence termination can be recognized.

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

READ REGISTER 1

This register contains the Special Receive condition status bits and Residue codes for the I-field in the SDLC Receive Mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End Of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

All Sent (D_0). In asynchronous modes, this bit is set when all the characters have completely cleared the transmitter. Transitions of this bit do not cause interrupts. It is always set in synchronous modes.

Residue Codes 0, 1 and 2 (D_1–D_3). In those cases of the SDLC receive mode where the I-field is not an integral multiple of the character length, these three bits indicate the length of the I-field. These codes are meaningful only for the transfer in which the End Of Frame bit is set (SDLC). For a receive character length of eight bits per character, the codes signify the following:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Field bits are right-justified in all cases.

If a receive character length different from eight bits is used for the I-field, a table similar to the previous one may be constructed for each different character length. For no residue (that is, the last character boundary coincides with the boundary of the I-field and CRC field), the Residue codes are:

Bits per Character	Residue Code 2	Residue Code 1	Residue Code 0
8 Bits per Character	0	1	1
7 Bits per Character	0	0	0
6 Bits per Character	0	1	0
5 Bits per Character	0	0	1

Parity Error (D_4). When parity is enabled, this bit is set for those characters whose parity does not match the programmed sense (even/odd). The bit is latched, so once an error occurs, it remains set until the Error Reset command (WR0) is given.

Receive Overrun Error (D_5). This bit indicates that more than three characters have been received without a read from the CPU. Only the character that has been written over is flagged with this error, but when this character is read, the error condition is latched until reset by the Error Reset command. If Status Affects Vector is enabled, the character that has been overrun interrupts with a Special Receive Condition vector.

CRC/Framing Error (D_6). If a Framing Error occurs (asynchronous modes), this bit is set (and not latched) for the receive character in which the Framing Error occurred. Detection of a Framing Error adds an additional one-half of a bit time to the character time so the Framing Error is not interpreted as a new start bit. In synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the appropriate check value. This bit is reset by issuing an Error Reset command. The bit is not latched, so it is always updated when the next character is received. When used for CRC error and status in synchronous modes, it is usually set since most bit combinations result in a non-zero CRC except for a correctly completed message.

End Of Frame (D_7). This bit is used only with the SDLC mode and indicates that a valid ending flag has been received and that the CRC Error and Residue codes are also valid. This bit can be reset by issuing the Error Reset command. It is also updated by the first character of the following frame.

READ REGISTER 2 (Ch. B Only)

This register contains the interrupt vector written into WR2 if the Status Affects Vector control bit is not set. If the control bit is set, it contains the modified vector shown in the Status Affects Vector paragraph of the Write Register 1 section. When this register is read, the vector returned is modified by the highest priority interrupting condition at the time of the read. If no interrupts are pending, the vector is modified with $V_3 = 0$, $V_2 = 1$ and $V_1 = 1$. This register may be read only through Channel B.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3 Variable if Status Affects Vector is enabled	V_2	V_1	V_0

A P P E N D I X E

EXTERNAL CABLES & TEST PLUGS

NORSK DATA A.S

PIOC
Test Plugs

WIRE NO.	SIGNAL	POLARITY	EUROPLUG NO.	PLUGPANEL 25-pins "D" Female PIN NO.	DEVICE PLUG PIN NO.	
1	CHASSIS		c 1	1		
2			a 1	14		
3	TXD		c 2	2		
4	TXC		a 2	15		
5	RXD		c 3	3		
6			a 3	16		
7	RTS		c 4	4		
8	RXC		a 4	17		V - Level Testplug
9	RFS		c 5	5		
10	UTC		a 5	18		
11	DSR		c 6	6		
12	VLEV		a 6	19		
13	GND		c 7	7		
14	DTR		a 7	20		
15	SIGN.DET.		c 8	8		
16			a 8	21		
17			c 9	1		
18	TB		a 9	14		
19	TA		c 10	2		
20	CB		a 10	15		
21	CA		c 11	3		
22	RB		a 11	16		
23	RA		c 12	4		
24	IA		a 12	17		X - Level Testplug
25	IB		c 13	5		
26	SB		a 13	18		
27	SA		c 14	6		
28	UB		a 14	19		
29	UA		c 15	7		
30	YB		a 15	20		
31	GND		c 16	8		
32	YA		a 16	21		

EXTERNAL CABLE TYPE : 64 wire flatcable
EXTERNAL CABLE TYPE 1 :
EXTERNAL CABLE TYPE 2 :
EXTERNAL CABLE TYPE 3 :
EXTERNAL CABLE TYPE 4 :

Drawn by	Remarks	Replacement for	Date
Approved		Replaced by	Date
Date			

NORSK DATA A.S

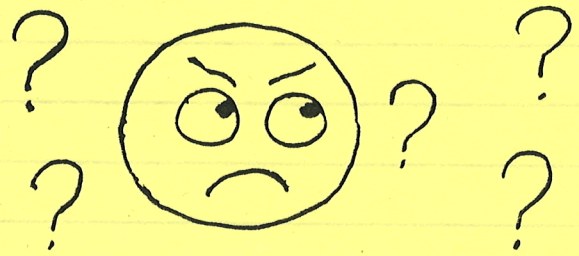
PIOC
External Cables

WIRE NO.	SIGNAL	POLARITY	EUROPLUG NO.	PLUGPANEL 25-pins "D" Female PIN NO.	DEVICE PLUG PIN NO.	
1	CHASSIS		c 1	1	1	PIOC TO V-TYPE MODEM 25 pin "D" partno. 325872
2			a 1	14		
3	TXD		c 2	2	2	
4	TXC		a 2	15	15	
5	RXD		c 3	3	3	
6			a 3	16		
7	RTS		c 4	4	4	
8	RXC		a 4	17	17	
9	RFS		c 5	5	5	
10	UTC		a 5	18	NC	
11	DSR		c 6	6	6	
12	VLEV		a 6	19	NC	
13	GND		c 7	7	7	
14	DTR		a 7	20	20	
15	SIGN.DET.		c 8	8	8	
16			a 8	21		
17			c 9	1		PIOC TO X-21 MODEM 15 PIN "D" partno. 325872
18	TB		a 9	14	9	
19	TA		c 10	2	2	
20	CB		a 10	15	10	
21	CA		c 11	3	3	
22	RB		a 11	16	11	
23	RA		c 12	4	4	
24	IA		a 12	17	12	
25	IB		c 13	5	5	
26	SB		a 13	18	13	
27	SA		c 14	6	6	
28	UB		a 14	19		
29	UA		c 15	7		
30	YB		a 15	20		
31	GND		c 16	8	8	
32	YA		a 16	21		
33	CHASSIS		c 17	1		
34			a 17	14		
35	TXD		c 18	2		
36			a 18	15		
37	RXD		c 19	3		
38			a 19	16		
39	RTS		c 20	4		
40			a 20	17		
41	RFS		c 21	5		
42			a 21	18		
43	DSR		c 22	6		
44			a 22	19		
45	GND		c 23	7		
46	DTR		a 23	20		
47	SIGN.DET.		c 24	8		
48			a 24	21		
49	CHASSIS		c 25	1		PIOC COMPUTER LINK 25 PIN "D" partno. 325233
50	TB		a 25	14	16	
51	TA		c 26	2	4	
52	CB		a 26	15	15	
53	CA		c 27	3	5	
54	RB		a 27	16	14	
55	RA		c 28	4	2	
56	IA		a 28	17	17	
57	IB		c 29	5	3	
58	SB		a 29	18	19	
59	SA		c 30	6	7	
60	UB		a 30	19	18	
61	UA		c 31	7	6	
62	YB		a 31	20	NC 20	
63	GND		c 32	8		
64	YA		a 32	21	NC 21	

EXTERNAL CABLE TYPE : 64 wire flatcable
EXTERNAL CABLE TYPE 1 :
EXTERNAL CABLE TYPE 2 :
EXTERNAL CABLE TYPE 3 :
EXTERNAL CABLE TYPE 4 :

Drawn by	Remarks	Replacement for	Date
Approved		Replaced by	Date
Date			

* * * * * * * * * * SEND US YOUR COMMENTS!!! * * * * * * * * * *



Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card - and an answer to your comments.

Please let us know if you

* find errors
* cannot understand information
* cannot find information
* find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!!

* * * * * * * * * HELP YOURSELF BY HELPING US!! * * * * * * * * *

Manual name: PIOC Hardware Reference Mnaual

Manual number: ND–02.003.01

What problems do you have? (use extra pages if needed) _____

Do you have suggestions for improving this manual? _____

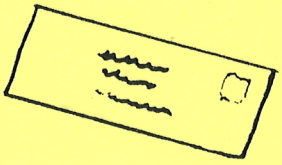
Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

What are you using this manual for? _____

Send to: Norsk Data A.S.
Documentation Department
P.O. Box 4, Lindeberg Gård
Oslo 10, Norway



Norsk Data's answer will be found on reverse side

Answer from Norsk Data _____

Answered by _____ Date _____

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway

Systems that put people first

NORSK DATA A.S OLAF HELSETHS VEI 5 P.O. BOX 25 BOGERUD 0621 OSLO 6 NORWAY
TEL.: 02 - 62 60 00 - TELEX: 18284 NDN