

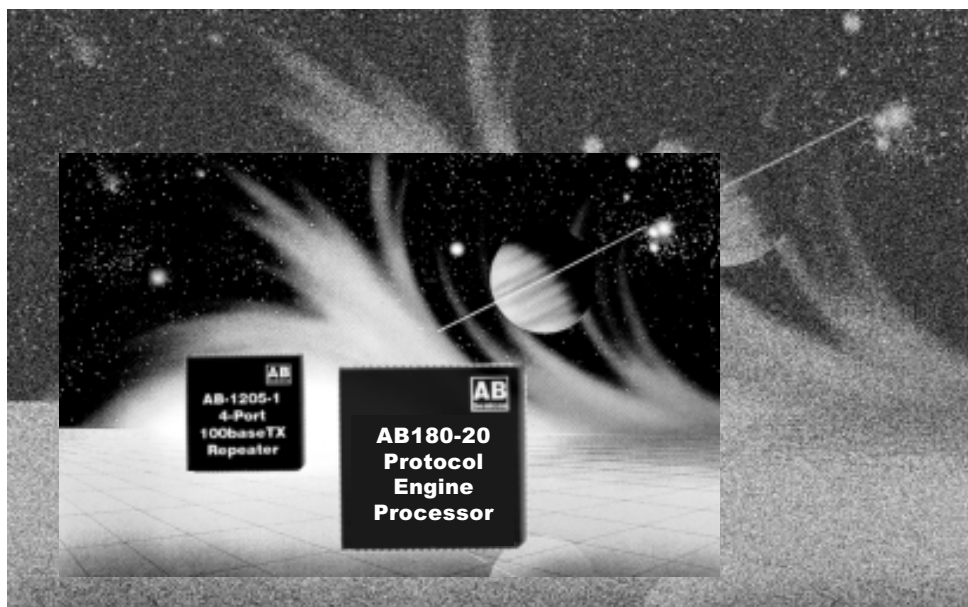


# AB180-20™

## 8-bit OCA Processor

for General Purpose  
and Protocol Engines

Combines High Performance and Low-Cost



## Product Specification

## **AB Semicon AB180-20™**

### **General Purpose and Protocol Engine Processor Product Specification**

For the latest information on the AB180-20, check the product specification on the AB Semicon web-site at:

<http://www.ab-semicon.com>

### **Copyright**

© Copyright 1999 AB Semicon Limited. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or any computer language, in any form or by any third party, without the prior written permission of AB Semicon Limited.

### **Disclaimer**

AB Semicon Limited reserves the right to revise this publication and to make changes from time to time to the contents hereof without obligation to notify any person or organization of such revision or changes. AB Semicon Limited has endeavoured to ensure that the information in this publication is correct, but will not accept liability for any error or omission.

# AB180-20™

## General Purpose and Protocol Engine One Cycle Architecture Processor

**The answer to every Z80, Z180, HD64180 user in the world - yes it has the horsepower you were looking for, no you do not have to re-write your code, you can use your existing Z80 Assemblers, Linkers and C-compilers. Some minor differences do exist.**

### Features:

- \* 40 MHz frequency synthesized 8 bit processor
- \* Memory to Memory Block Transfer at 10 Mbytes/sec
- \* Memory to I/O and I/O to Memory Block Transfer at 10 Mbytes/sec
- \* 8 bit Data bus
- \* 20 bit Address bus
- \* Synchronous serial I/O suitable for Apple Local Talk up to 1Mbaud
- \* 100pin Quad Flat Pack packaging
- \* Two 16bit Timers
- \* Dual 3.3/5V operation or single 3.3V
- \* Each Clock cycle (at 20 MHz) one instruction (for single byte instructions) or the instruction is carried out at the end of the last byte fetch of a multi-byte instruction



# Contents

<b>Introduction .....</b>	<b>7</b>
Chip Structure .....	7
Applications .....	8
Programmer Guide .....	9
Hardware Guide .....	10
<b>Device Details .....</b>	<b>11</b>
Package Information for AB180-20 .....	11
Packaging Information .....	12
I/O Pin Assignment .....	13
Current Consumption v Frequency for the AB180-20 IC .....	17
<b>Overview .....</b>	<b>18</b>
AB180-20 Architecture .....	18
Internal I/O Registers .....	19
Table of Registers .....	20
I/O Addressing Notes .....	22
Dynamic RAM Refresh Control .....	22
Refresh Control and RESET .....	23
Dynamic RAM Refresh Operation Notes .....	23
Logical Address Spaces .....	24
Logical to Physical Address Translation .....	24
<b>Memory Management Unit (MMU) .....</b>	<b>25</b>
Wait State Control .....	25
MMU Block Diagram .....	26
MMU Register .....	26
MMU Register Description .....	28
Physical Address Translation .....	28
MMU and RESET .....	29
MMU Register Access Timing .....	29
<b>Interrupt Control .....</b>	<b>30</b>
Interrupt Control Registers and Flags .....	30
INT/TRAP Control Register (ITC) .....	31
TRAP Interrupt .....	32
External Interrupts .....	34
Internal Interrupts .....	36
Interrupt Acknowledge Cycle Timing .....	38
Interrupt Sources and RESET .....	38
<b>I/O Functions .....</b>	<b>39</b>

<b>Direct Memory Access (DMA)</b>	<b>42</b>
DMAC Register Description	43
DMA Operation	47
DMA Bus Timing	49
DMAC Channel Priority	49
DMAC and BUSREQ*, BUSACK*	49
DMAC Internal Interrupts	50
DMAC and NMI	50
DMAC and RESET	50
<b>Programmable Reload Timer (PRT)</b>	<b>51</b>
PRT Block Diagram	51
PRT Register Description	51
<b>Clocked Serial I/O Port (CSI/O)</b>	<b>54</b>
CSI/O Block Diagram	54
CSI/O Register Description	54
CSI/O Interrupts	55
CSI/O Operation	56
CSI/O Operation Notes	56
CSI/O and RESET	56
CSI/O Operation Timing Notes	57
<b>Bus Timing Information</b>	<b>59</b>
Basic CPU Timing	59
Read Cycle Timing	60
Write Cycle Timing	61
Refresh Cycle Timing	62
Bus Grant Timing	63
Wait State Generation	64
Clock Generation	64
Crystal Oscillator	65
PLL Filter Network	65
External Clock	65

## Appendices

<b>OP-Code Maps</b>	<b>66</b>
---------------------	-----------

## Introduction

This guide provides product information and specifications for the AB180-20 General Purpose and Protocol Engine Processor, detailing the performance capability of this fast, state-of-the-art microprocessor using AB Semicon's unique OCA (One Cycle Architecture) technology.

Applications include:

Network Connected Devices

Digital Cameras

Cell Phones

Automotive

Motor Controllers

Digital Signal Processing (up to 500 kHz)

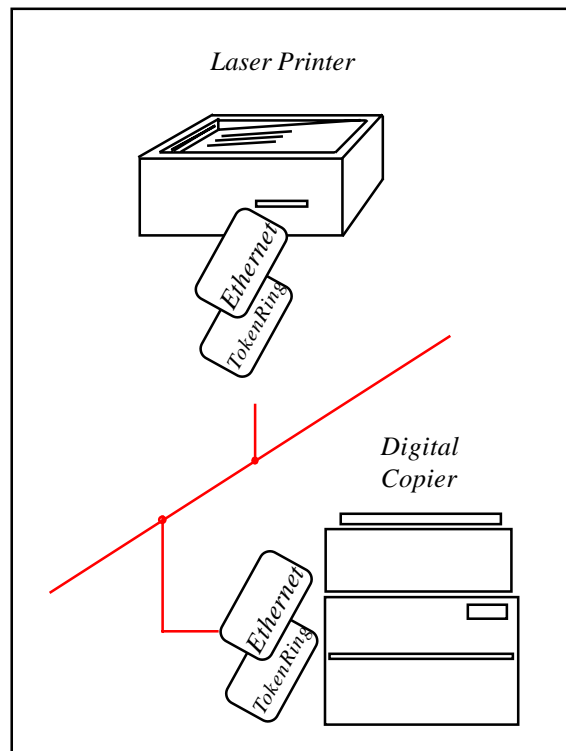
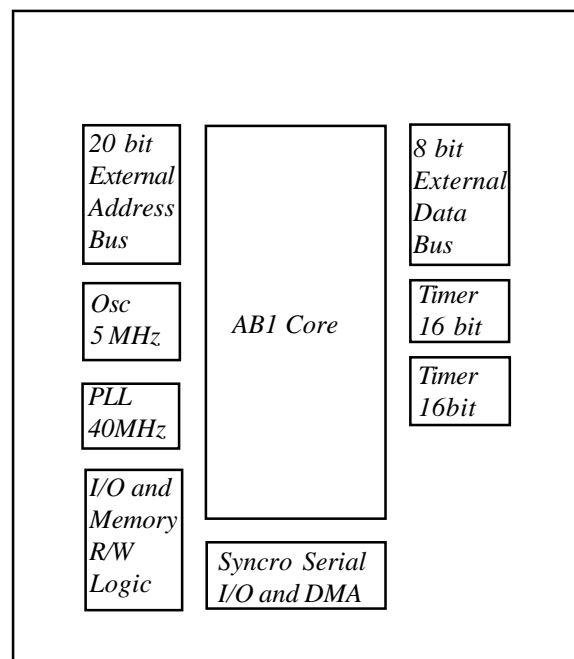


Figure 1

## Chip Structure

### 8 bit processor core

The processor is driven by a 5MHz crystal oscillator and phase locked loop (PLL) which generates the on chip 40MHz clock. The address bus is connected to the outside world giving a 20bit externally addressable RAM or ROM space. Two DMA channels Memory to Memory and Memory to I/O and I/O to Memory are available. The processor wakes up in 4 Wait State Mode allowing lower speed external ROM to be used.



Functional Blocks

Figure 2

## Applications

The chip can be used in many different applications but is most powerful for protocol engine applications such as Network Controller or Fax modem controller for multi function Copiers, Laser Printers, BubbleJet Printers and Fax Machines.

In an Application where the existing Processor is already fully utilised this low cost Protocol Engine Processor can take over all the work of the handling of Network Protocol Stacks and provide the raw data to the printer controller. It can also handle the SNMP data to and from the Printer Controller or NPMP™ management information without taking up any of the Main Printer Controllers time.

### Digital Copier Application / Multi Function Printer

The AB180-20 is a very good match for Digital Copier Designs or Multi Function Printer/Scanner Designs. It allows Network or Modem connectivity to be achieved at extremely low hardware cost and board space. The following diagram shows a typical Digital Copier Design making use of an AB180-20 processor.

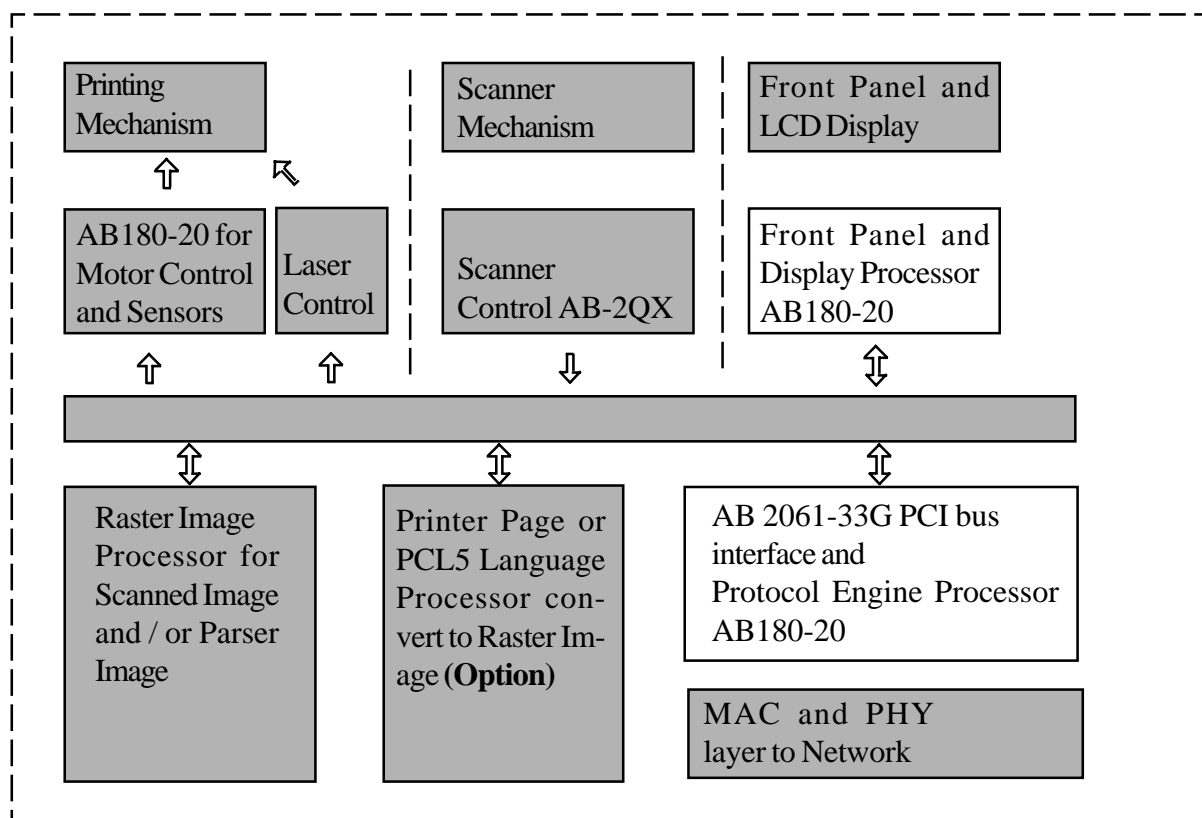


Figure 3



## Programmer Guide

Operationally, the AB180-20 is compatible with the classic Z80 CPU. The programmer has the same register sets, and uses the same op-codes. With a few exceptions (noted later), any program that is designed to run on the Z80 will function on the AB180-20 with little or no modification. In addition to the Z80 instruction set, the AB180-20 has some extra instructions designed to increase efficiency in many common applications.

As well as the CPU core, the AB180-20 also has some on-chip peripherals that are easily accessed by the programmer, and makes the device very powerful in most applications without the need for the plethora of support logic usually required by the Z80. These peripherals are code and functionally compatible with the "180" processors manufactured by Zilog and Hitachi.

The peripherals included on-chip are:

- 1) A versatile wait-state generator that may be programmed to insert different wait-states in up to 3 different areas of memory, in order to allow different speed memories to be used.
- 2) A Memory Management Unit (MMU) that allows the total addressed memory range to be up to 1048576 bytes (20 bit addressing).
- 3) 2 DMA channels, allowing memory-memory and memory-I/O transfers in background.
- 4) 2 clock-timers, which may be programmed to interrupt the CPU at regular intervals and/or generate a hardware signal.
- 5) 1 synchronous serial Communications port, to allow inter-processor communication. Also suitable for downloading many popular FPGA devices.
- 6) A 4 channel interrupt handler.
- 7) A DRAM refresh circuit (same as the Z80). This may be disabled for high-speed operation.

The following is a list of exceptions to the Z80 instruction set that may need to be taken into account when porting code:

- 1) Not all Z80 interrupt modes are supported. The AB180-20 uses interrupt mode 2 only. The interrupt mode instructions are treated as NOPs. Z80 interrupt acknowledge cycles are not generated.
- 2) Due to the fast internal state machine, software timing loops will execute much faster than on a Z80. This may need to be taken into account with some programs.
- 3) The AB180-20 powers up with maximum wait states, and DRAM refresh cycles enabled. To obtain maximum performance some additional instructions should be inserted during the initialisation phase to set the optimum configuration.

- 4) The Z80 has some undocumented instructions, such as being able to load the upper or lower 8 bits of the index register (IY and IX) independently. This was used by a handful of programs, especially some "copy protected" code. The AB180-20 does not support all of the undocumented Z80 instructions.

## Hardware Guide

The AB180-20 interface signals follow the classic Z80 conventions, and interfacing is straightforward using conventional logic, or configurable logic devices. Internally, the CPU state-machine has been greatly enhanced to give very fast code execution times. Most instructions have no internal "T" states, and the AB180-20 (without wait-states) will execute based on 1 memory access per internal clock cycle. The internal clock runs at 4 times the crystal speed. This internal clock is available on an output pin.

Due to this difference, and the fact that conventional Z80 peripherals cannot be used (they are too slow), the Z80 hardware state signals E, ST, LIR are not provided. This should not present any down-sides, and the AB180-20 may be used in most applications as a replacement for the 180 processors. Due to the internal wait-state generator, fast memory is not a necessity, although faster execution times are naturally available if wait-states can be avoided. With 1 wait-state, there is at least a three-fold increase in speed over the fastest 180 processor currently available, and a five-fold increase is usual with faster memory and no wait-states. Some applications will see an even more dramatic improvement, because the "block" instructions (e.g. LDIR) are greatly enhanced.

The external -WAIT- input has different timing to that of the Z80 or the 180. Generally this makes interfacing easier, as -WAIT- may be asserted together with -IOE- or -ME-, without needing to gate -RD- or -WR- signals. It is a consideration that may trip the unwary however.

The core logic of the AB180-20 operates at 3.3V, and so this must be provided. The I/O pins are capable of either 3.3V or true 5V operation, depending upon what voltage is applied to the I/O power supply pins. Thus the AB180-20 is suitable for use in either 3.3V or 5V circuits. For 5V use, a simple zener-diode voltage dropping circuit is ample to supply the low-current core.

The I/O bus operates at 2 times the core frequency (see timing diagrams) which is 8 times the crystal frequency.

## Device Details

### Package Information for AB180-20

The following diagram shows the package information for the AB180-20 Protocol Engine Processor:

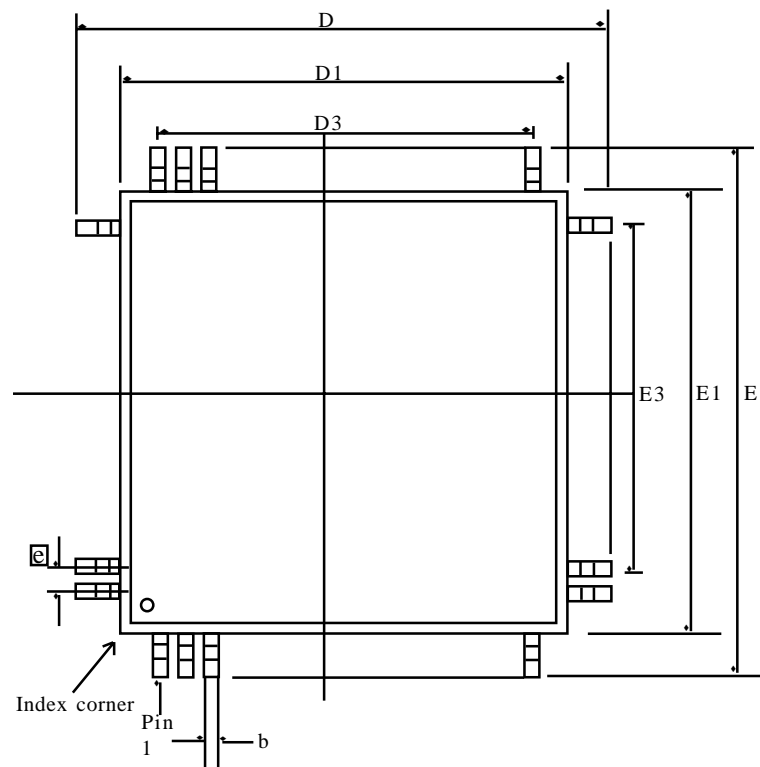


Figure 4

## Packaging Information

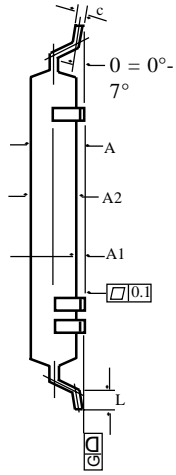


Figure 5

Symbol	Control Dimensions in millimetres			Alternative Dimensions in inches		
	MIN	Nominal	MAX	MIN	Nominal	MAX
A	2.80		3.40	0.110		0.134
A1	0.25		0.85	0.010		0.033
A2	2.55		3.05	0.100		0.120
D	23.65		24.15	0.931		0.951
D1	19.80		20.20	0.780		0.795
D3	18.85 REF.			0.742 REF.		
E	17.65		18.15	0.695		0.715
E1	13.80		14.20	0.543		0.559
E3	12.35 REF.			0.486 REF.		
L	0.73		1.03	0.029		0.041
e	0.65 BSC.			0.026 BSC.		
b	0.22		0.38	0.009		0.015
c	0.11		0.23	0.004		0.009
Pin features						
N	100					
ND	30					
NE	20					
NOTE	RECTANGULAR					

Conforms to JEDEC MO-112 CC-1 Iss. B.

◆ Note: This package is rectangular

## I/O Pin Assignment

<i>Pin ID</i>	<i>Signal</i>	<i>Designator</i>
1	-	MixVss3 0V
2	-	N/C
3	-	N/C
4	-	MixVss 0V
5	-	MixVss1 0V
6	input	NMI
7	Output	A<0>
8	Output	A<1>
9	Output	A<2>
10	Output	A<3>
11	Output	A<4>
12	Output	A<5>
13	Output	A<6>
14	Output	A<7>
15	-	MixVdd 3.3/5V
16	Output	A<8>
17	Output	A<9>
18	-	N/C
19	Output	A<10>
20	Output	A<11>
21	Output	A<12>
22	Output	A<13>
23	Output	A<14>
24	Output	A<15>
25	Output	A<16>

## I/O Pin Assignment

<i>Pin ID</i>	<i>Signal</i>	<i>Designator</i>
26	-	<i>MixVdd3 3.3/5V</i>
27	-	<i>MixVdd4 3.3/5V</i>
28	-	<i>N/C</i>
29	-	<i>N/C</i>
30	-	<i>MixVdd1 3.3/5V</i>
31	-	<i>MixVdd2 3.3/5V</i>
32	<i>Output</i>	<i>A&lt;19&gt; /TOUT</i>
33	<i>Input/Output</i>	<i>D&lt;0&gt;</i>
34	<i>Input/Output</i>	<i>D&lt;1&gt;</i>
35	<i>Input/Output</i>	<i>D&lt;2&gt;</i>
36	<i>Input/Output</i>	<i>D&lt;3&gt;</i>
37	<i>Input/Output</i>	<i>D&lt;4&gt;</i>
38	-	<i>Vssi 0V</i>
39	<i>Input/Output</i>	<i>D&lt;5&gt;</i>
40	<i>Input/Output</i>	<i>D&lt;6&gt;</i>
41	<i>Input/Output</i>	<i>D&lt;7&gt;</i>
42	-	<i>Vssi 0V</i>
43	<i>Output</i>	<i>OUT0</i>
44	<i>Input</i>	<i>IN1</i>
45	<i>Input</i>	<i>IN0</i>
46	-	<i>N/C</i>
47	<i>Output</i>	<i>A&lt;17&gt;</i>
48	<i>Output</i>	<i>A&lt;18&gt;</i>
49	-	<i>MixVdd3 3.3/5V</i>
50	-	<i>MixVdd4 3.3/5V</i>

## I/O Pin Assignment

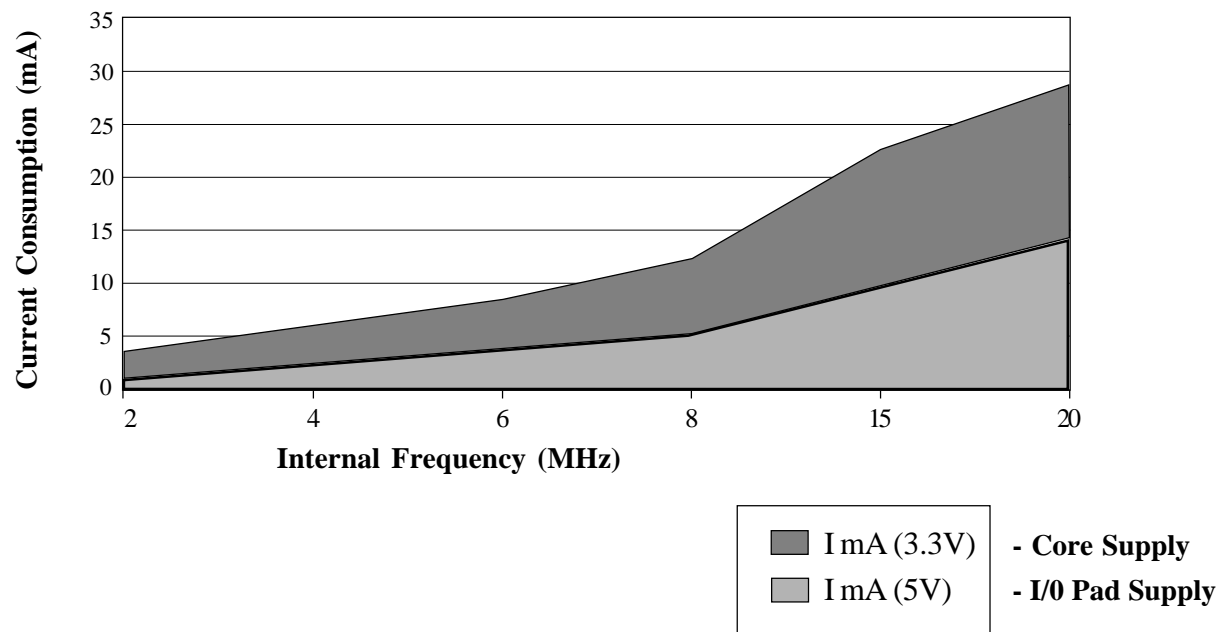
<i>Pin ID</i>	<i>Signal</i>	<i>Designator</i>
51	-	N/C
52	-	N/C
53	-	N/C
54	-	MixVdd1 3.3/5V
55	-	MixVdd1 3.3/5V
56	-	Vssi 0v
57	Input/Output	FRQ0/DREQ0
58	-	N/C
59	---	N/C
60	-	Vssi 0v
61	Output	FRQ1/TEND0
62	Output	TXS
63	Input	RXS/IN2
64	-	Vssi 0V
65	-	Vdd 3.3/5V
66	Output	CKS
67	Input	DREQ1
68	Output	TEND1
69	Output	HALT
70	Output	REF
71	Output	IOE
72	Output	WR
73	Output	RD
74	Output	ME
75	-	N/C

## I/O Pin Assignment

<i>Pin ID</i>	<i>Signal</i>	<i>Designator</i>
76	-	<i>MixVss2 0V</i>
77	-	<i>MixVss3 0V</i>
78	-	<i>N/C</i>
79	-	<i>N/C</i>
80	-	<i>MixVss 0V</i>
81	-	<i>MixVss1 0V</i>
82	<i>Output</i>	<i>PHY</i>
83	<i>Input</i>	<i>INT0</i>
84	<i>Input</i>	<i>INT1</i>
85	<i>Input</i>	<i>INT2</i>
86	-	<i>Vdd 3.3/5V</i>
87	-	<i>Vddi3V 3.3V</i>
88	-	<i>AllVddShort 3.3V</i>
89	-	<i>N/C</i>
90	-	<i>N/C</i>
91	-	<i>N/C</i>
92	<i>Input</i>	<i>LFT (3V sig)</i>
93	<i>Input</i>	<i>EXTAL (3V sig)</i>
94	<i>Output</i>	<i>XTAL (3V sig)</i>
95	-	<i>AllVssShorted 0V</i>
96	<i>Input</i>	<i>WAIT</i>
97	<i>Output</i>	<i>BUSACK</i>
98	<i>Input</i>	<i>RESET</i>
99	<i>Input</i>	<i>BUSREQ</i>
100	-	<i>MixVss2 0V</i>



## Current Consumption v Frequency for the AB180-20 IC



Frequency	I mA (5V)	I mA (3V3)	V (5V)	V (3V3)
2	1.39	2.08	5	3.28
4	2.6	3.4	5	3.28
6	3.43	4.44	5	3.28
8	5.4	6.65	4.98	3.28
15	9.99	12.35	5.01	3.28
20	13.26	15.72	4.93	3.196

# Overview

## AB180-20 Architecture

The AB180-20 CPU has five functional blocks:

- Central Processing Unit:** The AB180-20 uses a superset of the Z80 CPU instruction set, with interface signals that follow Z80 conventions, using conventional logic or configurable logic devices. The CPU has been engineered to give very fast code execution times. In addition to the Z80 instruction set, the AB180-20 includes extra instructions to increase efficiency in many common applications.
- Clock Generator:** The clock generator consists of a 5 MHz crystal oscillator and Phased locked loop (PLL) which generates the on-chip 40MHz and 20MHz clocks.
- Bus State Controller:** This performs all of the bus control and status activity associated with the CPU and on-chip peripherals, including DMA bus exchanges, reset cycles, DRAM refresh, and wait state timing.
- Memory Management Unit:** The MMU allows the user to increase the available memory of the CPU from 64k (for the Z80) to 1M Byte. This is achieved with a *common/banked area* structure.
- Interrupt Controller:** In order to provide the correct responses from the CPU, the interrupt controller monitors and prioritizes the various external and internal interrupts and traps.

The AB180-20 CPU has four on-chip peripherals:

- DMA Controller:** The DMA controller provides high speed transfers from memory to memory, memory to/from I/O, and I/O to I/O. It supports the modes *request* and *cycle steal*. DMA transfers can access the complete 1Mbyte addressing range.
- Programmable Reload Timer (PRT - two channels):**  
The PRT consists of two separate channels containing a 16 bit timer and count reload register. Before reaching the counters, the system clock provides the time base required. Channel 1 has an optional output that enables waveform generation.
- Clock Serial I/O (CSIO):** The CSIO channel enables synchronous high-speed data communication with other microprocessors, microcomputers and peripherals using a half-duplex serial transmitter and receiver.

## Internal I/O Registers

The internal I/O registers of the AB180-20 occupy the 64 addresses between 00h and 3Fh on reset. To avoid conflicts with external devices these registers can be relocated within the bottom 256 bytes of the I/O address space.

### I/O Control Registers

IOA7	IOA6	Ø	-	-	-	-	-
------	------	---	---	---	---	---	---

### IOA [7:6] I/O Address Relocation

These bits relocate the internal I/O registers as shown below in Table 1:

IOA [7:6] = 11	{		00FFh
			0000h
IOA [7:6] = 10	{		00BFh
			0080h
IOA [7:6] = 01	{		007Fh
			0040h
IOA [7:6] = 00	{		003Fh
			00BFh

Table 1

## Table of Registers

	Register	Mnemonic	Address	
			Binary	Hexadecimal
I/O	I/O Control Register	IOREGA	XX000000	00H
	I/O Control Register	IOREGB	XX000001	01H
	I/O Control Register	IOREGC	XX000010	02H
	I/O Control Register	IOREGD	XX000011	03H
	I/O Control Register	IOREG E	XX000100	04H
	I/O Control Register	IOREGF	XX000101	05H
	Reserved			
	Reserved			
	Reserved			
	Reserved			
CSI/O	CSI/O Control Register	CNTR	XX001010	0AH
	CSI/O Transmit/Receive Data Register	TRDR	XX001011	0BH
Timer	Timer Data Register Ch OL	TMDROL	XX001100	0CH
	Timer Data Register Ch OH	TMDROH	XX001101	0DH
	Reload Register Ch OL	RLDROL	XX001110	0EH
	Reload Register Ch OH	RLDROH	XX001111	0FH
	Timer Control Register	TCR	XX010000	10H
	Reserved		XX010001	11H
			}	}
			XX010011	13H
	Timer Data Register Ch 1L	TMDR1L	XX010100	14H
	Timer Data Register Ch 1H	TMDR1H	XX010101	15H
	Reload Register Ch 1L	RLDR1L	XX010110	16H
	Reload Register Ch 1H	RLDR1H	XX010111	17H
	Reserved		XX011000	18H
			}	}
			XX011111	1FH
				Unused

	Register	Mnemonic	Address	
			Binary	Hexadecimal
DMA	DMA Source Address Register Ch OL	SAROL	XX100000	20H
	DMA Source Address Register Ch OH	SAROH	XX100001	21H
	DMA Source Address Register Ch OB	SAROB	XX100010	22H
	DMA Destination Address Register Ch OL	DAROL	XX100011	23H
	DMA Destination Address Register Ch OH	DAROH	XX100100	24H
	DMA Destination Address Register Ch OH	DAROB	XX100101	25H
	DMA Byte Count Register Ch OL	BCROL	XX100110	26H
	DMA Byte Register Ch OH	BCROH	XX100111	27H
	DMA Memory Address Register Ch 1L	MAR1L	XX101000	28H
	DMA Memory Address Register Ch 1H	MAR1H	XX101001	29H
	DMA Memory Address Register Ch 1B	MAR1B	XX101010	2AH
	DMA I/O Address Register Ch 1L	IAR1L	XX101011	2BH
	DMA I/O Address Register Ch 1H	IAR1H	XX101100	2CH
	Internal Wait Control Register	IMWR	XX101101	2DH
	DMA Byte Count Register Ch 1H	BCR1H	XX101111	2FH
	DMA Status Register	DSTAT	XX110000	30H
	DMA Mode Register	DMODE	XX110001	31H
	DMA/ WAIT Control Register	DCNTL	XX110010	32H
INT	IL Register (INT VECtor Register)	IL	XX110011	33H
	INT/ TRAP Control Register	ITC	XX110100	34H
	Reserved		XX110101	35H
Refresh	Refresh Control Register	RCR	XX110110	36H
	Reserved		XX110111	37H
MMU   I/O	MMU Common Base Register	CBR	XX111000	38H
	MMU Bank Base Register	BBR	XX111001	39H
	MMU Common/Bank Area Register	CBAR	XX111010	3AH
	Reserved		XX111011	3BH
			XX111110	3EH
	I/O Control Register	ICR	XX111111	3FH

## I/O Addressing Notes

The on-chip register addresses are located in the I/O address space from 0000H to 00FFH (16-bit I/O addresses). In order to access the on-chip I/O registers (using I/O instruction), the high-order 8 bits of the 16-bit I/O address must be 0.

The conventional I/O instruction (OUT (m),A/ IN A, (m) / OUTI /INI etc.) place the contents of a CPU register on the high-order 8 bits of the address bus, and may be difficult to use for accessing on chip I/O registers. For more efficient on-chip I/O register access, the AB180-20 has additional instructions above the Z180 which force the high-order 8 bits of the I/O address to 0. These instructions are IN0, OUT0, OTIMR, OTDMR and TSTIO (See Appendicies at the back of this book).

When writing to an internal I/O register, the same I/O write occurs on the external bus. However, the duplicate external I/O write cycle will exhibit internal I/O write cycle timing. For example, the WAIT\* input and programmable wait state generator are ignored. This will be the same for internal I/O read cycles - however, the external read data is ignored by the AB180-20.

It is advised that external I/O addresses should be chosen to avoid overlap with internal I/O addresses. This avoids duplicate I/O accesses.

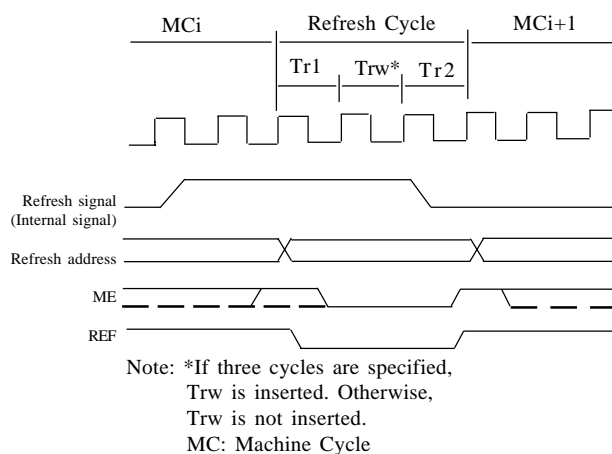
## Dynamic RAM Refresh Control

The AB180-20 incorporates a dynamic RAM refresh control circuit including 8 bit refresh address generation and programmable refresh timing. This circuit generates asynchronous refresh cycles inserted at the programmable interval independent of CPU program execution. For systems which don't use dynamic RAM, the refresh function can be disabled.

When the internal refresh controller determines that a refresh cycle should occur, the current cycle is inserted by placing the refresh address on A0-A7 and the REF\* output is driven LOW.

By programming the REFW (Refresh Wait) bit in RCR (Refresh Control Register), the refresh cycles can be set for a clock cycle of either two or three. The external WAIT\* input and the internal wait state generator are not effective during refresh.

Fig.6 shows the timing of a refresh cycle with a refresh wait (Trw) cycle.



**Figure 6 Refresh Timing**

### Refresh Control Register (RCR)

RCR sets the interval and length of refresh cycles and enables or disables the refresh function.

bit 7	6	5	4	3	2	1	0
REFE	REFW	-	-	-	-	CYC1	CYC0
R/W		R/W				R/W	R/W

#### REFE: Refresh Enable (bit 7)

REFE = 0 disables the refresh controller while REFE = 1 enables refresh cycle insertion. REFE is set = 1 during RESET.

#### REFW: Refresh Wait (bit 6)

REFW = 0 causes the refresh cycle to be two clocks in duration. REFW = 1 causes the refresh cycle to be three clocks in duration by adding a refresh wait cycle (Trw). REFW is set = 1 during RESET.

In the case of dynamic RAMs requiring 128 refresh cycles every 2 ms (or 256 cycles every 4ms), the required refresh interval is less than or equal to 15.625 ms.

### Refresh Control and RESET

After RESET, based on the initialised value of RCR, refresh cycles will occur with an interval of 248 clock cycles and be 3 clock cycles in duration.

### Dynamic RAM Refresh Operation Notes

Refresh cycle insertion is stopped when the CPU is in following states.

- (a) During RESET
- (b) When the bus is released in response to BUSREQ\*
- (c) During WAIT states

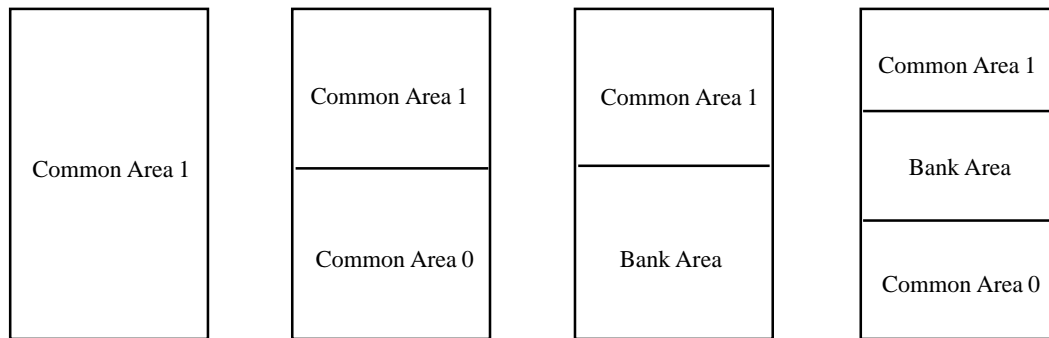
Refresh cycles are suppressed when the bus is released in response to BUSREQ\*, but the refresh timer continues to operate, so the time at which the first refresh cycle occurs after the AB180-20 re-acquires the bus depends on the refresh timer. There is no timing relationship with the bus exchange.

**Note:** each refresh bus cycle will use a refresh address that has increased by 1 from the previous refresh bus cycle (This is for each completed refresh cycle, not each refresh request).

## Logical Address Spaces

The 64k bytes CPU logical address space is interpreted by the MMU as consisting of up to three separate logical address areas, Common Area 0, Bank Area and Common Area 1.

Fig 7 displays the logical memory configurations that are possible. The boundaries between the Common and Bank areas can be programmed with 4k bytes resolution.

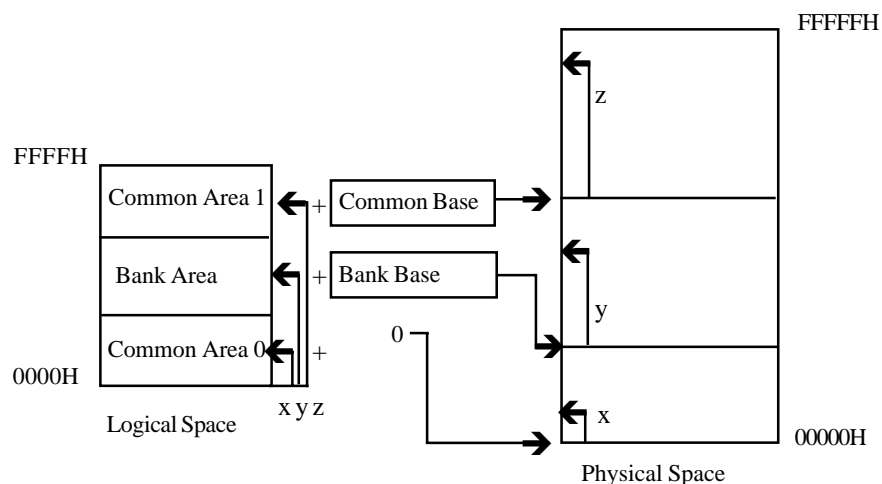


**Figure 7**

**Logical Address Mapping Examples**

## Logical to Physical Address Translation

Fig 8 is an example in which the three logical address space portions are mapped into a 1M bytes physical address space. It can be seen that Common and Bank areas can overlap and that Common Area 1 and Bank Area can be relocated (on 4k bytes physical address boundaries). Common Area 0 (if it exists) is always based at physical address 0.



**Figure 8**

**Logical → Physical Mapping Example**



## Memory Management Unit (MMU)

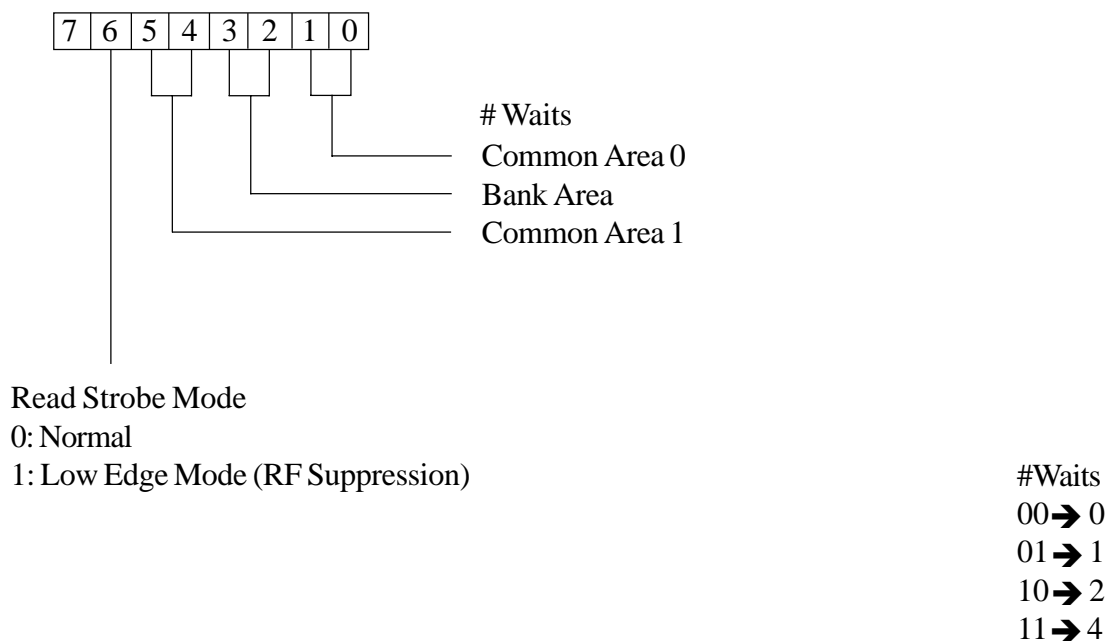
The AB180-20 contains an on-chip MMU which performs the translation of the CPU 64k bytes (16-bit addresses- 0000H to FFFFH) logical memory address space into a 1M bytes (20-bit addresses- 00000H to FFFFFH) physical memory address space. Address translation is performed internally in parallel with other CPU operation.

## Wait State Control

Each logical memory address space can have an independantly assigned number of wait states. This is controlled through the IMWR register (2dh).

The internal memory map and register function is similar to the Z180, with the following additions.

Bus Controller - IMWR (External Wait Control) Register Address = 2d hex.

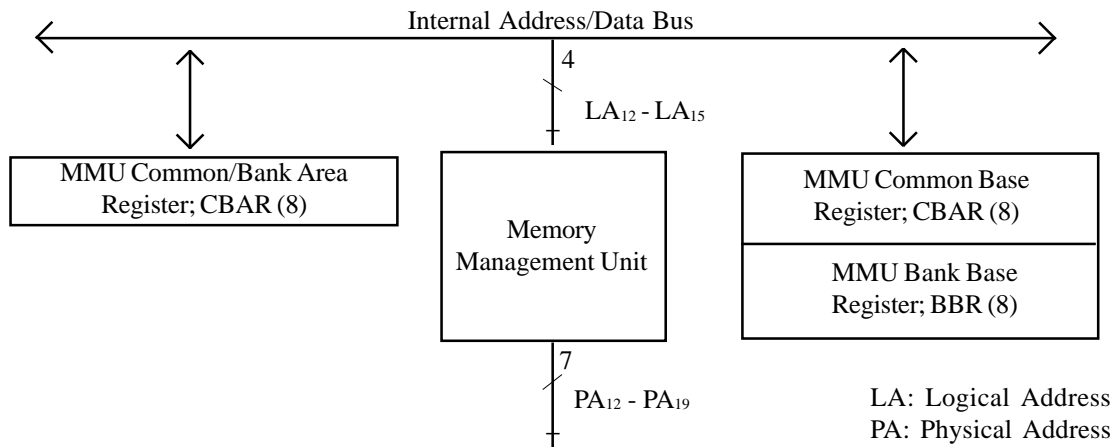


NB: When using the IMWR to set different numbers of wait states for each bank it is required that the following restriction is observed:

$$CBR \geq BBR$$

## MMU Block Diagram

Fig 20 shows the MMU block diagram. The MMU translates internal 16-bit logical addresses to external 20-bit physical addresses.



**Figure 20**

**MMU Block Diagram**

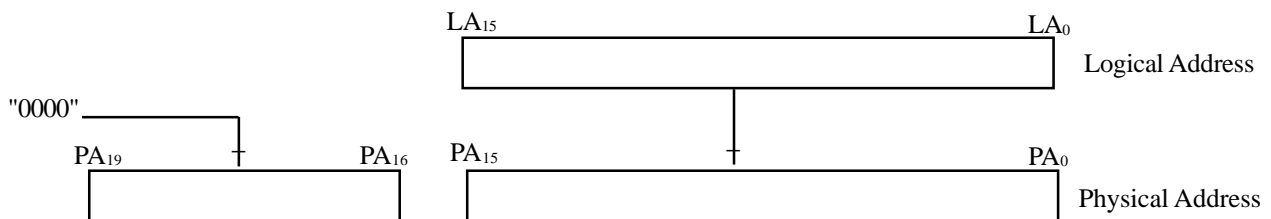
Whether address translation takes place depends on the type of CPU cycle as follows:

### Memory Cycles

Address Translation occurs for all memory access cycles including instruction and operational fetchs, memory data reads and write, hardware interrupt vector fetch and software interrupt restarts.

### I/O Cycles

The MMU is logically bypassed for I/O cycles. The 16-bit logical I/O address space corresponds directly with the 16-bit physical I/O address space. The four high order bits (A16-A19) of the physical address are always 0 during I/O cycles.



**Figure 21**

**I/O Address Translation**

### DMA Cycles

When the AB180-20 on-chip DMAC is using the external bus, the MMU is physically bypassed. The 20-bit source and destination registers in the DMAC are directly output on the physical address bus (A0-A19).

## MMU Register

Three MMU registers are used to program a specific configuration of logical and physical memory.

- (1) MMU Common/Bank Area Register (CBAR)
- (2) MMY Common Base Register (CBR)
- (3) MMU Bank Base Register (BBR)

CBAR is used to define the logical memory organisation. CBR and BBR are used to relocate logical areas within the 512k bytes physical address space. The resolution for setting boundaries within space and relocation within the physical space is 4k bytes.

The CAR fiels of CBAR determines the start address of Common Area 1 (Up-per Common) and by default, the end address of the Bank Area. The BAR field determines the start address of the Bank Area and by default, the end address of Common Area 0 (Lower Common).

The CA and BA field of CBAR may be programmed subject to the restriction that CA may never be less than BA Fig 22 and Fig 23 show examples of logical memory organisations associated with different values of CA and BA.

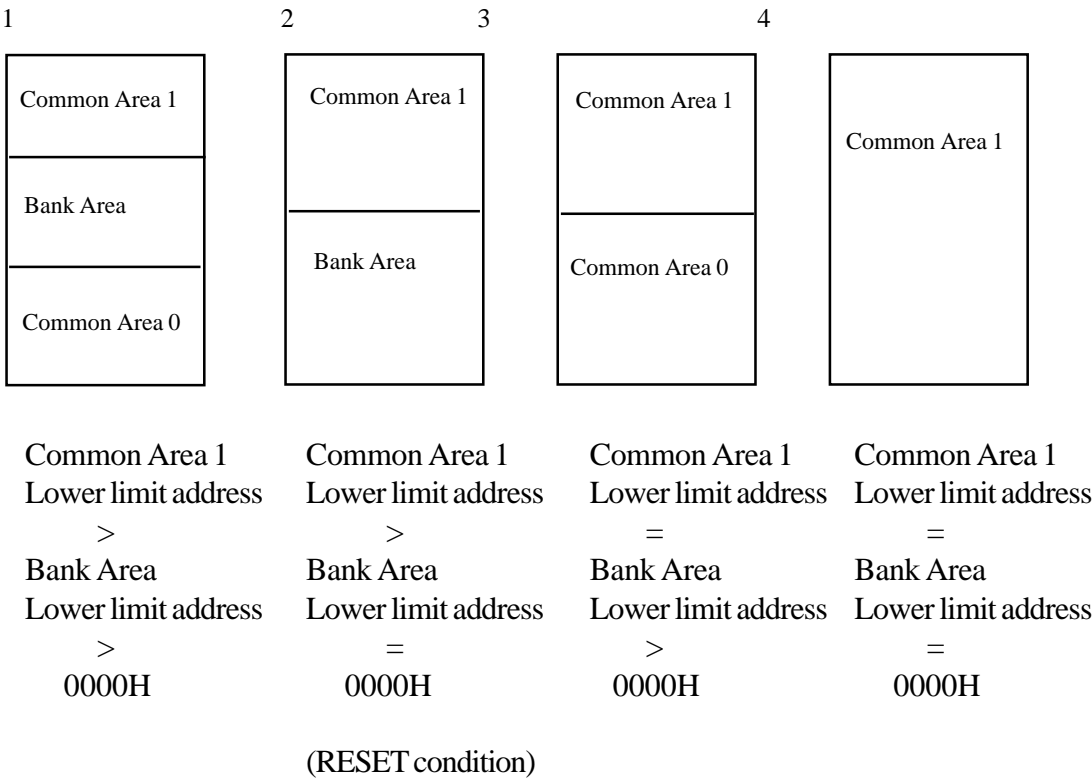


Figure 22
Logical Memory Organisation

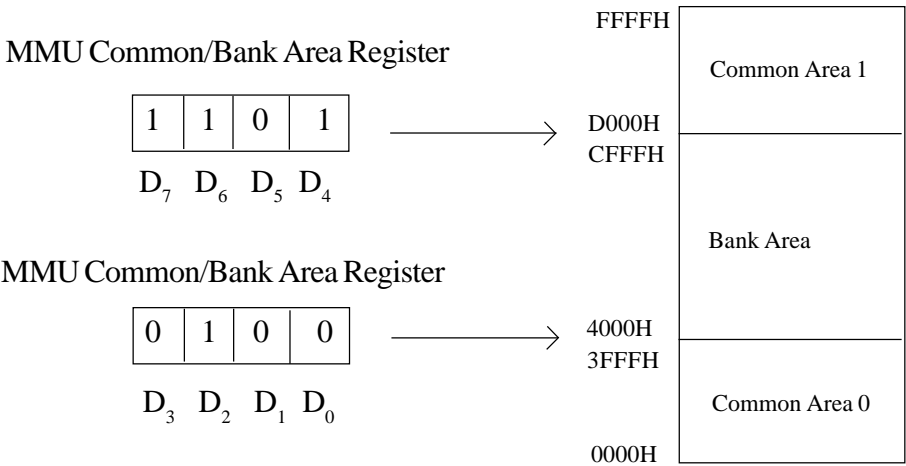


Figure 23
Logical Space Configuration (Example)

## MMU Register Description

### MMU Common/Bank Area Register (CBAR)

CBAR specifies boundaries within the AB180-20 64k bytes logical address space for up to three areas, Common Area 0, Bank Area and Common Area 1.

MMU Common/Bank Area Register (CBAR : I/O Address = 3AH)								
bit	7	6	5	4	3	2	1	0
	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

#### CA3-CA0: CA (bits 7-4)

CA specifies the start (low) address (on 4k bytes boundaries) for the Common Area 1. This also determines the last address of the Bank Area. All bits of CA are set to 1 during RESET.

#### BA3-BA0: BA (bits 3-0)

BA specifies the start (low) address (on 4k bytes boundaries) for the Bank Area. This also determines the last address of the Common Area 0. All bits of BA are reset to 0 during RESET.

### MMU Common Base Register (CBR)

CBR specifies the base address (on 4k boundaries) used to generate 19-bit physical address for Common Area 1 accesses. All bits of CBR are reset to 0 during RESET.

MMU Common/Bank Area Register (CBAR : I/O Address = 38H)								
bit	7	6	5	4	3	2	1	0
	CB7	CB6	CB5	CB4	CB3	CB2	CB1	CB0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

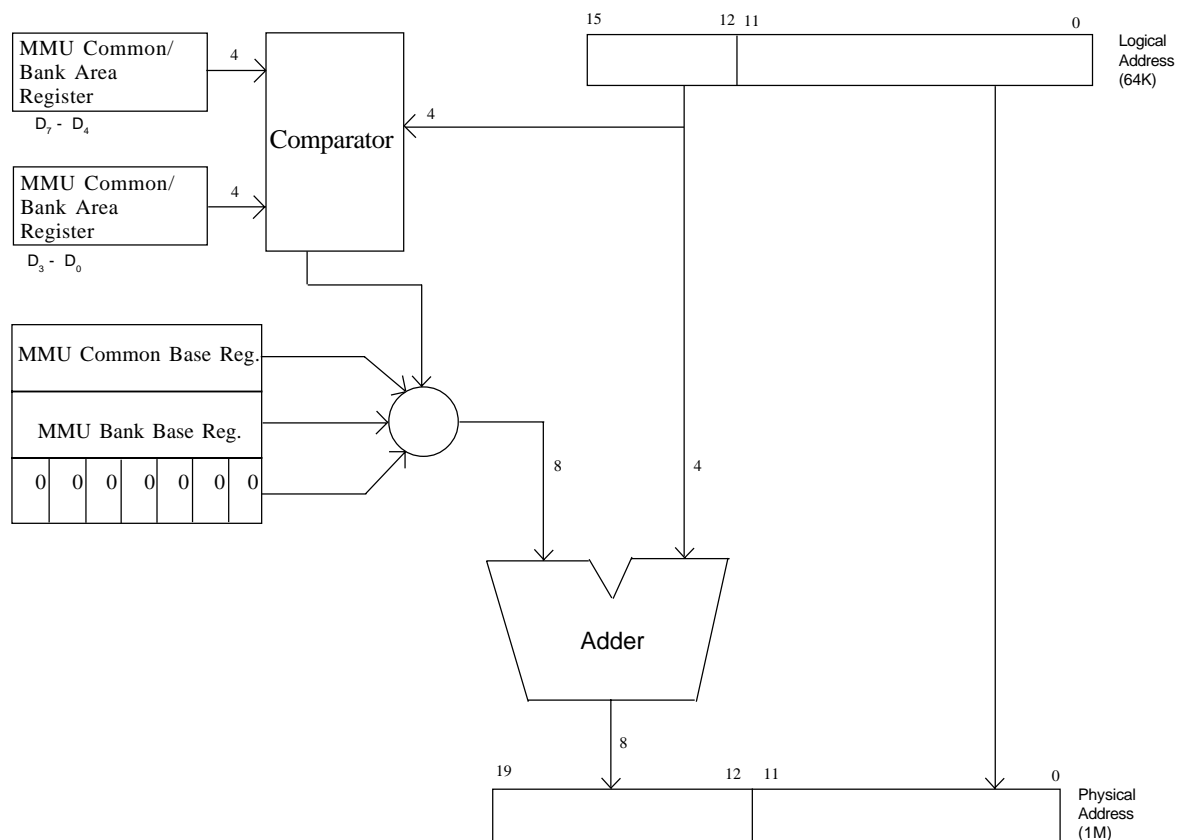
### MMU Bank Base Register (BBR)

BBR specifies the base address (on 4k boundaries) used to generate 19-bit physical address for Common Area 1 accesses. All bits of BBR are reset to 0 during RESET.

MMU Bank Base Register (BBR : I/O Address = 39H)								
bit	7	6	5	4	3	2	1	0
	BB7	BB6	BB5	BB4	BB3	BB2	BB1	BB0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## Physical Address Translation

Fig 24 shows the way in which physical addresses are generated based on the contents of CBAR, CBR and BBR. MMU comparators classify an access by logical area as defined by CBAR. Depending on which of the three potential logical areas (Common Area 1, Bank Area or Common Area 0) is being accessed, the appropriate 7-bit base address is added to the high-order 4 bits of the logical address, yielding a 20-bit physical address. CBR is associated with Common Area 1 accesses. Common Area 0 accesses use an internal base register (non-accessible) which contains 0. Thus, Common Area 0, if defined, is always based at physical address 0.



**Figure 24**

**Physical Address Generation**

## MMU and Reset

During REST, all bits of the CA field of CBAR are set to 1. All nits of the BA field of CBAR, CBR and BBR are reset to 0. The logical 64k bytes address space corresponds directly with the first 64k bytes (0000H to FFFFH) of the 1M bytes (00000H to FFFFFH) physical address space, so after RESET the AB180-20 will begin execution at logical and physical address 0.

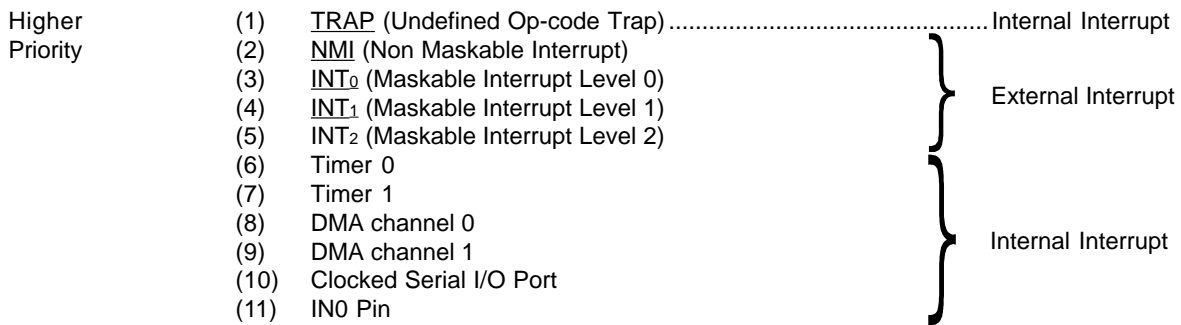
## MMU Register Access Timing

When data is written into CBAR, CBR or BBR, the value will be effective from the cycle immediately following the I/O write cycle which updates these registers.

**Note:** care must be taken during MMU programming to insure that CPU program execution is not disrupted. The next cycle following MMU register programming will normally be an op-code fetch from the newly translated address. One way of ensuring this is to localise all MMU programming routines in a Common Area that is always enabled.

## Interrupt Control

The AB180-20 CPU has eleven interrupt sources, four external and eight internal, with fixed priority.



**Figure 9**

**Interrupt Sources**

This section explains the CPU registers associated with interrupt processing, the TRAP interrupt, interrupt response modes and the external interrupts. For full information on internal interrupt generation other than TRAP see the sections I/O, DMA, PRT and CSI/O as required.

### Interrupt Control Registers and Flags

The AB180-20 contains three registers and two flags which are associated with interrupt processing.

	<u>Function</u>	<u>Name</u>	<u>Access Method</u>
(1)	Interrupt Vector High	I	LD A, I and LD I, A instructions
(2)	Interrupt Vector Low	IL	I/O instruction (addr=33H)
(3)	Interrupt/Trap Control	ITC	I/O instruction (addr=34H)
(4)	Interrupt Enable Flag 1,2	IEF1,2	EI and DI LD A, I LD A, R instructions

#### Interrupt Vector Register (I)

External interrupts INT1, INT2 and all internal interrupts (except TRAP) use a programmable vectored technique (similar to mode 2) to determine the address at which interrupt processing starts. In response to the interrupt a 16-bit address is generated. This address accesses a vector table in memory to obtain the address at which execution restarts.

INT 0 generates a jump to an ISR at 0038h

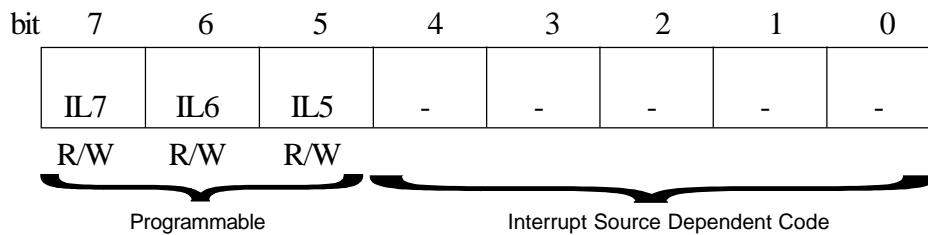
The methods for generation of the least significant byte of the table address are different, but all vectors use the contents of I as the most significant byte of the table address. By programming the contents of I, vector tables can be relocated on 256 bytes boundaries throughout the 64k logical address space.

**Note:** that I is read/written with the LD A, I and LD I, A instructions rather than I/O (IN, OUT) instructions.

I is initialized to 0 during RESET.

## Interrupt Vector Low Register (IL)

Interrupt Vector Low Register (IL: I/O Address = 33H)

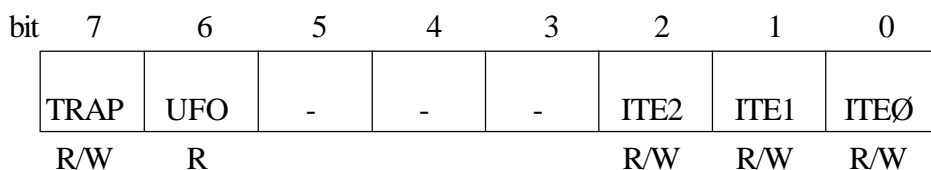


This register determines the most significant three bits of the low order byte of the interrupt vector table address for external interrupt INT1 and INT2 and all internal interrupts (except TRAP). The five least significant bits are fixed for each specific interrupt source. By programming IL the vector table can be relocated on 32 bytes boundaries.

IL is initialised to 0 during RESET.

## INT/TRAP Control Register (ITC)

INT/TRAP Control Register (ITC : I/O Address = 34H)



ITC is used to handle TRAP interrupts and to enable or disable the external maskable interrupt inputs INT0\*, INT1\* and INT2\*.

### TRAP (bit 7)

This bit is set to 1 when an undefined up-code is fetched. TRAP can be reset under program control by writing with 0, however it cannot be written with 1 under program control. TRAP is reset to 0 during RESET.

### UFO: Undefined Fetch Object (bit 6)

When a TRAP interrupt occurs (TRAP bit set to 1), the contents of UFO allow determination of the starting address of the undefined instruction. This is necessary since the TRAP may occur on either the third byte of the op-code. UFO allows the stacked PC value (stacked in response to TRAP) to be correctly adjusted. If UFO = 0, the first op-code should be interpreted as the stacked PC - 1. If UFO = 1, the first op-code address is stacked PC-2. UFO is read-only.

### ITE2, 1,0: Interrupt Enable 2, 1,0 (bits 2-0)

ITE2, ITE1, and ITE0 enable and disable the external interrupt inputs INT2\*, INT1\* and INT0\* respectively. If reset to 0, the interrupt is masked. During RESET, ITE0 is initialised to 1 while ITE1 and ITE2 are initialised to 0.

### Interrupt Enable Flag 1,2 (IEF1,2)

IEF1 controls the overall enabling and disabling of all internal and external maskable interrupts (i.e. all interrupts except NMI and TRAP).

If IEF1 = 0, all maskable interrupts are disabled. IEF1 can be reset to 0 by the DI (Disable Interrupts) instruction and set to 1 by the EI (Enable Interrupts) instruction.

The purpose of IEF2 is to correctly manage the occurrence of NMI. During NMI the prior interrupt reception state is saved and all maskable interrupts are automatically disabled (IEF1 copied to IEF2 and then IEF1 cleared to 0). At the end of the NMI interrupt service routine, execution of the RETN (Return from Non-maskable Interrupt) will automatically restore the interrupt receiving state (by copying IEF2 to IEF1) prior to the occurrence of NMI.

IEF2 state can be reflected in the P/V bit of the CPU Status register by execution of the (a) LD A, I or (b) LD A, R instructions.

Table 10 shows the state of IEF1 and IEF2.

CPU Operation	IEF1	IEF2	REMARKS
RESET	0	0	Inhibits the interrupt except NMI and TRAP
NMI	0	IEF1	Copies the contents of IEF1 to IEF2
RETN	IEF2	not affected	Returns from the NMI service routine
Interrupt except NMI and TRAP	0	0	Inhibits the interrupt
RET1	not affected	not affected	
TRAP	not affected	not affected	
EI	1	1	
DI	0	0	
LD A, I	not affected	not affected	Transfers the contents of IEF2 to P/V flag
LD A, R	not affected	not affected	Transfers the contents of IEF2 to P/V flag

**Table 10**

**State of IEF1 and IEF2**

### TRAP Interrupt

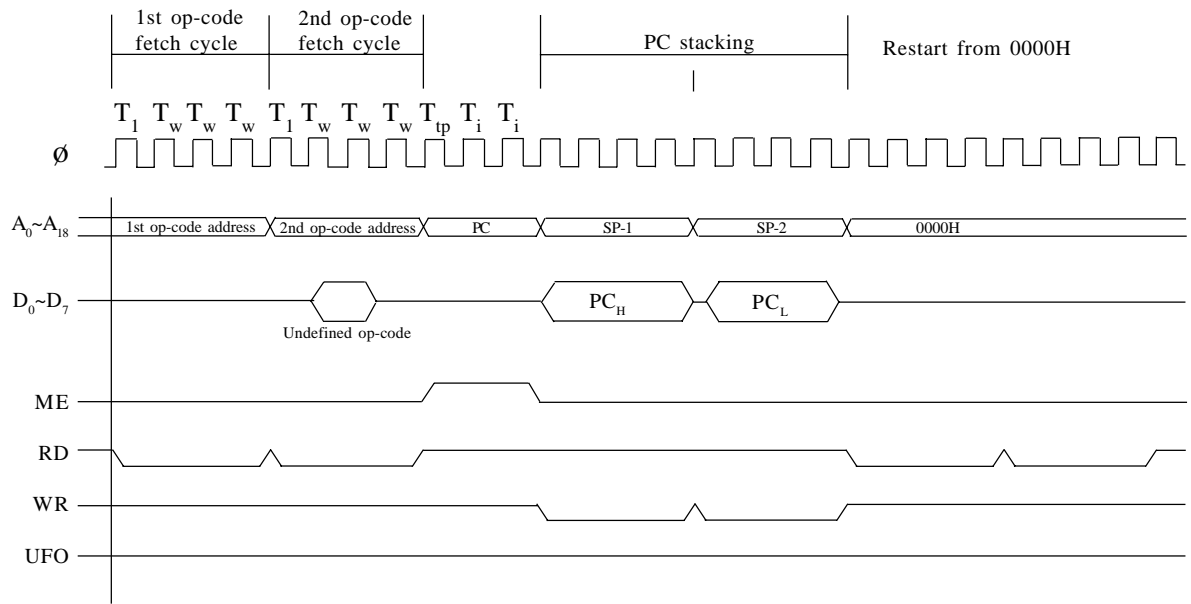
The AB180-20 creates a non-maskable TRAP interrupt when an undefined op-code fetch occurs (This is not affected by the of IEF1). This feature can be used to increase software reliability, implement an 'extended' instruction set, or both. TRAP may occur during op-code fetch cycles.

When a TRAP interrupt occurs the AB180-20 operates as follows;

- (1) The TRAP bit in the Interrupt TRAP/Control (ITC) register is set to 1.

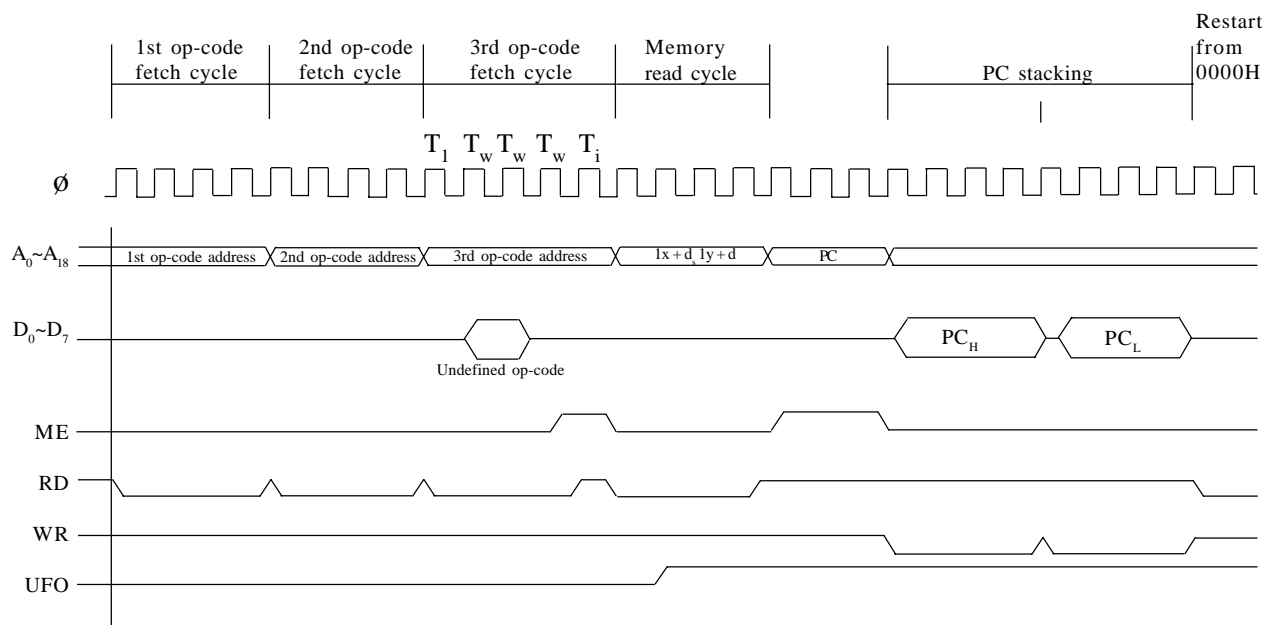


- (2) The current PC (Program Counter) value, reflecting the location after the undefined op-code, is saved on the stack.
- (3) The AB180-20 vectors to logical address 0. If logical address 0 is mapped to physical address 0, the vector is the same as for RESET. Testing the TRAP bit in ITC will reveal whether the restart at physical address 0 was caused by RESET or TRAP. The state of the UFO (Undefined Fetch Object) bit in ITC allows TRAP handling software to correctly adjust the stacked PC depending on whether the second or third byte of the op-code generated the TRAP. If UFO = 0, the starting address of the invalid instruction is equal to the stacked PC-2. If UFO = 1, the starting address of the invalid instruction is equal to the stacked PC-4. Fig 11 shows TRAP Timing.



**Figure 11**

**TRAP - 2nd Op-code Undefined**



**Figure 12**

**TRAP - 3rd Op-code Undefined**

## External Interrupts

The AB180-20 has four external hardware interrupt inputs.

- (1) NMI - Non-Maskable Interrupt
- (2) INT0 - Maskable Interrupt Level 0
- (3) INT1 - Maskable Interrupt Level 1
- (4) INT2 - Maskable Interrupt Level 2

NMI, INT0, INT1 and INT2 have fixed interrupt response modes.

### NMI - Non-Maskable Interrupt

The NMI\* interrupt input is edge sensitive and cannot be masked by software. When NMI\* is detected, the AB180-20 operates as follows:

- (1) DMAC operation is suspended by the clearing of the DME (DMA Main Enable) bit in DCNTL.
- (2) The PC is pushed onto the stack.
- (3) The contents of IEF1 are copied to IEF2. This saves the interrupt reception state that existed prior to NMI.
- (4) IEF1 is cleared to 0. This disables all external and internal maskable interrupts except NMI and TRAP).
- (5) Execution commences at logical address 66H.

The last instruction of an NMI service routine should be RETN (Return from Non-maskable Interrupt). This restores the stacked PC, allowing the interrupted program to continue, and RETN causes IEF2 to be copied to IEF1, restoring the interrupt reception state that existed prior to the NMI.

**Note:** since NMI can be accepted during AB180-20 on-chip DMAC operation, it can be used to externally interrupt DMA transfer. The NMI service routine can reactivate or abort the DMA operation as required by the application.

Special care must be taken to insure that interrupt inputs do not over run the NMI service routine. Unlimited NMI\* inputs without a corresponding number of RETN instructions will eventually cause stack overflow.

Fig 13 shows the use of NMI and RETN and Fig. 14 details NMI response timing. The NMI response sequence is activated if the internally latched edge sensitive NMI\* input is detected at the rising edge of  $\phi$ .

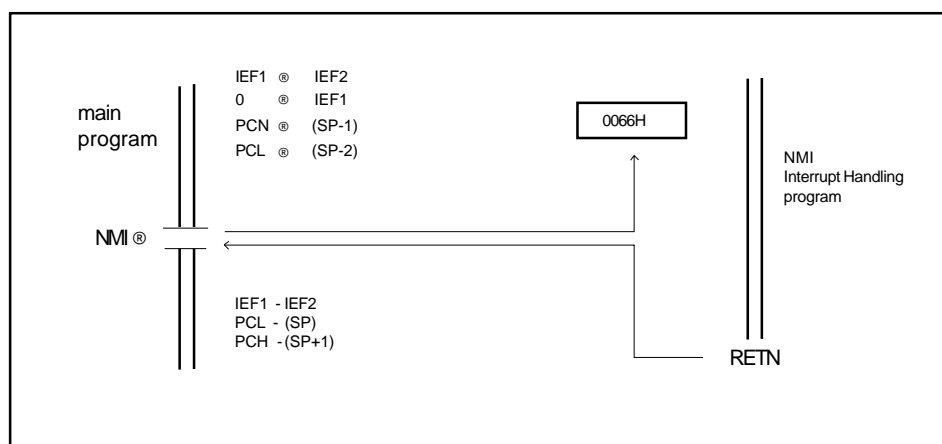
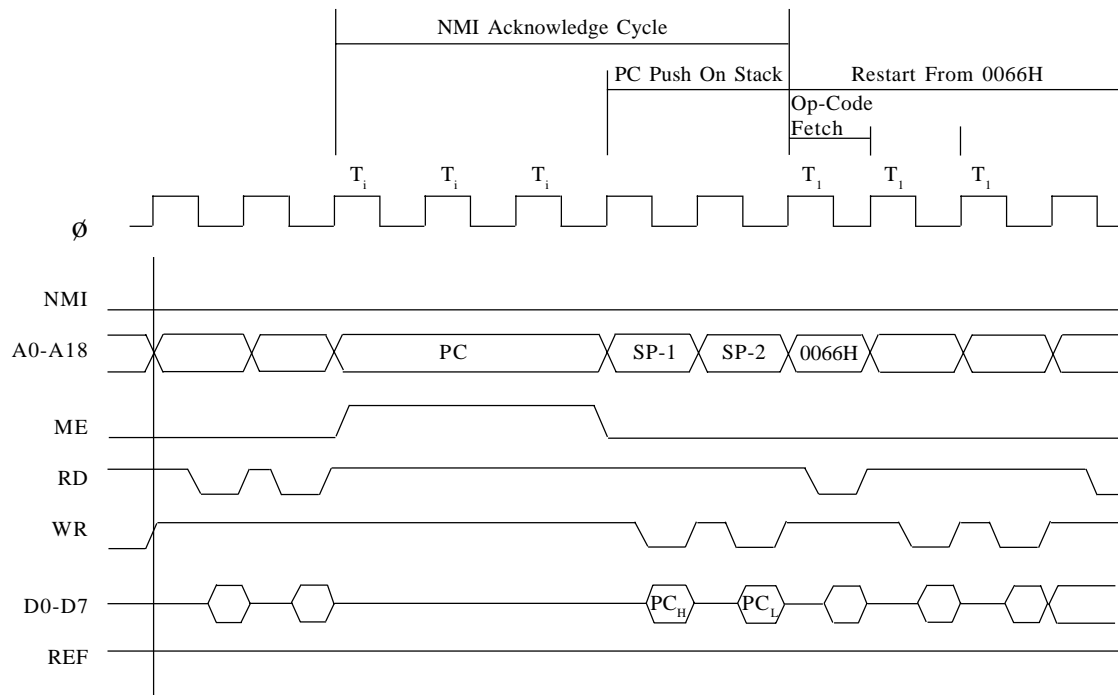


Figure 13

NMI Sequence



**Figure 14**

**NMI Timing**

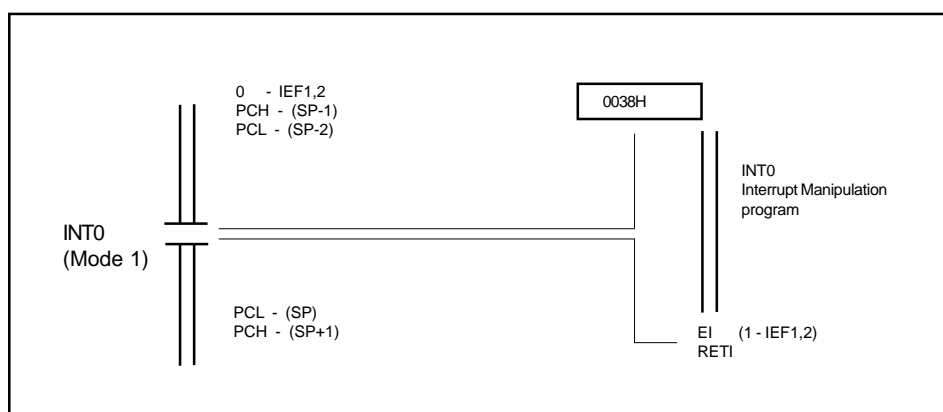
### INT0 - Maskable Interrupt Level 0

The next highest priority external interrupt after NMI is INT0. The interrupt is masked if either the IEF1 flag or the ITE0 (Interrupt Enable 0) bit in ITC are reset to 0. Note that after RESET the state is as follows:

- (1) IEF1 is 0, so INT0 is masked.
- (2) ITE0 is 1, so INT0 is enabled by execution of the EI (Enable Interrupts) instruction.

### INT0

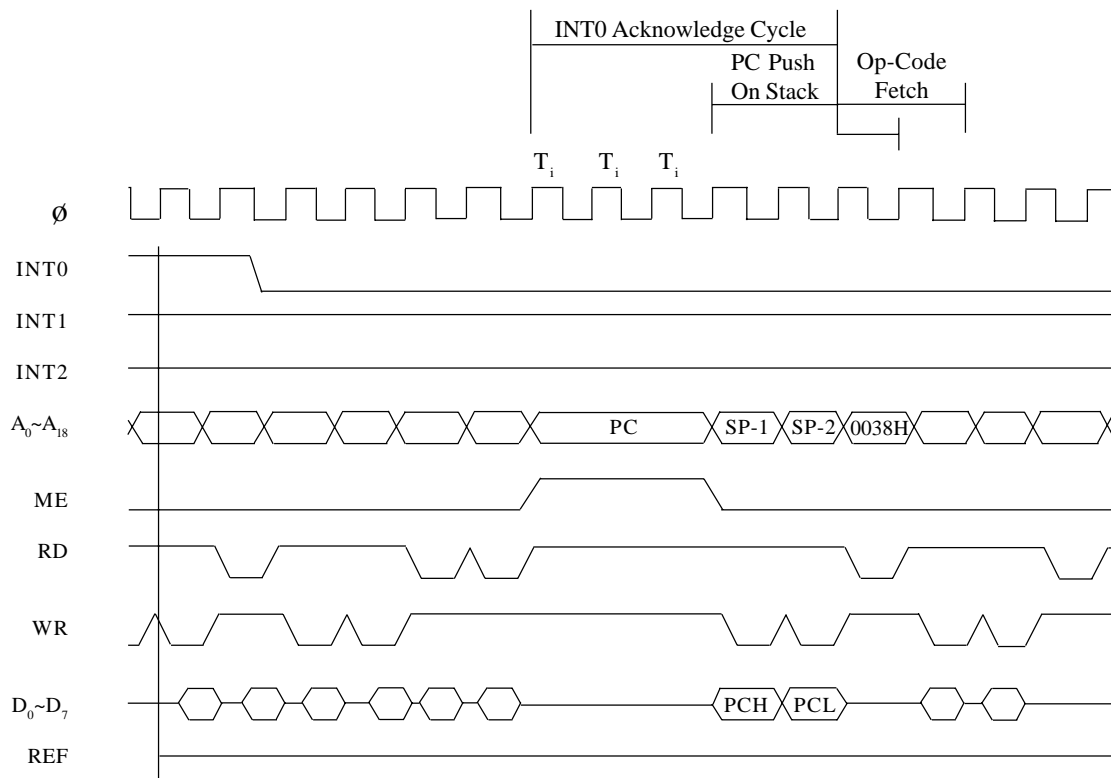
When INT0\* is received, the PC is stacked and instruction execution restarts at logical address 38H. Both IEF1 and IEF2 flags are reset to 0, disabling all maskable interrupts. The interrupt service routine normally terminates with the EI (Enable Interrupts) instruction followed by the RETI (Return from Interrupt) instruction, so that the interrupts are re-enabled. Fig 15 shows the use of INT0 and RETI. Fig 16 shows INT0 Response Timing.



**Figure 15**

**INT0 Interrupt Sequence**

Note that TRAP interrupt will occur if an invalid instruction is fetched during an interrupt acknowledge cycle.



**Figure 16**

**INT0 Timing**

## INT1, INT2

The operation of external interrupts INT1 and INT2 is a vector mode. INT1 and INT2 generate the low-order byte of vector table address using the IL (interrupt Vector Low) register and this is also the interrupt response used for all internal interrupts (except TRAP).

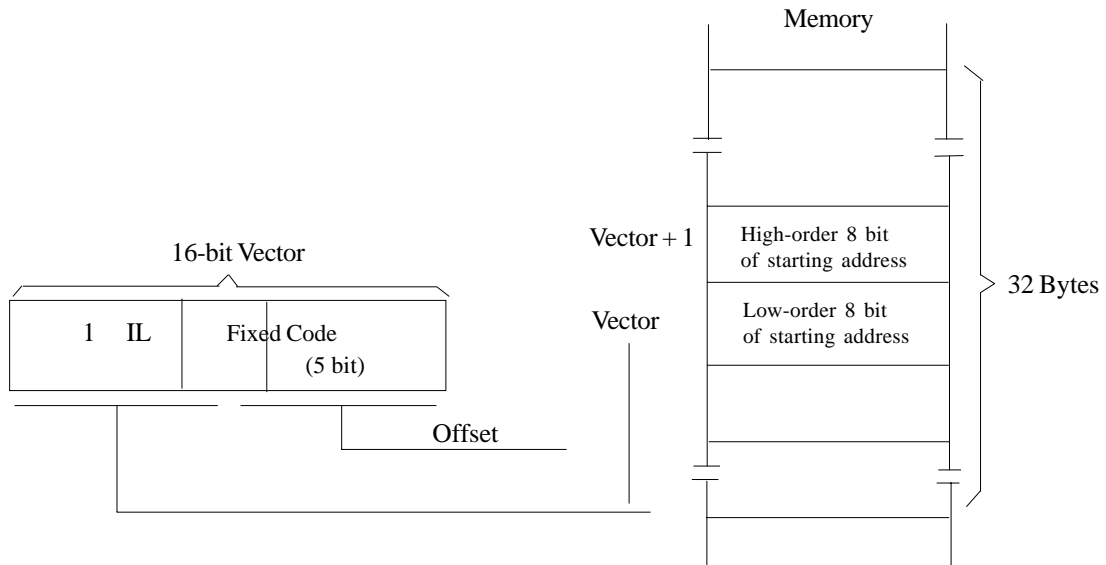
The low-order byte of vector table address is comprised of the most significant three bits of the software programmable IL register while the least significant five bits are a unique fixed value for each interrupt (INT1, INT2 and internal) source. This is shown in Fig 17

INT1\* and INT2\* are globally masked by IEF1 = 0. Each one is also individually maskable by respectively clearing the ITE1 and ITE2 (bits 1, 2) of the ITC register to 0.

During RESET, IEF1, ITE1 and ITE2 bits are reset = 0.

## Internal Interrupts

Internal interrupts (except TRAP) use the same vectored response mode as INT1 and INT2. Internal interrupts are globally masked by IEF = 0. Individual internal interrupts are enabled/disabled by programming each individual peripheral (ASCI, DMAC, PRT, CSI/O) control register. See Table 18 for a summary of the lower vector of INT1, INT2 and internal interrupts.



**Figure 17** **INT1, INT2 and Internal Interrupts Vector Acquisition**

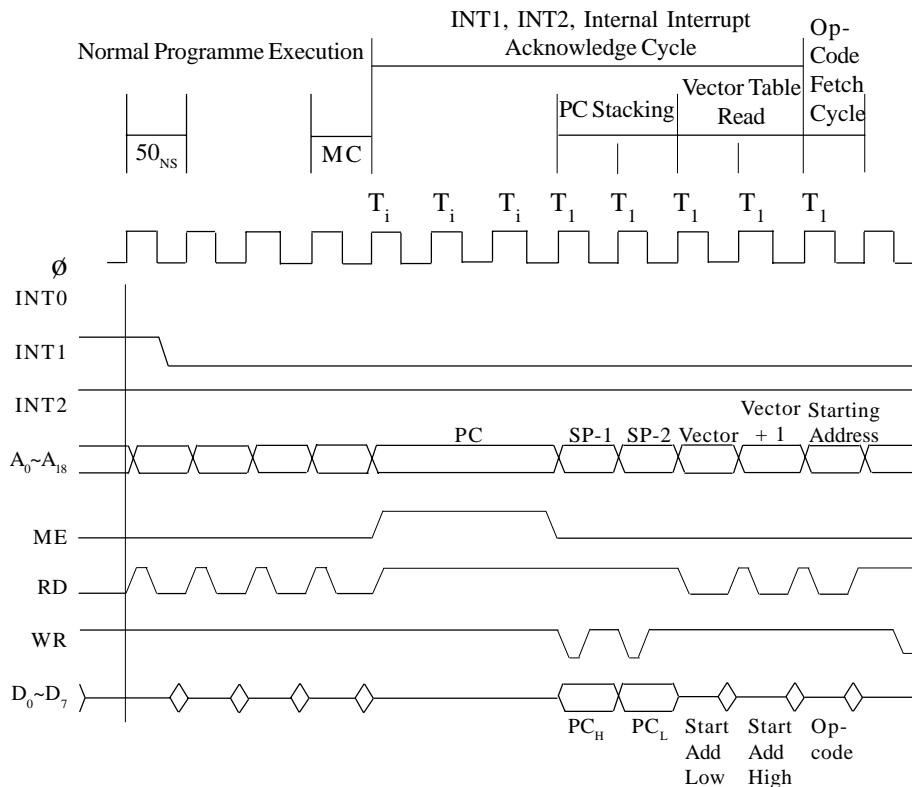
Interrupt Source	Priority	IL			Fixed Code				
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
INT1	Highest	*	*	*	0	0	0	0	0
INT2		*	*	*	0	0	0	1	0
Timer channel 0		*	*	*	0	0	1	0	0
Timer channel 1		*	*	*	0	0	1	1	0
DMA channel 0		*	*	*	0	1	0	0	0
DMA channel 1		*	*	*	0	1	0	1	0
CSI/O	Lowest	*	*	*	0	1	1	0	0
IN0 Pin		*	*	*	0	1	1	1	0

\*Programmable

**Table 18** **Interrupt Source and Lower Vector**

## Interrupt Acknowledge Cycle Timing

Fig. 19 shows interrupt acknowledge cycle timing for internal interrupts, INT1 and INT2.



MC: Machine Cycle

**Figure 19** INT1, INT2 and Internal Interrupts Timing

## Interrupt Sources and RESET

### I Register

All bits reset to 0.

1 = 0 locates the vector tables starting at logical address 0, so vectored interrupts (INT0, INT1, INT2 and internal interrupts) will overlap with fixed restart interrupts like RESET (0), NMI (66H), INT0 Mode 1 (38H) and RST (00H - 38H). The vector table(s) can be built elsewhere in memory and located on 256 bytes boundaries by reprogramming I with the LD I, A instruction.

### IL Register

Bits b7, b6 and b5 are reset to 0.

The IL register can be programmed to locate the vector for INT1, INT2 and internal interrupts on 32 bytes sub-boundaries within the 256 bytes area specified by 1.

### IEF1, IEF2 Flags

Reset to 0.

This disables all Interrupts other than NMI and TRAP.

### ITC Register

ITE0 set to 1. ITE1, ITE2 reset to 0.

INT0\* and INT2\* requires that the ITE1 and ITE2 bits be respectively set = 1 by writing to ITC.

### I/O Control Registers

Interrupt enable bits reset to 0.

All AB180-20 on-chip peripheral interrupts are disabled and can be individually enabled by writing to each I/O control register interrupt enable bit.

## I/O Functions

The AB180-20 possesses several pins that can be used for glueless I/O on a small scale. These are useful in many applications for switch and LED interfacing or even configuring programmable logic at startup. There are also two programmable frequency generators that can independently produce output frequencies from CLK/320 down to CLK/245760. With a 20 MHz clock this gives a frequency range of 62.5 KHz down to 81 Hz.

The I/O pins are designated OUT0 which is a simple output port, IN1 and IN2 which are simple input ports and IN0 which is an input which can generate a maskable interrupt on a low to high transition.

### IOREGA 00h

-	0	0	OUT0	0	0	0	0
---	---	---	------	---	---	---	---

#### OUT0

When OUT0 is reset, the OUT0 pin will go low. When OUT0 is set the OUT0 pin will immediately go high. Set to 1 during reset.

### IOREGB 01h

-	0	0	TE0EN	0	0	0	0
---	---	---	-------	---	---	---	---

#### TE0EN

Set this bit to 1 to allow the FRRQ1/TEND0 pin to be used for the TEND0 function. When set to 0 this pin will output the programmed frequency from frequency generator 1. Set to 0 during reset.

### IOREGC 02h

-	-	IN1/DS1	-	DS2	FD2	FD1	FD0
---	---	---------	---	-----	-----	-----	-----

#### IN1/DS1

When read, this bit reflects the state of the IN1 pin where a low level is represented by 0 and a high level by 1. When written this bit determines the divide ratio of the first division stage in frequency generator 0 (0=/10 and 1=/30).

**DS2**

Determines the divide ratio of the second division stage in frequency generator 0 (0=/16 and 1=/64).

**FD[2..0]**

These bits determine the division ratio of the final divider stage in frequency generator 0.

FD2	FD1	FD0	Divide Ratio
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	OFF

**IOREGD 03h**

-	-	IN2/DS2	-	DS2	FD2	FD1	FD0
---	---	---------	---	-----	-----	-----	-----

**IN2/DS2**

When read, this bit reflects the state of the IN2 pin where a low level is represented by 0 and a high level by 1. When written this bit determines the divide ratio of the first division stage in frequency generator 1 (0=/10 and 1=/30).

**DS2**

Determines the divide ratio of the second division stage in frequency generator 1 (0=/16 and 1=/64).

**FD[2...0]**

These bits determine the division ratio of the final divider stage in frequency generator 1.

FD2	FD1	FD0	Divide Ratio
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	OFF



## **IOREGGE 04h**

-	-	-	-	INTEN	INO	-	0
---	---	---	---	-------	-----	---	---

### **INTEN**

When set to one an interrupt will be generated on a low>high transition on the INO pin. Cleared to 0 on reset.

### **IN0**

Set to 1 when INO pin is high, remains at 1 until cleared by reading IOREGE whilst the IN0 pin is low. 0 on reset.

## **IOREGF 05h**

-	-	-	-	0	IN2EN	-	0
---	---	---	---	---	-------	---	---

### **IN2EN**

Selects the function of the RXS/IN2 pin.

0: Pin is RX data for CSIO (RXS)

1: Pin is IN2 input port

### **Important Note:**

In this section, where a '-' is shown in a bit position, this indicates the value is undefined for read and a don't care for write.

Where a 0 or a 1 is shown, these values **MUST** be written during writes.

# Direct Memory Access (DMA)

## DMA Controller (DMAC)

The AB180-20- contains a two channel DMA (Direct Memory Access) controller which supports high speed data transfer. Both channels (channel 0 and channel 1) have the following capabilities.

## Memory Address Space

Memory source and destination addresses can be specified anywhere within the 1M bytes physical address space using 20-bit source and destination memory addresses. Memory transfer can also cross 64k bytes physical address boundaries arbitrarily, without CPU intervention.

## I/O Address Space

I/O source and destination addresses can be directly specified anywhere within the 64k bytes I/O address space (16-bit source and destination I/O addresses).

## Transfer Length

Up to 64k bytes can be transferred based on 1 16-bit count register.

## DREQ\* Length

Level and edge sense DREQ\* input detection are selectable.

## TEND\* Output

Used to indicate DMA completion to external devices.

## Transfer Rate

Each byte transfer can occur every three bus cycles. Wait states can be inserted in DMA cycles for slow memory or I/O devices. At the system clock ( $\emptyset$ ) = 20MHz, the DMA transfer rate is as high as 6.7 megabytes/second (no wait states).

Additional feature disc for DMA interrupt request by DMA END. Each channel has additional specific capabilities.

## Channel 0

Memory  $\leftarrow$  memory, memory  $\leftarrow$  I/O, memory  $\leftarrow$  memory mapped I/O transfers

Memory address increment, decrement, np-change

Higher priority then DMAC channel 1

## Channel 1

Memory  $\leftarrow$  I/O transfer

Memory address increment, decrement

## DMAC Registers

Each channel of the DMAC (channel 0, 1) has three registers specifically associated with that channel.

## Channel 0

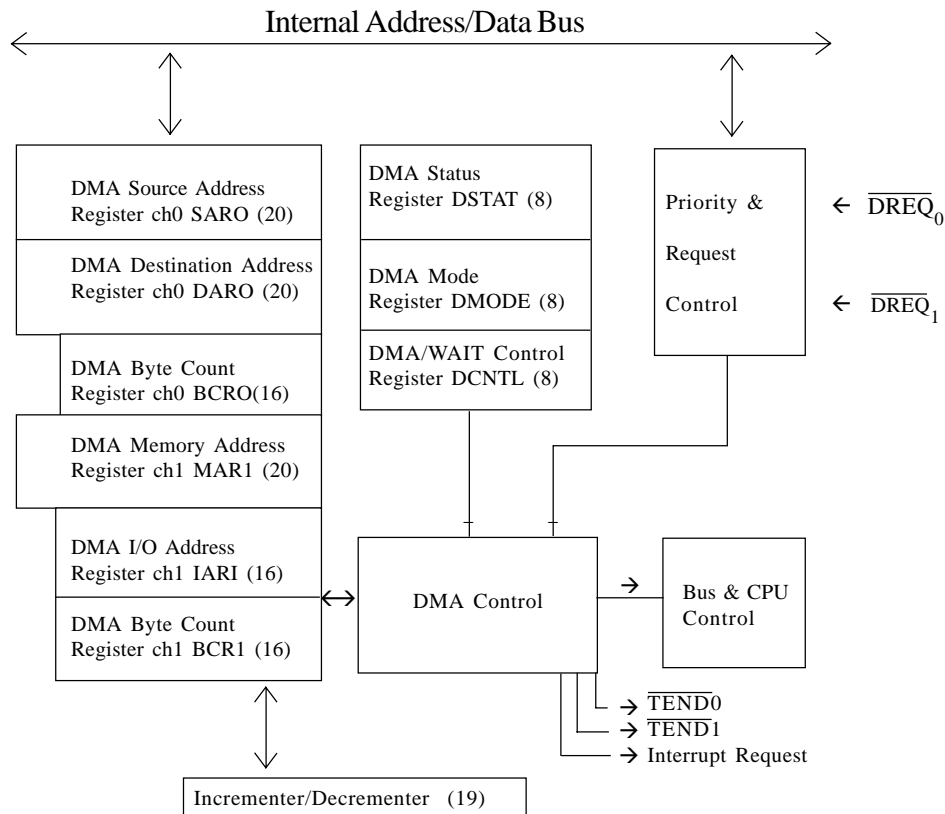
SAR0 - Source Address Register  
DAR0 - Destination Address Register  
BCR0 - Byte Count Register

## Channel 1

MAR1 - Memory Address Register  
IAR1 - I/O Address Register  
BCR1 - Byte Count Register

The two channels share three additional registers in common.

DSTAT - DMA Status Register  
DMODE - DMA Mode Register  
DCNTL - DMA Control Register



**Fig 30**

**DMAC Block Diagram**

## DMAC Register Description

### Channel 0 Source Address Register (SARO: I/O Address = 20H to 22H)

Specifies the physical source address for channel 0 transfers. The register contains 20 bits and may specify up to 1M bytes memory addresses or up to 64k bytes I/O addresses. Channel 0 source can be memory, I/O or memory mapped I/O.

### Channel 0 Destination Address Register (DAR0: I/O Address = 23H to 25H)

Specifies the physical destination address for channel 0 transfers. The register contains 20 bits and may specify up to 1M bytes memory addresses or up to 64k bytes I/O addresses. Channel 0 destination can be memory, I/O or memory mapped I/O.

### Channel 0 Bytes Count Register (BCR0: I/O Address = 26H to 27H)

Specifies the number of bytes to be transferred. This register contains 16 bits and may specify up to 64k bytes transfers. When one byte is transferred, the register is decremented by one. If "n" bytes should be transferred, "n" must be stored before the DMA operation.

### Channel 1 Memory Address Register (MAR1: I/O Address = 28H to 2AH)

Specifies the physical memory address for channel 1 transfers. This may be destination of source memory address. This register contains 20 bits and may specify up to 1M bytes memory addresses.

**Channel 1 I/O Address Register (IAR1: I/O Address = 2BH to 2CH)**

Specifies the I/O address for channel 1 transfers. This may be destination or source I/O address. This register contains 16 bits and may specify up to 64k bytes I/O addresses.

**Channel 1 Byte Count Register (BCR1: I/O Address = 2EH to 2FH)**

Specifies the number of bytes to be transferred. This register contains 16 bits and may specify up to 64k bytes transfers. When one byte is transferred, the register is decremented by one.

**DMA Status Register (DSTAT)**

DSTAT is used to enable and disable DMA transfer and DMA termination interrupts. DSTAT also allows determining the status of a DMA transfer (in progress or completed).

**DMA Status Register (DSTAT : I/O Address = 30H)**

bit 7	6	5	4	3	2	1	0
DE 1	DE 0	DWE <sub>1</sub>	DWE <sub>0</sub>	DIE1	DIE0	-	DME
R/W	R/W	W	W	R/W	R/W		R

**DE1: DMA Enable Channel 1 (bit 7)**

When DE1 = 1 and DME = 1, channel 1 DMA is enabled. When a DMA transfer terminates (BCR1 = 0), DE1 is reset to 0 by the DMAC. When DE1 = 0 and the DMA interrupt is enabled (DIE1 = 1), a DMA interrupt request is made to the CPU.

To perform a software write to DE1, DWE1\* should be written with 0 during the same register write access. Writing DE1 = 0 disables channel 1 DMA, but DMA is restartable. Writing DE1 = 1 enables channel 1 DMA and automatically sets DME (DMA Main Enable) = 1. DE1 is cleared = 0 during RESET.

**DE0: DMA Enable Channel 0 (bit 6)**

When DE0 = 1 and DME = 1, channel 0 DMA is enabled. When a DMA transfer terminates (BCR0 = 0), DE0 is reset to 0 by the DMAC. When DE0 = 0 and the DMA interrupt is enabled (DIE0 = 1), a DMA interrupt request is made to the CPU.

To perform a software write to DE0, DWE0\* should be written with 0 during the same register write access. Writing DE0 = 0 disables channel 0 DMA. Writing DE0 = 1 enables channel 0 DMA and automatically sets DME (DMA Main Enable) = 1. DE0 is cleared = 0 during RESET.

**DWE1\*: DE1 Bit Write Enable (bit 5)**

When performing any software write to DE1, DWE1\* should be written with 0 during the same access. DWE1\* write value of 0 is not held and DWE1\* is always read as 1.

**DWE0\*: DE0 Bit Write Enable (bit 4)**

When performing any software write to DE0, DWE0\* should be written with 0 during the same access. DWE0\* write value of 0 is not held and DWE0\* is always read as 1.

**DIE1: DMA Interrupt Enable Channel 1 (bit 3)**

When DIE1 is set = 1, the termination of channel 1 DMA transfer (indicated when DE1 = 0) causes a CPU interrupt request to be generated. When DIE1 = 0, the channel 1 DMA termination interrupt is disabled. DIE1 is cleared = 0 during RESET.

**DIE0: DMA Interrupt Enable Channel 0 (bit 2)**

When DIE0 is set = 1, the termination channel 0 of DMA transfer (indicated when DE0 = 0) causes a CPU interrupt request to be generated. When DIE0 = 0, the channel 0 DMA termination interrupt is disabled. DIE0 is cleared = 0 during RESET.

**DME: DMA Main Enable (bit 0)**

A DMA operation is only enabled when its DE bit (DE0 for channel 0, DE1 for channel 1) and the DME bit are set = 1.

When NMI occurs, DME is reset = 0, thus disabling DMA activity during the NMI interrupt service routine. To restart DMA, DE0 and/or DE1 should be written with 1 (even if the contents are already 1). This automatically sets DME = 1, allowing DMA operations to continue. Note that DME cannot be directly written. It is cleared = 0 by NMI or indirectly set = 1 by setting DE0 and /or DE1 = 1. DME is cleared = 0 during RESET.

**DMA Mode Register (DMODE)**

DMODE is used to set the addressing and transfer mode for channel 0.

**DMA Mode Register (DMODE : I/O Address = 31H)**

bit	7	6	5	4	3	2	1	0
	-	-	DM1	DM0	SM1	SM0	MMOD	-
			R/W	R/W	R/W	R/W		

**DM1, DM0: Destination Mode Channel 0 (bits 5-4)**

Specifies whether the destination for channel 0 transfers is memory, I/O or memory mapped I/O and the corresponding address modifier. DM1 and DM0 are cleared = 0 during RESET.

DM1	DM0	Memory/I/O	Address Increment/Decrement
0	0	Memory	+1
0	1	Memory	-1
1	0	Memory	fixed
1	1	I/O	fixed

**SM1, SM0: Source Mode Channel 0 (bits 3-2)**

Specifies whether the source for channel 0 transfers is memory, I/O or memory mapped I/O and the corresponding address modifier. SM1 and SM0 are cleared = 0 during RESET.

SM1	SM0	Memory/I/O	Address Increment/Decrement
0	0	Memory	+1
0	1	Memory	-1
1	0	Memory	fixed
1	1	I/O	fixed

The following Table shows all DMA transfer mode combinations of DM0, DM1, SM0, SM1. Since I/O  $\ll$  transfers are not implemented, twelve combinations are available.

DM1	DM0	SM1	SM0	Transfer Mode	Address Increment/Decrement
0	0	0	0	Memory to Memory	SAR+1, DAR+1
0	0	0	1	Memory to Memory	SAR-1, DAR+1
0	0	1	0	Memory mapped I/O to Memory	SAR fixed, DAR+1
0	0	1	1	I/O to Memory	SAR fixed, DAR+1
0	1	0	0	Memory to Memory	SAR+1, DAR-1
0	1	0	1	Memory to Memory	SAR-1, DAR-1
0	1	1	0	Memory mapped I/O to Memory	SAR fixed, DAR-1
0	1	1	1	I/O to Memory	SAR fixed, DAR-1
1	0	0	0	Memory to Memory mapped I/O	SAR+1, DAR fixed
1	0	0	1	Memory to Memory mapped I/O	SAR-1, DAR fixed
1	0	1	0	reserved	
1	0	1	1	reserved	
1	1	0	0	Memory to I/O	SAR+1, DAR fixed
1	1	0	1	Memory to I/O	SAR+1, DAR fixed
1	1	1	0	reserved	
1	1	1	1	reserved	

For channel 0 DMA with I/O source or destination, the DREQ0\* input times the transfer.

**Important Note:** When using I/O to memory or memory to I/O transfers always ensure that bits 16-19 are set to zero in the SAR and DAR respectively.

### MMOD DMA Bus Mode

When this bit is written with 1 the DMA takes all available bus cycles in memory-memory mode and program execution is paused until the end of the transfer (Burst Mode). When set to 0, the bus is shared between the execution engine and DMA permitting program execution to continue whilst the transfer takes place (Cycle Steal).

### DMA/WAIT Control Register (DCNTL)

DCNTL controls the insertion of wait states into DMAC (and CPU) accesses of I/O and the DMA request mode for each DREQ\* (DREQ\* and DREQ1\*) input is defined as level or edge sense. DCNTL also sets the DMA transfer mode for channel 1, which is limited to memory  $\ll$  I/O transfers.

# DMA/WAIT Control Register (DCNTL : I/O Address = 32H)

bit 7	6	5	4	3	2	1	0
—	—	IWI1	IWI0	DMS1	DMS0	DIM1	DIM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

## IWI1, IWI0: I/O Wait Insertion (bits 5-4)

Specifies the number of wait states introduced into CPU or DMAC I/O access cycles. IWI1 and IWI0 are set = during RESET (see the *Wait State Control* section under *Memory Management Unit* for details).

## DMS1, DMS0: DMA Request Sense (bits 3-2)

DMS1 and DMS0 specify the DMA request sense for channel 0 (DREQ0\*) and channel 1 (DREQ1\*) respectively. When reset to 0, the input is level sense and when set to 1 the input is edge sense. DMS1 and DMS0 are cleared = 0 RESET.

## DIM1, DIM0: DMA Channel 1 I/O and Memory Mode (bits 1-0)

Specifies the source/destination and address modifier for channel 1 memory « I/O transfer modes. IM1 and IM0 are cleared = 0 during RESET.

DM1	DM0	Memory/ I/O	Address Increment/Decrement
0	0	Memory to I/O	MAR+1, IAR fixed
0	1	Memory to I/O	MAR-1, IAR fixed
1	0	I/O to Memory	IAR fixed, MAR+1
1	1	I/O to Memory	IAR fixed, MAR-1

Table 31

Channel 1 Transfer Mode

## DMA Operation

This section discusses the three DMA operation modes for channel 0, memory « memory, memory « I/O and memory « memory mapped I/O. It also describes the operation of channel 0 DMA with the on-chip ASCI (Asynchronous Serial Communication Interface), as well as Channel 1 DMA.

### Memory « Memory - Channel 0

For memory « memory transfers, the external DREQ0\* input is not used for DMA transfer timing. The DMA operation will automatically proceed until termination as shown by byte count (BCR0) = 0.

Perform the following operations to initiate memory memory DMA for channel 0.

- (1) Load the memory source and destination addresses into SAR0 and DAR0.
- (2) Specify memory memory mode addresses increment/decrement in the SM0, SM1, DM0 and DM1 bits of DMODE.
- (3) Load the number of bytes to transfer in BCR0.
- (4) Program DE0 = 1 (with DWE0\* = 0 in the same access) in DSTAT and the DMA operation will start 1 machine cycle later. If interrupt occurs at the same time, the DIE0 bit should be set = 1.

### **Memory « I/O (Memory Mapped I/O) - Channel 0**

For memory « I/O (and memory « memory mapped I/O) the DREQ0\* input is used to time the DMA transfers. In addition, the TEND0\* (transfer End) output is used to indicate the last (byte count register, BCR0 = 0) transfer.

The DREQ0\* input can be programmed as level or edge sensitive.

When level sense is programmed, the DMA operation begins when DREQ0\* is sampled low. If DREQ0\* is sampled high, control is relinquished to the AB180-20-CPU after the next DMA byte transfer.

When edge sense is programmed, DMA operation begins at the falling edge of DREQ0\*. If another falling edge is during write cycle, the DMAC continues operating. If an edge is not detected, the CPU is given control after the current byte DMA transfer completes. The CPU will continue operating until a DREQ0\* falling is detected at which time the DMA operation will (re) start.

During the transfer for channel 0, the TEND0\* output will go low synchronous with the write cycle of the last (BCR0 = 0) DMA transfer.

To use the TEND0\* function 10h must be written to 10REGB (01h), setting bit 4.

To initiate memory « I/O (and memory « memory mapped I/O) DMA transfer for channel 0, perform the following operations.

- (1) Load the memory and I/O or memory I/O source and destination addresses into SAR0 and DAR0. I/O addresses (not memory mapped I/O) are limited to 16 bits (A0-A15). Make sure that bits A16 and A17 are 0 (A18 is a don't care) to correctly enable the external DREQ0\* input.
- (2) Specify memory « I/O or memory « memory mapped I/O mode and address increment/decrement in the SM0, SM1, DM0 and DM1 bits of DMODE.
- (3) Load the number of bytes to transfer in BCR0.
- (4) Specify whether DREQ0\* is edge or level sense by programming the DMS0 bit of DCNTL.
- (5) Enable or disable DMA termination interrupt with the DIE0 bit in DSTAT.
- (6) Program DE0 = 1 (with DWE0\* = 0 in the same access) in DSTAT and the DMA operation will begin under the control of the DREQ0\* input.

### **Channel 1 DMA**

DMA channel 1 can perform memory « I/O transfers. Except for different registers and status/control bits, operation is exactly the same as described for channel 0 memory « I/O DMA.



To initiate DMA channel 1 memory « I/O operation perform the following operations:

- (1) Load the memory address (19 bits) into MAR1.
- (2) Load the I/O address (16 bits) into IAR1.
- (3) Program the source/destination and address increment/decrement mode using the DIM1 and DIM0 bit in DCNTL.
- (4) Specify whether DREQ1\* is level or edge sense in the DMS1 bit in DCNTL.
- (5) Enable or disable DMA termination interrupt with the DIE1 bit in DSTAT.
- (6) Program DE1 = 1 (with DWE1\* = 0 in the same access) in DSTAT and the DMA operation with the external I/O device will begin using the external DREQ1\* input and TEND1\* output.

## **DMA BUS Timing**

When memory (and memory mapped I/O) is specified as a source or destination, ME\* goes low during the memory access. When I/O is specified as a source or destination, IOE\* goes low during the I/O access.

When I/O (and memory mapped I/O) is specified as a source or destination, the DMA timing is controlled by the external DREQ\* input and the TEND\* output indicates DMA termination. The external I/O devices may not overlap addresses with internal I/O and control register, even using DMA.

Wait states can be inserted by programming the on-chip wait state generator or using the external WAIT\* input.

For memory to memory transfer (channel 0 only), the external DREQ0\* input is ignored.

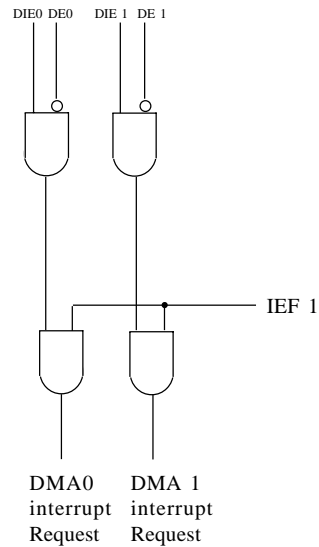
## **DMAC Channel Priority**

For simultaneous DREQ\* request, channel 0 has priority over channel 1. When channel 0 is performing a memory memory transfer, channel 1 cannot operate until the channel 0 operation has terminated. If channel 1 is operating, channel 0 cannot operate until channel 1 releases control of the bus.

## **DMAC and BUSREQ\*, BUSACK\***

The BUSREQ\* and BUSACK\* inputs allow another bus master to take control of the AB180-20 bus. BUSREQ\* and BUSACK\* have priority over the on-chip DMAC and will suspend DMAC operation. The DMAC releases the bus to the external bus master at the breakpoint of the DMAC memory or I/O access. Since a single byte DMAC transfer requires a read write cycle, it is possible for the DMAC to be suspended after the DMAC read, but before the DMAC write. Even in this case, when the external master releases the AB180-20-DX bus (BUSREQ\* HIGH), the on-chip DMAC will correctly continue the suspended DMA operation.

## DMAC Internal Interrupts



**Figure32 DMAC Interrupt Request Circuit Diagram**

DE0 and DE1 are automatically cleared = 0 by the AB180-20-DX at the completion (byte count = 0) of a DMA operation for channel 0 and channel 1 respectively. They remain 0 until a 1 is written. Since DE0 and DE1 use level sense, an interrupt will occur if the CPU IEF1 is set to 1. So the DMA termination interrupt service routine should disable further DMA interrupt (by programming the channel DIE bit = 0) before enabling CPU interrupts (i.e. IEF1 is set = 1). After reloading the DMAC address and count register, the DIE bit can be set = 1 to reenble the channel interrupt, and at the same time DMA can resume by programming the channel DE bit = 1.

### DMAC and NMI

NMI, unlike all other interrupts, automatically disables DMAC operation by clearing the DME bit of DSTAT. So the NMI interrupt service routine may respond to time critical events without delay due to DMAC bus usage and NMI can be effectively used as an external DMA abort input, recognising that both channels are suspended by the clearing of DME.

If the falling edge of NMI occurs before the falling clock of the state prior to T3 (T2 or Tw), the DMAC will be suspended and the CPU will start the NMI response at the end of the current cycle.

By setting a channels DE bit = 1, that channels operation can be restarted and DMA will correctly resume from the point at which it was suspended by NMI.

### DMAC and RESET

During RESET the bits in DSTAT, DMODE and DCNTL are initialised as stated in their individual register description. Any DMA operation in progress is stopped allowing the CPU to use the bus to perform the RESET sequence but the address register (SAR0, DAR0, MAR1, IAR1) and byte count register (BCR0, BCR1) contents are not stopped during RESET.

## Programmable Reload Timer (PRT)

The AB180-20 contains two 16-bit Programmable Reload Timers. Each PRT channel contains a 16-bit down counter and a 16-bit reload register. The down counter can be directly read and written and a down counter overflow interrupt can be programmed as enabled or disabled. PRT channel 1 also has a TOUT output pin (pin 32 - multiplexed with A19) which can be set high or low and toggled, so PRT1 can be programmed to perform output waveform generation.

### PRT Block Diagram

Fig 33 shows the PRT block diagram. The two channels have separate timer data and reload registers and a common status/control register. The PRT input clock for both channels is equal to the system clock ( $\phi$ ) divided by 20.

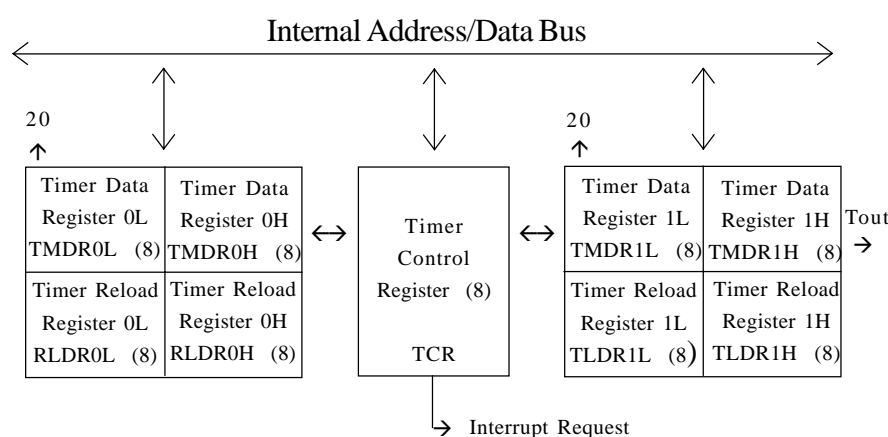


Figure 33

PRT Block Diagram

### PRT Register Description

#### Timer Data Register (TMDR: I/O Address = CH0: 0DH, 0CH CH1: 15H, 14H)

PRT0 and PRT1 each have 16-bit Timber Data Registers (TMDR). TMDR0 and TMDR1 are each accessed as low and high byte registers (TMDR0H, TMDR0L and TMDR1H, TMDR1L). During RESET, TMDR0 and TMDR1 are set = FFFFH.

TMDR is decremented once every twenty  $\phi$  clocks. When TMDR counts down to 0, it is automatically reloaded with the value contained in the Reload Register (RLDR).

TMDR can be read and written by software using the following procedures. The read procedure uses a PRT internal temporary storage to return accurate data without requiring the timer to be stopped. The write procedure requires that the timer be stopped.

For reading (without stopping the timer), TMDR must be read in the order of lower byte - higher byte (TMDRnL, TMDRnH). The lower byte read (TMDRnL) will store the higher byte value in an internal register. The following higher byte read (TMDRnH) will access this internal register. This procedure ensures timer data validity by eliminating the problem of potential 16-bit timer updating between each 8 bit read. Note that reading TMDR in higher byte - lower byte order may result in invalid data, and that there are implications for TMDR higher byte internal storage for applications which may read only the lower and/or higher bytes. In normal operation all TMDR read routines should access both the lower and higher bytes, in that order.

For writing, the TMDR down counting must be inhibited using the TDE(Timber Down Count Enable) bits in the TCR (Timer Control Register), following which any or both higher and lower bytes of TMDR can be freely written (and read) in any order.

#### **Reload Register (RLDR: I/O Address = CH0: 0EH, 0FH CH1: 16H, 17H)**

PRT0 and PRT1 each have 16 bit timer Reload Registers (RLDR0). RLDR0 and RLDR1 are each accessed as low and high byte registers (RLDR0L and RLDR0H, RLDR1L, RLDR1H). During RESET RLDR0 and RLDR1 are set = FFFFH.

When the TMDR counts down to 0, it is automatically reloaded with the contents of RLDR.

#### **Timer Control Register**

TCR monitors both channel (PRT0, PRT1) TMDR status and controls enabling and disabling of both counting and interrupts as well as controlling the output pin (A19/TOUT-pin 32) for PRT 1.

#### **Timer Control Register (TCR : I/O Address = 10H)**

bit 7	6	5	4	3	2	1	0
TIF1	TIF0	TIE1	TIE0	TOC1	TOC0	TDE1	TDE0
R	R	R/W	R/W	R/W	R/W	R/W	R/W

#### **TIF1: Timer Interrupt Flag (bit 7)**

When TMDR1 decrements to 0, TIF1 is set = 1. This can generate an interrupt request if enabled by TIE1 = 1. TIF1 is reset = 0 when TCR is read and the higher or lower byte of TMDR1 are read. During RESET, TIF1 is cleared = 0.

#### **TIF0: Timer Interrupt Flag (bit 6)**

When TMDR0 decrements to 0, TIF0 is set = 1. This can generate an interrupt request if enabled by TIE0 = 1. TIF0 is reset = 0 when TCR is read and the higher or lower byte of TMDR0 are read. During RESET, TIF0 is cleared = 0.

#### **TIE1: Timer Interrupt Enable (bit 5)**

When TIE1 is set = 1, TIF1 = 1 will generate a CPU interrupt request. When TIE1 is reset = 0, the interrupt request is inhibited. During RESET, TIE1 is cleared = 0.

#### **TIE0: Timer Interrupt Enable (bit 4)**

When TIE0 is set = 1, TIF0 = 1 will generate a CPU interrupt request. When TIE0 is reset = 0, the interrupt request is inhibited. During RESET, TIE0 is cleared = 0.

### TOC1, 0: Timer Output Control (bits 3-2)

TOC1 and TOC0 control the output of PRT1 using the multiplexed A19/TOUT pin as shown below. During RESET, TOC1 and TOC0 are cleared = 0. This selects the address function for A19/TOUT. By programming TOC1 and TOC0, the A18/TOUT pin can be forced HIGH, LOW or toggled TMDR1 decrements to 0.

TOC1	TOC0	OUTPUT
0	0	Inhibited (A19/TOUT pin is selected as an address output function)
0	1	toggled
1	0	0
1	1	1

### TDE1, 0: Timer Down Count Enable (bits 1-0)

TDE1 and TDE0 enable and disable down counting for TMDR1 and TMDR0 respectively. When TDEn is set = 1, down counting is stopped and TMDRn can be freely read or written TDE1 and TDE0 are cleared = 0 during RESET and TMDRn will not decrement until TDEn is set = 1.

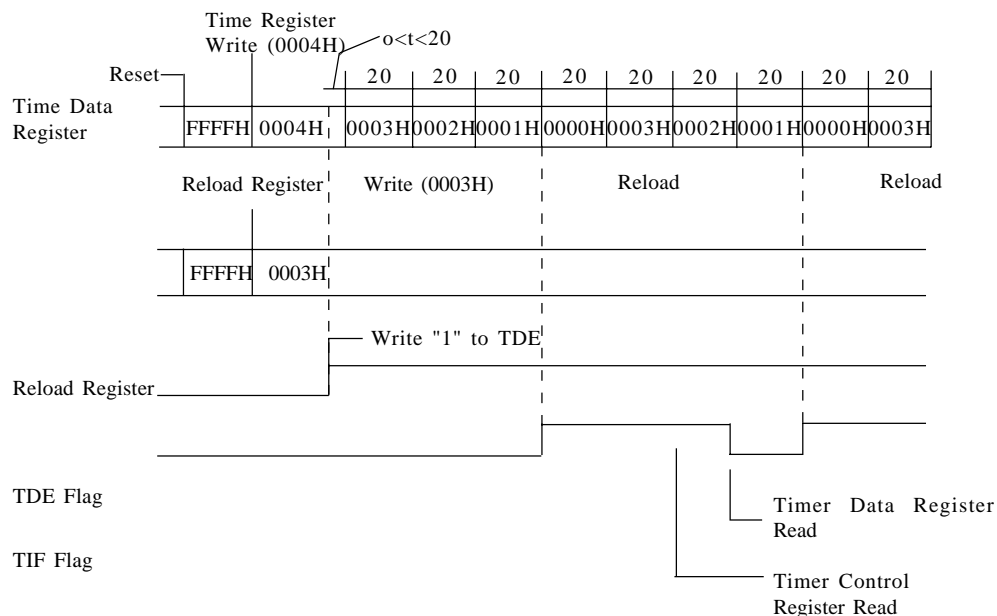


Figure 34

Timer Operation Timing

## Clocked Serial I/O Port (CSI/O)

The AB180-20 includes a simple, high-speed clock synchronous serial I/O port. The CSI/O includes transmit/receive (half duplex), fixed 8-bit data and internal or external data clock selection. High-speed operation (baud rate as high as 1M bits/second at  $f_c = 20$  MHz) is provided. The CSI/O is ideal for implementing a multiprocessor communication link between the AB180-20 and single ship controllers as well as additional AB180-20 CPUs. These secondary devices may typically perform a portion of the system I/O processing such as keyboard scan/decode, LCD interface, etc.

### CSI/O Block Diagram

The CSI/O consists of two register - the Transmit/Receive Data Register (TRDR) and Control Register

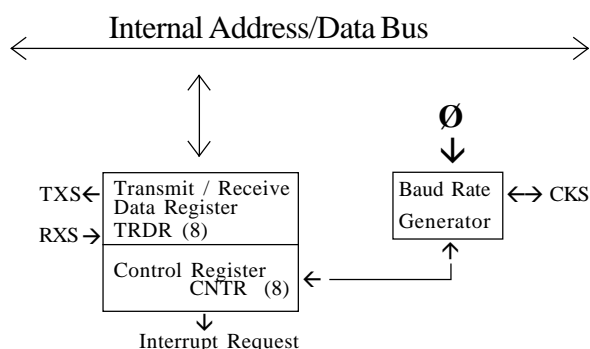


Figure 35

CSI/O Block Diagram

### CSI/O Register Description

Transmit/Receive Data Register (TRDR: I/O Address = OBH)

TRDR is used for both CSI/O transmission and reception, so the system design must ensure that the constraints of half-duplex operations are met (Transmit and receive operations can't occur simultaneously). For example, if a CSI/O transmission is attempted at the same time the CSI/O is receiving data, a CSI/O will not work. Because the TRDR is not buffered, attempting to perform a CSI/O transmit while the previous transmit data is still being shifted out causes the shift data to be immediately updated, thereby corrupting the transmit operation in progress. Similarly, reading TRDR while a transmit or receive is in progress should be avoided.

Control/Status Register (CNTR: I/O Address = OAH)

CNTR is used to monitor CSI/O status, enable and disable the CSI/O enable and disable interrupt generation and select the data clock speed and source.

bit 7	6	5	4	3	2	1	0
EF	EIE	RE	TE	-	SS2	SS1	SS0
R	R/W	R/W	R/W		R/W	R/W	R/W

#### EF: End Flag (bit 7)

EF is set = 1 by the CSI/O to indicate completion of an 8 bit data transmit or receive operation. If EIE (End Interrupt Enable) bit = 1 when EF is set = 1, a CPU interrupt request will be generated. Program access of TRDR should only occur if EF = 1 to generate a CPU interrupt request. The interrupt request is inhibited if EIE is reset = 0. EIE is cleared. = 0 during RESET.

**RE: Receive Enable (bit 5)**

A CSI/O receive operation is started by setting RE = 1. When RE is set = 1, the data clock is enabled. In internal clock mode, the data clock is output from the CKS pin. In external clock mode, the clock is input on the CKS pin. In either case, data is shifted in on the RXS pin in synchronisation with the (internal or external) data clock. After receiving 8 bits of data, the CSI/O automatically clears RE = 0, EF is set = 1 and an interrupt (if enabled by EIE = 1) will be generated. RE and TE should never both be set = 1 at the same time. RE is cleared = 0 during RESET and IOSTOP mode.

**Note:** the RXS pin (pin 52) is multiplexed with the IN2 input port pin. In order to enable the RXS function, the IN2EN bit in IOREGF should be reset = 0.

**TE: Transmit Enable (bit 4)**

A CSI/O transmit operation is started by setting TE = 1. When TE is set = 1, the data block is enabled. In internal clock mode, the data clock is output from the CKS pin. In external clock mode, the clock is input on the CKS pin. In both cases, data is shifted out on the TXS pin synchronous with the (internal or external) data clock. After transmitting 8 bits of data, the CSI/O automatically clears TE = 0, EF is set = 1 and an interrupt (if enabled by EIE = 1) will be generated. TE and RE should never both be set = 1 at the same time. TE is cleared = 0 during RESET and IOSTOP mode.

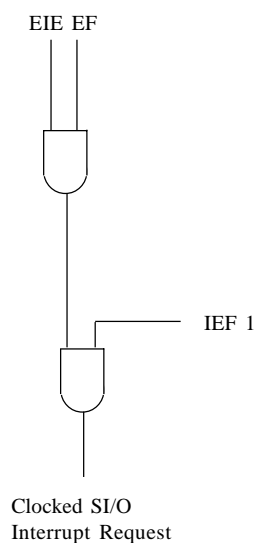
**SS2,1, 0: Speed Select 2, 1, 0 (bits 2-0)**

SS2, SS1 and SS0 select the CSI/O transmit/receive clock source and speed. SS2, SS1 and SS0 are all set = 1 during RESET.

After RESET, the CKS pin is configured as an external clock input (SS2, SS1, SS0 = 1). Changing these values causes CKS to become an output pin and the selected clock will be output when transmit or receive operations are enabled.

**CSI/O Interrupts**

The CSI/O interrupt request circuit is shown below .



**Figure 36 CSI/O Interrupt Circuit Diagram**

## CSI/O Operation

The CSI/O can be operated using status polling or interrupt driven algorithms.

### Transmit - Polling

1. Poll the TE bit in CNTR until = 0.
2. Write the transmit data into TRDR.
3. Set the TE bit in CNTR = 1.
4. Repeat 1 to 3 for each transmit data byte.

### Transmit - Interrupts

1. Poll the TE bit in CNTR until = 0.
2. Write the first transmit data byte into TRDR.
3. Set the TE and EIE bits in CNTR = 1.
4. When the transmit interrupt occurs, write the next transmit data byte into TRDR.
5. Set the TE bit in CNTR = 1.
6. Repeat 4 to 5 for each transmit data byte.

### Receive - Polling

1. Poll the RE bit in CNTR until = 0.
2. Set the RE bit in CNTR = 1.
3. Poll the RE bit in CNTR until = 0.
4. Read the receive data from TRDR.
5. Repeat 2 to 4 for each receive data byte.

### Receive - Interrupts

1. Poll the RE bit in CNTR until = 0.
2. Set the RE and EIE bit in CNTR = 1.
3. When the receive interrupts occur read the receive data from TRDR.
4. Set the RE bit in CNTR = 1.
5. Repeat 3 to 4 for each receive data byte.

## CSI/O Operation Notes

- (1) Disable the transmitter and receiver (TE and RE = 0) before initialising or changing the baud rate. When changing the baud rate after completion of transmission or reception, a delay of at least one bit time is required before baud rate modification.
- (2) When RE or TE is cleared = 0 by software, a corresponding receive or transmit operation is immediately terminated. Other than in exceptional circumstances, TE or RE should only be cleared = 0 when EF = 1.
- (3) Simultaneous transmission and reception is not possible, so TE and RE should not both be 1 at the same time.

## CSI/O and RESET

During RESET each bit in the CNTR is initialised as defined in the CNTR register description. CSI/O transmit and receive operations in progress are aborted during RESET but the contents of TRDR are not changed.



## CSI/O Operation Timing Notes

- (1) Note that transmitter clocking and receiver sampling timings are different from internal and external clocking modes. Fig 37 to Fig 40 shows CSI/O Transmit/Receive Timing.
- (2) The transmitter and receiver should be disabled ( $TE$  and  $RE = 0$ ) when initialising or changing the baud rate.

Transmit data outputs from the falling edge of the clock up until the next falling edge.

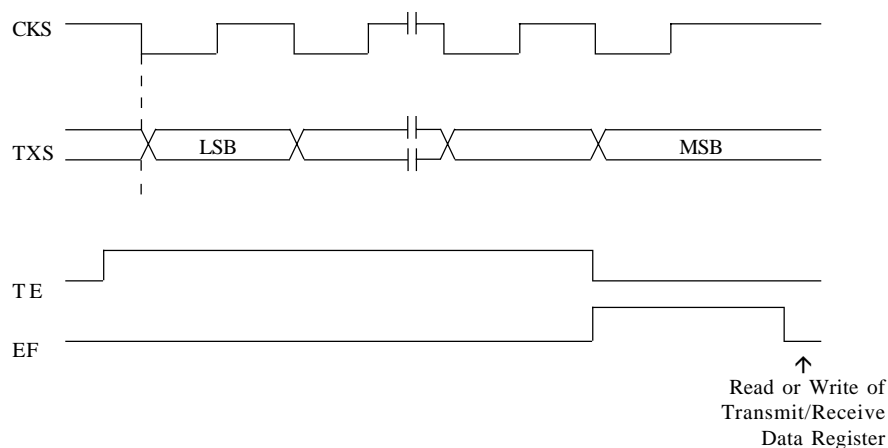


Figure 37

Transmit Timing - Internal Clock

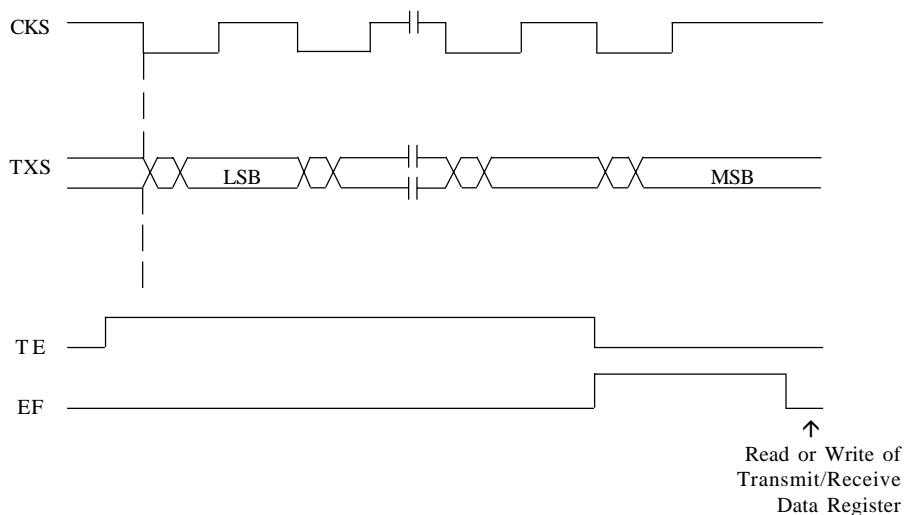
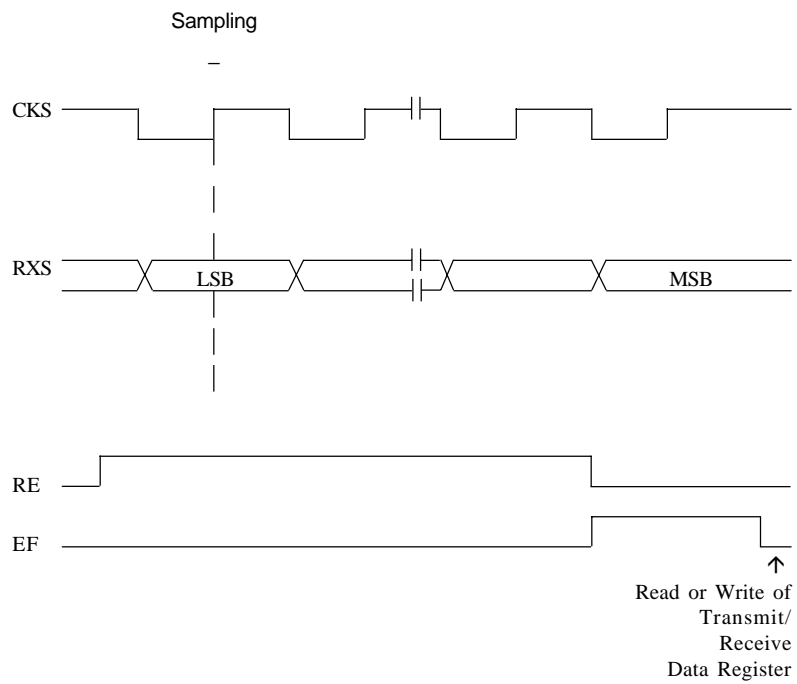


Figure 38

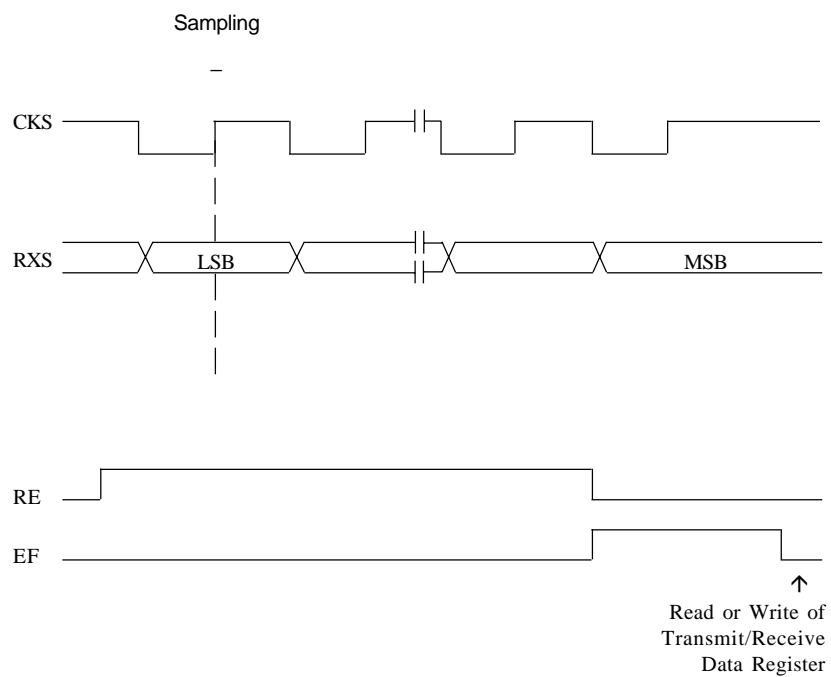
Transmit Timing - External Clock

Receive data is latched on to the rising edge of the clock



**Figure 39**

**Receive Timing - Internal Clock**



**Figure 40**

**Receive Timing - External Clock**

## Bus Timing Information

### Basic CPU Timing

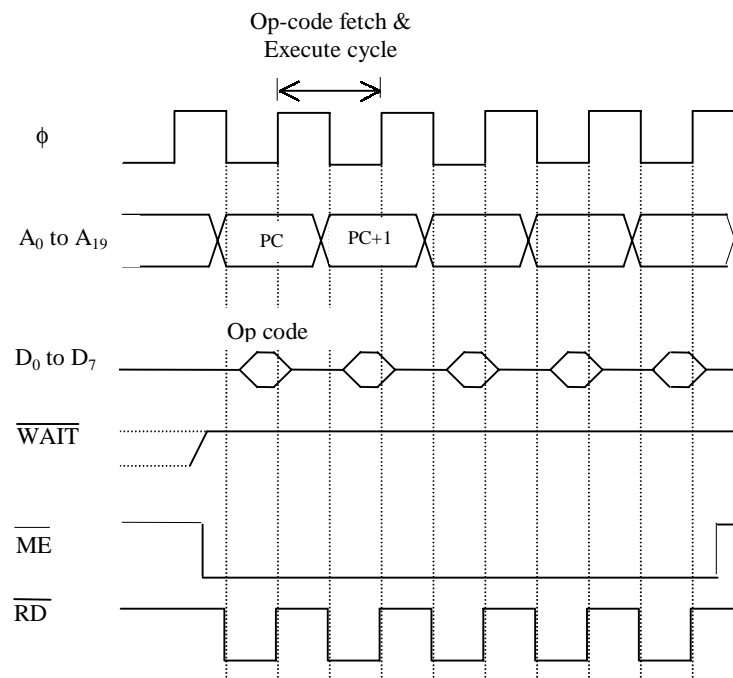


Figure 41

## Read Cycle Timing

No wait states

Symbol	Meaning	Min	Nom	Max
Tpp	PHI Clock Period	50	-	-
Tpl	PHI low time	25	-	-
Tph	PHI high time	25	-	-
Tav	PHI high to address valid	16	-	22
Tmea	PHI high to ME asserted	15	-	21
Tmed	PHI high to ME deasserted	16	-	21
Tioa	PHI high to IOE asserted	15	-	21
Tiod	PHI high to IOE deasserted	15	-	21
Trsa	PHI low to RD asserted	2	-	7
Trsd	PHI high to RD deasserted	3	-	8
Tdsu	Data setup time	6	-	-
Tdh	Data hold time	0	-	-

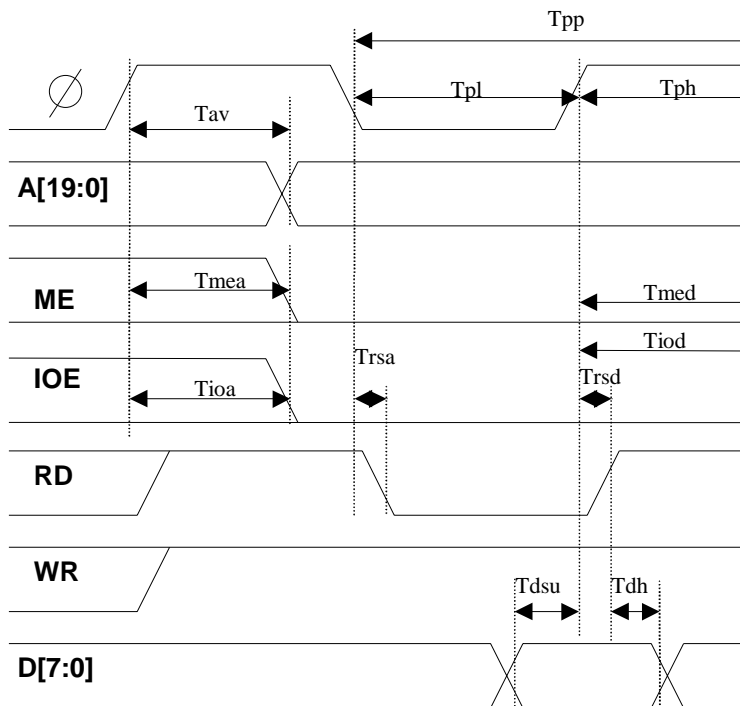


Figure 42

ME is asserted in memory space read cycles, IOE is asserted in I/O space reads.

## Write Cycle Timing

No wait states

Symbol	Meaning	Min	Nom	Max
Tpp	PHI Clock Period	50	-	-
Tpl	PHI low time	25	-	-
Tph	PHI high time	25	-	-
Tav	PHI high to address valid	16	-	22
Tmea	PHI high to ME asserted	15	-	21
Tmed	PHI high to ME deasserted	16	-	21
Tioa	PHI high to IOE asserted	15	-	21
Tiod	PHI high to IOE deasserted	15	-	21
Twsa	PHI low to WR asserted	2	-	5
Twsd	PHI high to WR deasserted	2	-	5
Tdv	Data out valid delay time	27	-	30
Tdh	Data out hold time	25	-	-

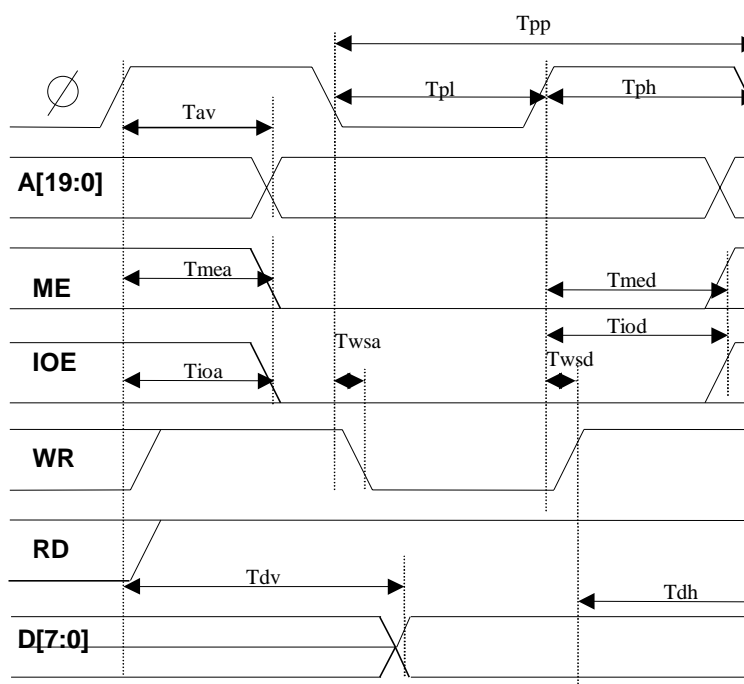


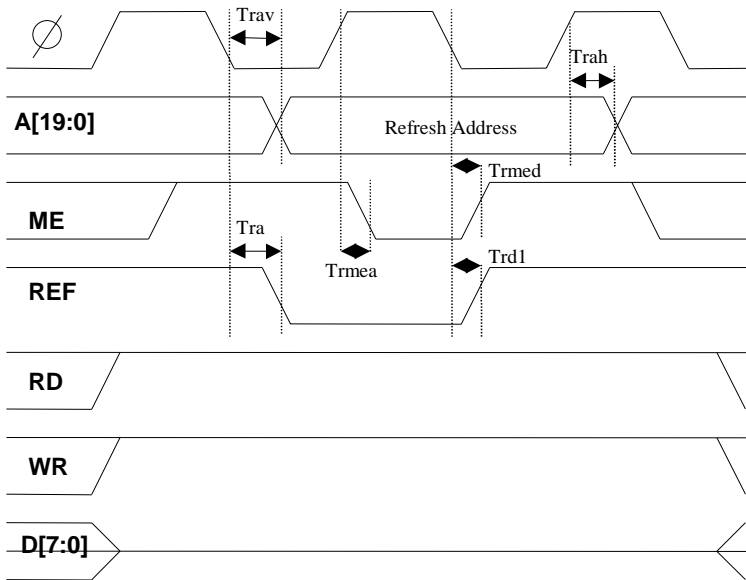
Figure 43

ME is asserted in memory space write cycles, IOE is asserted in I/O space writes.

# Refresh Cycle Timing

Two clock

Symbol	Meaning	Min	Nom	Max
Trav	PHI low to refresh address valid	2	-	8
Trah	Refresh address hold	3	-	8
Trmea	PHI high to ME asserted	2	-	6
Trmed	PHI low to ME deasserted	2	-	6
Tra	PHI low to REF asserted	2	-	6
Trd1	PHI low to REF deasserted (2 cycle refresh)	3	-	7
Trd2	PHI high to REF deasserted (3 cycle refresh)	3	-	7



Three clock

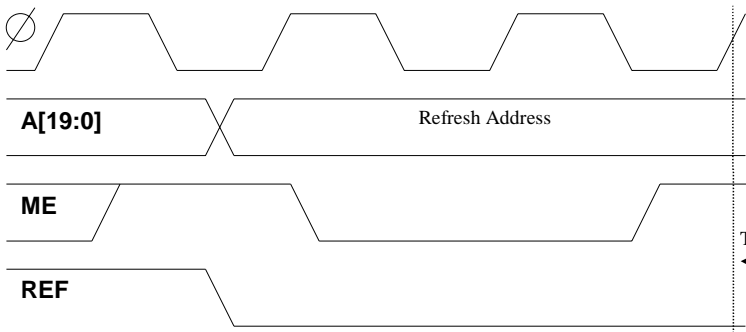


Figure 44

## Bus Grant Timing

Symbol	Meaning	Min	Nom	Max
Trba	PHI high to BUSACK asserted	2	-	5
Traz	PHI high to address tristate	2	-	6
Trcz	PHI high to control tristate	3	-	6
Trdz	PHI high to data tristate	3	-	10
Tsbd	PHI low to BUSACK deasserted	2	-	6
Tsad	PHI low to address drive	3	-	8
Tscd	PHI low to control drive	3	-	7

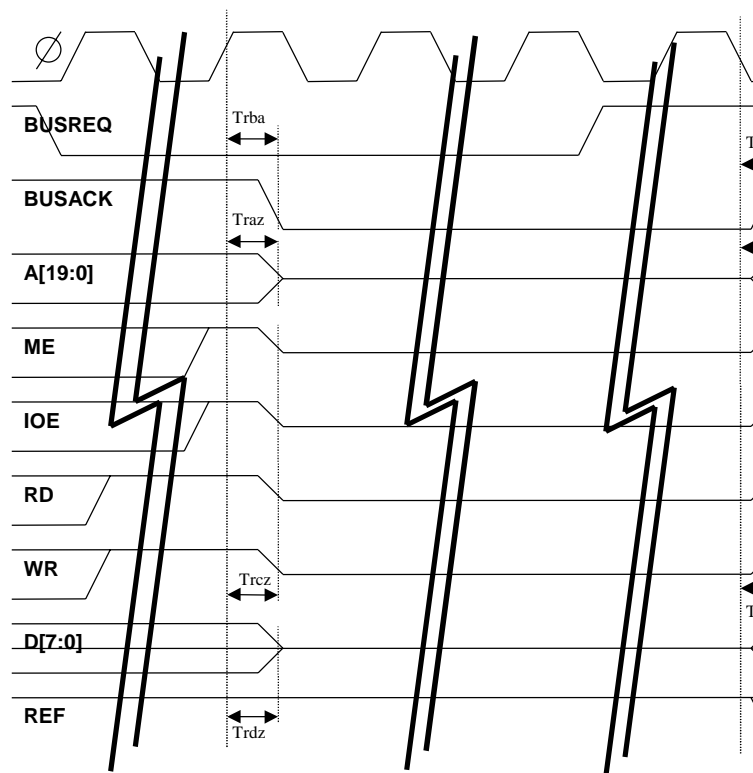


Figure 45

## Wait State Generation

Symbol	Meaning	Min	Nom	Max
Twsu	WAIT setup time	20	-	-
Twh	WAIT hold time	3	-	-

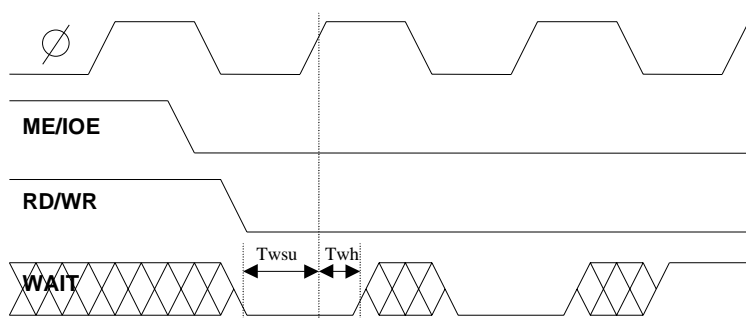


Figure 46

## Clock generation

The AB180-20 is provided with an on-chip crystal oscillator and phase locked loop (PLL) which provides frequency multiplication. An external clock can be directly input or the on-chip crystal oscillator can be utilised. The PLL multiplies the oscillator clock frequency by a factor of 8, up to a maximum internal operating frequency of 40MHz. An external clock source or crystal should be used which has a maximum frequency of 5MHz.

### Direct Clock Input

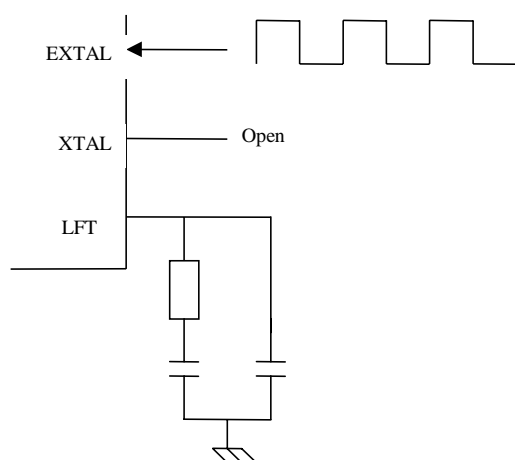


Figure 47

### Crystal Oscillator

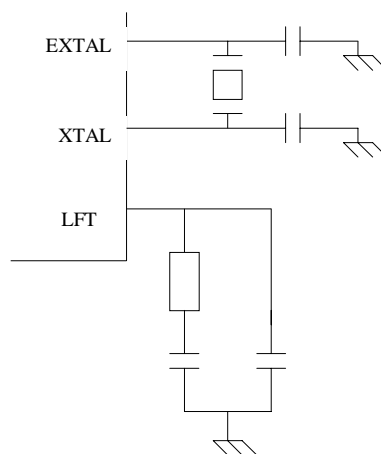


Figure 48



## Crystal Oscillator

The AB180-20 contains an oscillator circuit designed to be used with a 5 MHz AT cut crystal. To use this circuit two load capacitances of 15 pF are required.

When using a crystal it is important to ensure that the noise induced on the traces between the crystal and the package is kept to a minimum. This will require siting the crystal as physically and electrically close to the processor as possible. Trace lengths should be short and no power supplies (VCC) or other signals routed near them. Interference with the clock frequency by electrical noise will cause the PLL watchdog to reset the processor.

## PLL Filter Network

The on-chip PLL requires a filter network made up of three components as shown in Figure 49.

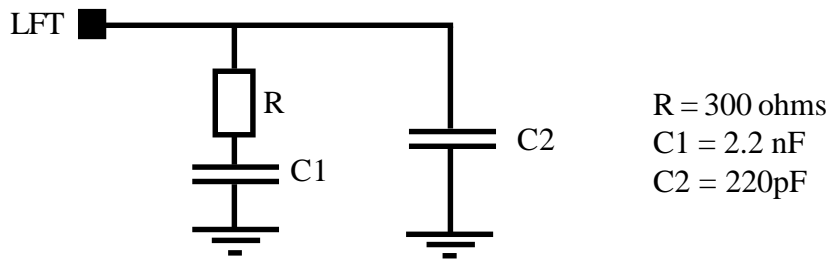


Figure 49

It is important that these components are close to the package pin, that traces are short and not near to other signals. Electrical noise on the LFT pin can cause spurious resets by the PLL watchdog.

## External Clock

An externally generated clock of 5 MHz can be applied to the EXTAL pin and with the XTAL pin left open the frequency of PHY will be four times the frequency fed into this pin.

Note: varying the frequency of this waveform will cause the PLL to lose lock and the PLL watchdog will reset the processor.

# Appendices

## OP-code Map - Instruction Set

The symbols in the instruction set are explained as follows:

### Register

g and g' specify an 8-bit register. ww, xx, yy, and zz specify a pair of 16-bit registers. The symbols and registers correspond as follows:

g, g'	Reg	ww	Reg	xx	Reg	yy	Reg	zz	Reg
000	B	00	BC	00	BC	00	BC	00	BC
001	C	01	DE	01	DE	01	DE	01	DE
010	D	10	HL	10	IX	10	IY	10	HL
011	E	11	SP	11	SP	11	SP	11	AF
100	H								
101	L								
111	A								

**Note:** suffixed H and L to ww, xx, yy, zz (eg. wwH, IXL) indicate upper and lower 8-bit of the 16-bit register respectively.

### Bit

b specifies a bit to be manipulated in the bit manipulation instruction. Bits and b correspond as follows:

b	000	001	010	011	100	101	110	111
Bit	0	1	2	3	4	5	6	7

### Condition

f specifies the condition in program control instructions. f and conditions correspond as follows:

f	000	001	010	011	100	101	110	111
Condition	NZ	Z	NC	C	PO	PE	P	M
	non zero	zero	non carry	carry	parity odd	parity even	sign plus	sign minus

### Restart Address

v specifies a restart address. v and restart addresses correspond as follows:

v	000	001	010	011	100	101	110	111
Address	00H	08H	10H	18H	20H	28H	30H	38H

### Flag

The flag conditions are:

.	not affected
x	undefined
S	set to 1
R	reset to 0
P	Parity
V	overflow

### Others

( )M	data in mem address
( )I	data in I/O address
m or n	8-bit data
mn	16-bit data
r	8-bit register
R	16-bit register

b.( )M	content of bit b in mem address
b.gr	" " in register gr
d or j	8-bit signed d'placement
S	source addressing
D	destination add mode
.	AND operation

# Appendices: OP-code Map

Table 1 1st Op-code Map Instruction Format :XX

NOTE 1) (HL) replaces g.  
2) (HL) replaces s.  
3) If DDH is supplemented as 1st op-code for the instructions which have HL or (HL) as an operand in Table 1, the instructions are executed replacing HL with IX and (HL) with (IX+d).

Example. 22H : LD (mn), HL  
DDH 22H : LD (mn), IX

If FDH is supplemented as 1st op-code for the instructions which have HL or (HL) as an operand in Table 1, the instructions are executed replacing HL with IY and (HL) with (IY+d).

Example 34H : INC (HL)  
FDH 34H : INC (IY+d)

However, JP (HL) and EX DE, HL are exception and note the followings.

If DDH is supplemented as 1st op-code for JP (HL), (IX) replaces (HL) as operand and JP (IX) is executed. If FDH is supplemented as 1st op-code for JP (HL), (IY) replaces (HL) as operand and JP (IY) is executed. Even if DDH or FDH is supplemented as 1st op-code for EX DE, HL, HL is not replaced and the instruction is regarded as illegal instruction.

When operating in One Cycle mode (0 wait states) all conditional relative jump instructions should be followed by a NOP (00h).

ww(LO=ALL)														LO=0~7													
BCDEHLSP														BCDEHLAFzz													
g (LO=0~7)														NZNCPOPf													
B D H (HL) B D H (HL)														00H 10H 20H 30H v													
HILO																											
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111																											
0 1 2 3 4 5 6 7 8 9 A B C D E F																											
S (HI=ALL)	B	0000	0	NOP	DJNZ j	JR NZ	JR NC	LD g, s	NOTE1)	ADD A, s	SUB s	AND s	OR s	RET f				0									
	C	0001	1	LD ww, mn										POP zz				1									
	D	0010	2	LD(ww), A		LD (mn), HL								LD (mn), A		JP f, mn				2							
	E	0011	3	INC ww										JP mn, A				OUT(m), HL		EX(SP), DI		3					
	H	0100	4	INC g										CALL f, mn								4					
	L	0101	5	DEC g										PUSH zz								5					
	(HL)	0110	6	LD g, m										NOTE1)				ADDA,m SUB m AND m OR m				6					
	A	0111	7	RLCA	RLA	DAA	SCF					RST v				7											
	B	1000	8	EXAF,AF	JR j	JR Z, j	JR C, j	LD g, s	NOTE2)	ADC A, s	SBC A, s	XOR s	CP s	RET f				8									
	C	1001	9	ADD HL, ww										RET				EXX	JP (HL), HL		LD SP, HL	9					
	D	1010	A	LD A, (WW)		LD HL, (mn)								LD A, (mn)		JP f, mn				A							
	E	1011	B	DEC ww										ADC A, s				SBC A, s		XOR s		CP s	Table2	INA, (m)	EXDE,HL	EI	E
	H	1100	C	INC g										CALL f, mn								C					
	L	1101	D	DEC g										CALL mn				NOTE3)		Table3		NOTE3)		D			
	(HL)	1110	E	LD g, m										NOTE2)				NOTE2)		NOTE2)		NOTE2)		ADCA,m SBCA,m XOR m CP m			
	A	1111	F	RRCA	RRA	CPL	CCF									RST v				F							
0 1 2 3 4 5 6 7 8 9 A B C D E F																											
C E L A C E L A																											
g(LO=8~F)																											
														LO=8~F													

4)

If FDH is supplemented as 1st op-code for the instructions which have (HL) as operand in Table 2, the instructions are executed relacing (HL) with (IY+d).

Table 3 2nd Op-code map Instruction Format : ED XX

					ww(LO=ALL)															
					BC	DE	HL	SP												
					g (LO=0~7)															
					B	D	H		B	D	H									
<div>HI</div>	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111				
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F				
LO																				
0000	0	INO g, (m)				IN g, (C)								LDI	LDIR					
0001	1	OUTO (m), g				OUT (C), g								CPI	CPIR					
0010	2					SBC HL, ww								INI	INIR					
0011	3					LD (mn), ww				OTIM	OTIMR	OUTI	OUTIR							
0100	4	TST g		TST (HL)	NEG		TST m	TSTIO m												
0101	5					RETN														
0110	6					NOP	NOP													
0111	7					LD I, A	LD A, I	RRD												
1000	8	INO g, (m)				IN g, (C)								LDD	LDDR					
1001	9	OUTO (m), g				OUT (C), g								CPD	CPDR					
1010	A					ADC HL, ww								IND	INDR					
1011	B					LD ww, (mn)				OTDM	OTDMR	OUTD	OTDR							
1100	C	TST g				MLT ww														
1101	D					RETI														
1110	E							NOP												
1111	F					LD R, A	LD A, R	RLD												
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
					C	E	L	A	C	E	L	A								
					g(LO=8~F)															

AB Semicon Limited  
AB Semicon House, 62 Victoria Road,  
Burgess Hill, West Sussex RH15 9LR United Kingdom  
Tel: +44(0) 1444 870408  
Fax: +44(0) 1444 870452  
Email: info@hbmuk.com

AB Semicon Inc.  
8305 Highway 71 West, Austin, Texas 78735 United States  
Tel: +1 512 288 6750  
Fax: +1 512 288 7676  
Email: info@hbmuk.com

**Distributed in Japan**

Rikei Corporation  
1-26-2, Nishi-Shinjuku, Shinjuku-ku, Tokyo 163-05 Japan  
Tel: +81-3-3345-2189  
Fax: +81-3-3344-3949

Visit the AB Semicon web-site at: <http://www.ab-semicon.com>