WD16 MICROCOMPUTER

(Using MCP 3-Chip Microprocessor Set)

PROGRAMMER'S REFERENCE MANUAL

WESTERN DIGITAL
C O R P O R A T I O N

DWM-00100-04
REV A

WD1600 MICROCOMPUTER

(Using MCP 3-Chip Microprocessor Set)

PROGRAMMER'S REFERENCE MANUAL

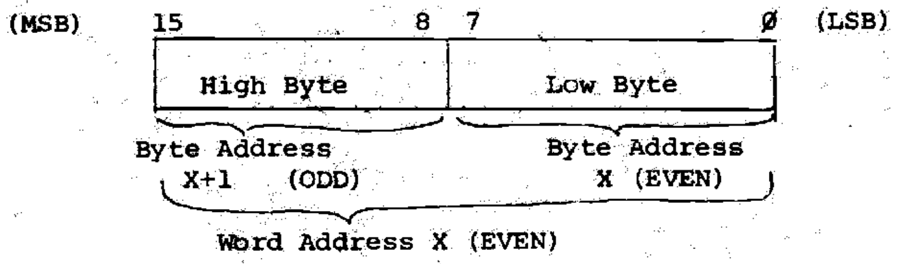
4 OCTOBER 1976

C-MD-12-80

TABLE OF CONTENTS

CHAPTER 1 - GENERAL

The WD16ØØ microcomputer is a 16 bit machine with both word and byte addressing, an automatic push down hardware stack, vectored interrupt handling, eight 16-bit registers, and PC relative addressing. A byte is defined as 8 bits, and a word is defined as 2 bytes. A memory address increment of one is an increment of 1 byte. An address increment of two is an increment of 1 word. Word addresses always start on even bytes. For any memory location the even byte is the least significant byte. Bit Ø is defined as the LSB of a memory location.



Unless otherwise stated, word addressing is implied. All addresses and op codes are done in hex unless otherwise stated. All hex numbers are enclosed within double quotes.

LEGEND OF ABBREVIATIONS

REG	= Register
SRC	= Source Address
(SRC)	= Contents of Source Address
DST	= Destination Address
(DST)	= Contents of Destination Address
$(SRC)_B$	= Contents of Source Byte Address
$(DST)_B$	= Contents of Destination Byte Address

$\overline{x}$	= Ones Complement of X
$-x$	= Twos Complement of X
Δ	= Logical And
∇	= Logical Or
$\underline{\nabla}$	= Exclusive or
@	= Indirect
$\downarrow$	= Push
$\uparrow$	= Pop
$\leftarrow$	= Destination Direction
+	= Addition
-	= Subtraction
*	= Multiplication
/	= Division
:	= Double Precision Chain Link

PROCESSOR STATUS WORD

A 16 bit Processor Status (PS) Word exists. The format is as follows:

15 - 8	7 - 4	3	2	1	Ø
Ext. Status Reg.	ALU	N	Z	V	C

Where bits 8-15 are the contents of the external status register (see chapter 2), bits 4-7 are the status of the microprocessor ALU flags, and bits Ø -3 are the status of the condition indicators at the time the PS is formed. The ALU flags are of no use or concern to the programmer. They are stored along with the condition indicators automatically as a function of the micro-op. The four condition flags are updated during the execution of most op codes, and are used by the branch instructions to test for valid branch conditions. The exact status of each indicator is defined along with the descriptions of individual op codes in chapter 3. In general, however, the indicators are set by the following conditions:

N = set if the MSB of the result is set.
Z = set if the result is zero.
V = set if arithmetic overflow (underflow) occurs during addition (subtraction)
    Set to exclusive-or of N and C indicators otherwise.
C= set if carry (borrow) occurs during addition (subtraction). Also set
    to last bit shifted out during a shift operation.

REGISTERS

There are 8 registers in the WD1600. All are 16 bits long. Six can be used as either accumulators or index registers, one is the stack pointer (SP), and one is the program counter (PC). The registers are numbered RØ - R7 with R6 = SP and R7 = PC. The register set is usually referred to in the following manner: RØ - R5, SP, PC.

CHAPTER TWO - INTRODUCTION

ADDRESSING MODES

In general there are 8 addressing modes for both source and destination addressing. Not all op codes accept all 8 modes (see chapter 3). Those that do use the following format: 3 bits for the index register (RØ - R5, SP, PC) and 3 bits for the mode. The mode bits are the upper 3 bits of the 6 bit set. The modes are defined below. The numbers in parenthesis refer to notes that follow the definitions.

MODE	NAME	SYMBOLIC	DESCRIPTION
Ø	Direct Register	REG	REG is or contains operand.
1	Indirect Register	@REG	REG contains address of operand.
2	Auto-increment	(REG)+	REG contains address of operand. REG is post-incremented (1).
3	Auto-increment deferred	@(REG)+	REG contains address of address of operand. REG is post-incremented by 2.
4	Auto-decrement	-(REG)	REG is predecremented (1). REG then contains address of operand.
5	Auto-decrement deferred	@-(REG)	REG is predecremented by 2. REG then contains address of address of operand.
6	Indexed register	X(REG)	Contents of REG plus X is address of operand (2).
7	Indexed register deferred	@X(REG)	Contents of REG plus X is address of address of operand (2).

NOTE 1: For word operations the increment/decrement is 2. For byte operations the increment/decrement is 1 unless the index register is SP or PC. In this case the increment/decrement is always 2.

NOTE 2: The contents of REG remain unchanged.

When using PC as the index register the assembler accepts the following 4 formats in place of the formats mentioned above for ease of programming.

MODE	NAME	SYMBOLIC	DESCRIPTION
2	Immediate	#N	Operand N follows op code.
3	Absolute	@#N	Address of operand is N and it follows the op code in memory.
6	Relative	A	PC relative offset to address A, which contains operand, follows op code.
7	Relative deferred	@A	PC relative offset to address A, which contains address of operand, follows the op code.

The 8 modes are referred to as Source Mode Ø to Source Mode 7 (SMØ -SM7) and Destination Mode Ø to Destination Mode 7 (DMØ -DM7). In Chapter 3 these modes are referred to in general terms during op code definitions as "SRC" and "DST".

STACK OPERATIONS

Although automatic stack operations are provided for, no specific area of memory is set aside for the stack. The user must assign an area of memory by loading the stack pointer with the top address of the designated stack area. Stack operations are push-down pop-up operations with predecrements and post-increments of SP. Stack operations may also be executed explicitly by using SP as an index register with op codes that allow SMØ - SM7 and/or DMØ - DM7 addressing.

When pushing the PS the word is formed just prior to the push. When popping the PS the condition indicators and interrupt enable flag are set to the status of the appropiate bits in the popped PS. Other than that the popped PS goes nowhere. Unless otherwise stated popping the PS from the stack performs the above mentioned operations and only the above mentioned operations.

When pushing the PC onto the stack PC will be set to the address of the op code that follows the op code that caused the push. There are cases where some op code formats can alter this rule. They generally involve advanced programming techniques. A few are mentioned in appendix C. In particular, system errors that are caused by programming errors and not real time error conditions will push a PC that points to the op code that follows the op code that caused the error. The stored PC must be decremented by two to get the address of the offending op code.

INTERRUPT LINES

There are 4 interrupt lines available to the system. They are labeled IØ - I3. These lines are assigned functions as follows:

IØ = Vectored interrupt line
I1 = Nonvectored interrupt line
I2 = Enable/disable for IØ and I1.
I3 = Halt switch

The priority among the lines is as follows:

I3, I1$\wedge$I2, IØ$\wedge$I2.

Note that I3 is always enabled. Note also that the nonvectored interrupt has priority over the vectored interrupt. The system is currently set up so that power fail and a real time clock can be assigned to I1, and up to 16 devices assigned to IØ.* The two interrupts operate as follows:

A) <u>Nonvectored Interrupt (I1)</u>
PS and PC are pushed onto the stack. I2 is disabled. The external status register is tested for a power fail. If power fail is true PC is fetched from location "14". If power fail is false PC is fetched from location "2A", and a microm state code is transmitted to clear the line clock (see appendix D).

B) <u>Vectored Interrupt (IØ)</u>
PS and PC are pushed onto the stack. I2 is disabled. An Interrupt Acknowledge is executed, and the device code of the interrupting device is read in and stripped to bits 1-4. PC is fetched from location

*NOTE: Although only a 4 bit device code is currently used, a minor microm change can allow a device code of from 1-15 bits.

"28" and the device code is added to it. The contents of this intermediate location are read in and added to PC to form the final address. Each intermediate location is a table entry that contains the PC relative offset from the start of the device handler routine to itself. The absolute address of the start of the table is in location "28".

PRIORITY MASK

Associated with the interrupts is a priority interrupt mask. This is a 16 bit mask where each bit position represents a priority level. Each priority level can be assigned to one or more devices. A one in any bit position can represent an interrupt enable or disable for its associated devices as the hardware dictates. The SAVS, RSTS, and MSKO op codes each alter the mask. When the mask is altered it is written into location "2E" for storage. While the mask is on the bus a microm state code is transmitted (see appendix D) to signal the I/O devices that a new mask is being transmitted. Each device can then look at its assigned mask bit while the memory write to location "2E" is taking place. Whether or not the mask feature is actually used by the I/O devices in no way alters the operations of the op codes mentioned above.

EXTERNAL STATUS REGISTER

As a part of the hardware external to the CPU the External Status Register supplies the CPU, upon demand, with information about the status of certain hardware areas. This register is gated onto the bus when its associated microm state code is present (see appendix D). The format of the register is as follows:

Bit 7 = Power Fail Status
Bit 6 = Bus Error (Time Out) Status
Bit 5 = Parity Error Status
Bit 4 = I2 Interrupt Line Status
Bit 3 = Halt Option Jumper #2
Bit 2 = Halt Option Jumper #1
Bit 1 = Power Up Option Jumper #2
Bit Ø = Power Up Option Jumper #1

Bits 8-15 are don't care. Bits 5-7 are real time error conditions that also generate a system reset (see next section). Bit 4 is the interrupt enable status. The jumpers can be logic units, switches, or hard wired jumpers as the user wishes. The various options associated with the 4 jumpers are discussed later.

POWER UP OPTIONS

A system reset indicate one of 4 conditions: power fail, bus error, parity error, or power up. There are 2 levels of power fail possible in this system (see appendix C): minor and major. Only a major power fail generates a system reset. Both types set bit 7 in the External Status Register. The following steps are performed after a system reset.

A1) Trace and wait flags are reset if on.
A2) The external Status Register is fetched.

A3) The Line-clock-clear state code is transmitted.
A4) I2 is reset.
A5) If power fail bit is set go to D1.
A6) If bus error bit is set go to C1.
A7) If parity error bit is set go to B1.
A8) Go to D2 otherwise.

B1) Push PS and PC onto stack.
B2) Fetch PC from location "12"and begin execution.

C1) Push PS and PC onto stack.
C2) Fetch PC from location "18" and begin execution.

D1) Wait until power fail status = Ø.
D2) Send a system reset microm state code.
D3) Wait 300 cycles.
D4) Execute power up option 1,2,3 or 4 per jumpers.

For a proper initial power up either bit 7 must be set or bits 5-7 must be reset when the system reset line is released.

The 4 power up options are as follows:

JUMPERS	OPERATION
ØØ	Execute user bootstrap routine.
Ø1	Pick up RØ-R5, SP, PC, and PS from memory locations Ø-"1Ø".
1Ø	Execute selected halt option.
11	Fetch PC from location "16".

HALT OPTIONS

When the halt switch (I3) is set during program execution one of 4 halt options is selected. The halt op code* and power up option #2 also select the halt option specified. The options are as follows:

JUMPERS	OPERATION
ØØ	Execute user bootstrap routine.
Ø1	Save RØ-R5,SP,PC and PS in memory locations Ø-"1Ø". Wait until I3 = Ø, then restore RØ-R5,SP,PC and PS from memory locations Ø-"1Ø".
1Ø	Lock up processor (requires a system reset to clear).
11	Fetch new PC from location "16".

*NOTE: Conditional. See Chapter 3.

USER BOOTSTRAP ROUTINE

When the user bootstrap routine is selected as an option the system creates the starting address by placing address "CØØØ" in PC and then replacing bits 8-13 with the contents of the 6 bit External Address Register. This register is gated in with a microm status code (see appendix D).

It allows the user 64 different starting addresses in the range "CØØØ" to "FFØØ".

SYSTEM ERROR TRAPS

With the exception of the major power fail error that is a function of a system reset, all error conditions perform a common routine as outlined below. A non-vectored interrupt and some op codes also use this routine. The numbers in parenthesis refer to notes that follow the table.

1) PS is pushed onto the stack
2) PC is pushed onto the stack
3) PC is fetched from location X where "X" is from the following table

(1)(2)(3)	"12"	for bus error PC
(1)(2)(3)	"14"	for nonvectored interrupt power fail PC
(1)(2)(3)	"18"	for parity error PC
(1)(2)(3)	"1A"	for reserved op code error PC
(1)(2)(3)	"1C"	for illegal op code format error PC
(1)(2)(3)	"1E"	for XCT error PC
(1)(2)	"2Ø"	for XCT trace PC
(1)(2)(3)	"2A"	for nonvectored interrupt PC
(1)(2)	"2C"	for BPT PC

NOTE 1: wait flag reset if on
NOTE 2: trace flag reset if on
NOTE 3: interrupt enable (I2) reset if on

The meaning of the wait and trace flags is discussed in chapter 3. Note that the nonvectored interrupt power fail PC is a minor power fail condition, not a major one. See appendix C for full detail on how to include both major and minor power fail conditions in the hardware.

RESERVED CORE LOCATIONS

The following is a complete list of memory locations that are reserved for specific system functions or options. Byte addresses are given.

LOCATIONS	RESERVED FUNCTION
Ø - "11"	RØ - R5, SP, PC and PS for power up/halt options
"12" - "13"	bus error PC
"14 - "15"	nonvectored interrupt power fail PC
"16" - "17"	power up/halt option power restore PC
"18" - "19"	parity error PC
"1A" -"1B"	reserved op code PC
"1C" - "1D"	illegal op code format PC
"1E" - "1F"	XCT error PC
"2Ø" - "21"	XCT trace PC
"22" - "23"	SVCA table address
"24" - "25"	SVCB PC
"26" - "27"	SVCC PC
"28"- "29"	vectored interrupt (IØ) table address
"2A" - "2B"	nonvectored interrupt (I1) PC
"2C" - "2D"	BPT PC
"2E" - 2F"	I/O priority interrupt mask
"3Ø" - "3F"	reserved for floating point option

CHAPTER 3 - OP CODES

This chapter is divided into a number of sections, each representing one class of op codes. At the beginning of each section there is a detailed description of the format for that class. A list of op codes and their base numeric values, less arguments, is also included. A detailed description of each op code in the class then follows.

FORMAT 1 OP CODES

Single word - no arguments

15 12	11 8	7 4	3 0
Ø	Ø	Ø	OPC

There are 16 op codes in this class representing op codes "ØØØØ" to "ØØØF". Each is a one word op code with no arguments with the exception of the SAVS op code which is a two word op code. Word two of the SAVS op code is the I/O priority interrupt mask. The op codes and their mnemonics are:

BASE OP CODE	MNEMONIC
ØØØØ	NOP
ØØØ1	RESET
ØØØ2	IEN
ØØØ3	IDS
ØØØ4	HALT
ØØØ5	XCT
ØØØ6	BPT
ØØØ7	WFI
ØØØ8	RSVC
ØØØ9	RRTT
ØØØA	SAVE
ØØØB	SAVS
ØØØC	REST
ØØØD	RRTN
ØØØE	RSTS
ØØØF	RTT

NOP NO OPERATION

FORMAT:	NOP
FUNCTION:	No operations are performed
INDICATORS:	Unchanged

RESET I/O RESET

FORMAT:	RESET
FUNCTION:	An I/O reset pulse is transmitted
INDICATORS:	Unchanged

IEN — INTERRUPT ENABLE

FORMAT: IEN

FUNCTION: The interrupt enable (I2) flag is set. Allows one more instruction to execute before interrupts are recognized.*

INDICATORS: Unchanged

IDS — INTERRUPT DISABLE

FORMAT: IDS

FUNCTION: The interrupt enable (I2) flag is reset. This instruction can honor interrupts, but the I2 bit in the PS that is stored on the stack is reset if an interrupt occurs.*

INDICATORS: Unchanged

*NOTE: On some machines I2 will be set or reset during the IEN or IDS . If so the change will be valid immediately, not one op code later.

HALT — HALT

FORMAT: HALT

FUNCTION: Tests the status of the Power Fail bit in the external status register. If the bit is set it is assumed that the HALT occured in a power fail routine, and the following operations occur:

1) The interrupt enable (I2) flag is reset
2) The CPU waits until the Power Fail bit is reset
3) PC is fetched from location "16", and program execution begins at this new location

If the power fail bit is reset then the CPU waits until the halt switch (I3) is set. At that time the selected halt option (see chapter 2) is executed. The interrupt enable flag is also reset.

INDICATORS: Unchanged

XCT — EXECUTE SINGLE INSTRUCTION

FORMAT: XCT

OPERATION:

```
PC ← @SP, SP ↑
PS ← @SP, SP ↑
Trace flag set,execute op code
↓SP, @SP ← PS
↓SP, @SP ← PC
Trace flag reset
PC  ←(loc "2Ø") if no error
PC  ←(loc "1E") if error
```

FUNCTION: PC and PS are popped from the stack, but I2 is not altered. The trace flag, which disables all inter- rupts except I3, is set. The op code is executed PS and PC are pushed back onto the stack, and PC is fetched from location "2Ø". The trace flag is reset. If the program tries to execute a HALT , XCT, BPT, or WFI the attempt is aborted, PS and PC are

pushed onto the stack, and PC is fetched from location "1E" instead.
I2 is also reset.

INDICATORS:	Depends upon executed op code

BPT — BREAKPOINT TRAP

FORMAT:	BPT
OPERATION:	$\downarrow$SP, @SP $\leftarrow$ PS $\downarrow$SP, @SP $\leftarrow$ PC PC $\leftarrow$ (loc "2C")
FUNCTION:	PS and PC are pushed onto the stack. PC is fetched from location "2C"
INDICATORS:	Unchanged

WFI — WAIT FOR INTERRUPT

FORMAT:	WFI
FUNCTION:	The CPU loops internally without accessing the data bus until an interrupt occurs. Program execution continues with the op code that follows the WFI after the interrupt has been serviced. The interrupt enable flag is also set.
INDICATORS:	Unchanged

SAVE — SAVE REGISTERS

FORMAT:	SAVE
OPERATION:	$\downarrow$SP, @SP $\leftarrow$ R5 $\downarrow$SP, @SP $\leftarrow$ R4 $\downarrow$SP, @SP $\leftarrow$ R3 $\downarrow$SP, @SP $\leftarrow$ R2 $\downarrow$SP, @SP $\leftarrow$ R1 $\downarrow$SP, @SP $\leftarrow$ RØ
FUNCTION:	Registers R5 to RØ are pushed onto the stack.
INDICATORS:	Unchanged.

SAVS — SAVE STATUS

FORMAT:	SAVS MASK
OPERATION:	SAVE $\downarrow$SP, @SP $\leftarrow$ (loc "2E") (loc "2E") $\leftarrow$ (loc "2E") ∇ mask MSKO IEN
FORMAT:	Registers R5 to RØ and the priority mask in location "2E" are pushed onto the stack. The old and new masks are ORED together and placed in location "2E". A mask out state code (see appendix D) is transmitted and the interrupt enable (I2) flag is set.
INDICATORS:	Unchanged

REST — RESTORE REGISTERS

FORMAT:	REST
OPERATION:	RØ $\leftarrow$ @SP, SP $\uparrow$ R1 $\leftarrow$ @SP, SP $\uparrow$ R2 $\leftarrow$ @SP, SP $\uparrow$

	R3 ← @SP, SP ↑
	R4 ← @SP, SP ↑
	R5 ← @SP, SP ↑

FUNCTION: Registers RØ to R5 are popped from the stack.
INDICATORS: Unchanged

RTT — RETURN FROM TRAP

FORMAT:	RTT
OPERATION:	PC ← @SP, SP ↑
	PS ← @SP, SP ↑
FUNCTION:	PC and PS are popped from stack
INDICATORS:	N = Set per PS bit 3
	Z = Set per PS bit 2
	V = Set per PS bit 1
	C = Set per PS bit Ø

RRTN — RESTORE AND RETURN FROM SUBROUTINE

FORMAT:	RRTN
OPERATION:	REST
	PC ← @SP, SP↑
FUNCTION:	Registers RØ to R5 and PC are popped from the stack
INDICATORS:	Unchanged

RRTT — RESTORE AND RETURN FROM TRAP

FORMAT:	RRTT
OPERATION:	REST
	RTT
FUNCTION:	Registers RØ to R5, PC and PS are popped from the stack.
INDICATORS:	Set per PS bits Ø - 3

RSTS — RESTORE STATUS

FORMAT:	RSTS
OPERATION:	(LOC "2E") ← @SP, SP ↑
	MSKO
	REST
	RTT
FUNCTION:	The priority mask is popped from the stack and restored to locaton "2E". A MASK OUT state code (See Appendix D) is transmitted. Registers RØ to R5, PC and PS are popped from the stack.
INDICATORS:	Set per PS bits Ø - 3

RSVC — RETURN FROM SUPERVISOR CALL (B or C)

FORMAT:	RSVC
OPERATION:	REST
	SP↑
	RTT

FUNCTION:	Registers RØ to R5, PC and PS are popped from the stack with the saved SP bypassed.
INDICATORS:	Set per PS bits Ø - 3

FORMAT 2 OP CODES

SINGLE WORD - 3 BIT REGISTER ARGUMENT

15 12	11 8	7 3	2 0
Ø	Ø	OPC	REG

There are 4 op codes in this class representing op codes "ØØ1Ø" to "ØØ2F". Each is a one word op code with a single 3 - bit register argument. The op codes and their mnemonics are:

BASE OP CODE	MNEMONIC
ØØ1Ø	IAK
ØØ18	RTN
ØØ2Ø	MSKO
ØØ28	PRTN

IAK	INTERRUPT ACKNOWLEDGE
FORMAT:	IAK REG
FUNCTION:	An interrupt acknowledge (READ and IACK) is executed, and the 16 bit code that is returned is placed in REG unmodified. Used with the nonvectored interrupt when the user does not wish to use the vectored format.
INDICATORS:	Unchanged

RTN	RETURN FROM SUBROUTINE
FORMAT:	RTN REG

OPERATION:	PC ← REG
	REG ← @SP,SP ↑

FUNCTION:	The linkage register is placed in PC and the saved linkage register is popped from the stack. The register used must be the same one that was used for the subroutine call.
INDICATORS:	Unchanged

MSKO	MASK OUT
FORMAT:	MSKO REG
OPERATION:	(LOC "2E") ← REG MSKO
FUNCTION:	The contents of REG are written into location "2E" and a MASK OUT state code (see appendix D) is transmitted.
INDICATORS:	Unchanged

PRTN	POP STACK AND RETURN
FORMAT:	PRTN REG

OPERATION:	TMP ← @SP
	SP ← SP+(TMP*2)
	RTN REG

FUNCTION:	Twice the value of the top word on the stack is added to SP, and a standard RTN call is then executed.
INDICATORS:	Unchanged

FORMAT 3 OP CODES

SINGLE WORD - 4 BIT NUMERIC ARGUMENT

15 — 12	11 — 8	7 — 4	3 — Ø
Ø	Ø	OPC	ARG

There is only one op code in this class representing op codes "ØØ3Ø" to "ØØ3F". It is a one word op code with a 4-bit numeric argument.

BASE OP CODE	MNEMONIC
ØØ3Ø	LCC

LCC	LOAD CONDITION CODES
FORMAT:	LCC ARG
FUNCTION:	The 4 indicators are loaded from bits Ø-3 of the op code as specified.
INDICATORS:	N = Set per bit 3 of op code
	Z = Set per bit 2 of op code
	V = Set per bit 1 of op code
	C = Set per bit Ø of op code

FORMAT 4 OP CODES

SINGLE WORD - 6 BIT NUMERIC ARGUMENT

15 12	11 8	7 6	5 Ø
Ø	Ø	OPC	ARG

There are 3 op codes in this class representing op codes "ØØ4Ø" to "ØØFF". All 3 are supervisor calls . All 3 are one word op codes with a 6-bit numeric argument.

BASE OP CODE	MNEMONIC
ØØ4Ø	SVCA
ØØ8Ø	SVCB
ØØCØ	SVCC

SVCA	SUPERVISOR CALL "A"
FORMAT:	SVCA ARG

OPERATION:	↓SP, @SP ← PS; ↓ SP, @SP ← PC
	PC ← (LOC "22") + (ARG *2)
	PC ← PC + @PC

FUNCTION:	PS and PC are pushed onto the stack. The contents of location "22" plus twice the value of the argument (which is always positive) is placed in PC to get the table address. The contents of the table address is added to PC to get the final destination address. Each table entry is the relative offset from the start of the desired routine to itself.
INDICATORS:	Unchanged

SVCB	SUPERVISOR CALL "B"
SVCC	SUPERVISOR CALL "C"
FORMAT:	SVCB ARG SVCC ARG

OPERATION:	TMPA ← SP
	↓SP, @SP ← PS
	↓SP, @SP ← PC
	TMPB ← SP
	↓SP, @SP ← TMPA
	SAVE
	R1 ← TMPB
	R5 ← ARG*2
	PC ← (LOC "24") if SVCB
	PC ← (LOC "26") if SVCC

FUNCTION:	PS and PC are pushed onto the stack. The value of SP at the start of op code execution is the pushed followed by registers R5 to RØ. The address of the saved PC is placed in R1, and twice the value of the 6-bit positive argument is placed in R5.

PC is loaded from location "24" for SVCB or "26" for SVCC.

INDICATORS: Unchanged.

FORMAT 5 OP CODES

SINGLE WORD - 8 BIT SIGNED NUMERIC ARGUMENT

15 8	7 Ø
OPC	DISPLACEMENT

There are 15 op codes in this class representing op codes "Ø1ØØ" to "Ø7FF" and "8ØØØ" to "87FF". All are branches with a signed 8 bit displacement that represents the word offset from PC (which points to the op code that follows) to the desired branch location. The op codes consist on one unconditional branch, 8 signed conditional branches, and 6 unsigned conditional branches. No op code in this class modifies any of the indicator flags. Maximum branch range is +128, -127 words from the branch op code.

BASE OP CODE	MNEMONIC
Ø1ØØ	BR
Ø2ØØ	BNE
Ø3ØØ	BEQ
Ø4ØØ	BGE
Ø5ØØ	BLT
Ø6ØØ	BGT
Ø7ØØ	BLE
8ØØØ	BPL
81ØØ	BMI
82ØØ	BHI
83ØØ	BLOS
84ØØ	BVC
85ØØ	BVS
86ØØ	BCC, BHIS
87ØØ	BCS, BLO

BR — BRANCH UNCONDITIONALLY

FORMAT:	BR DEST
OPERATION:	PC ← PC+ (DISP *2)
FUNCTION:	Twice the value of the signed displacement is added to PC.

SIGNED BRANCHES

BNE — BRANCH IF NOT EQUAL TO ZERO

FORMAT:	BNE DEST
OPERATION:	IF Z = Ø, PC ← PC + (DISP *2)

BEQ — BRANCH IF EQUAL TO ZERO

FORMAT:	BEQ DEST
OPERATION:	IF Z = 1, PC ← PC + (DISP *2)

BGE — BRANCH IF GREATER THAN OR EQUAL TO ZERO

FORMAT:	BGE DEST
OPERATION:	IF $N \veebar V$ = Ø, PC ← PC + (DISP *2)

BLT	**BRANCH IF LESS THAN ZERO**
FORMAT:	BLT DEST
OPERATION:	IF $N \veebar V$ = 1, PC ← PC + (DISP *2)
BGT	**BRANCH IF GREATER THAN ZERO**
FORMAT:	BGT DEST
OPERATION:	IF $Z \vee (N \veebar V)$ = Ø, PC ← PC + (DISP *2)
BLE	**BRANCH IF LESS THAN OR EQUAL TO ZERO**
FORMAT:	BLE DEST
OPERATION:	IF $Z \vee (N \veebar V)$ = 1, PC ← PC + (DISP *2)
BPL	**BRANCH IF PLUS**
FORMAT:	BPL DEST
OPERATION:	IF N = Ø, PC ← PC + (DISP *2)
BMI	**BRANCH IF MINUS**
FORMAT:	BMI DEST
OPERATION:	IF N = 1, PC ← PC + (DISP *2)
	UNSIGNED BRANCHES
BHI	**BRANCH IF HIGHER**
FORMAT:	BHI DEST
OPERATION:	IF $C \vee Z$ = Ø, PC ← PC + (DISP *2)
BLOS	**BRANCH IF LOWER OR SAME**
FORMAT:	BLOS DEST
OPERATION:	IF $C \vee Z$ = 1, PC ← PC + (DISP *2)
BVC	**BRANCH IF OVERFLOW CLEAR**
FORMAT:	BVC DEST
OPERATION:	IF V = Ø, PC ← PC + (DISP *2)
BVS	**BRANCH IF OVERFLOW SET**
FORMAT:	BVS DEST
OPERATION:	IF V = 1, PC ← PC + (DISP *2)
BCC	**BRANCH IF CARRY CLEAR**
BHIS	**BRANCH IF HIGHER OR SAME**
FORMAT:	BCC DEST BHIS DEST
OPERATION:	IF C = Ø, PC ← PC + (DISP *2)

BCS	**BRANCH IF CARRY SET**
BLO	**BRANCH IF LOWER**
FORMAT:	**BCS DEST** **BLO DEST**
OPERATION:	**IF C = 1, PC ← PC + (DISP *2)**

FORMAT 6 OP CODES

SINGLE WORD - SINGLE OPS - SPLIT FIELD - DMØ ONLY

15 9	8 6	5 4	3 0
OPC BASE	REG	OPC	COUNT

There are 12 op codes in this class representing op codes "Ø8ØØ" to "Ø9FF", "88ØØ" to "89FF", and "8EØØ" to "8FFF". There are 4 immediate mode op codes with a register as a destination, 4 multiple count single register shifts, and 4 multiple count double register shifts. In all op codes the actual count (or number in the case of the immediates) is the value of bits Ø - 3 plus one. Count is always a positive number in the range 1 - "1Ø", but it is stored in the op code as Ø - "F". All of these op codes are one word op codes with the op codes themselves split between bits 9-15 and 4-5.

In the case of the double shifts the 32 bit number (REG+1) : (REG) is the operand. If REG = PC then (REG+1) = RØ.

BASE OP CODE	MNEMONIC
Ø8ØØ	ADDI
Ø81Ø	SUBI
Ø82Ø	BICI
Ø83Ø	MOVI
88ØØ	SSRR
881Ø	SSLR
882Ø	SSRA
883Ø	SSLA
8EØØ	SDRR
8E1Ø	SDLR
8E2Ø	SDRA
8E3Ø	SDLA

ADDI	ADD IMMEDIATE
FORMAT;	ADDI NUMBER, REG
OPERATION:	REG ← REG + COUNT + 1
FUNCTION:	The stored number plus one is added to the destination register.
INDICATORS:	N = Set if bit 15 of the result is set
	Z = Set if the result = Ø
	V = Set if arithmetic overflow occurs; i.e. set if both operands were positive and the sign of the result is negative
	C = Set if a carry was generated from bit 15 of the result

SUBI	SUBTRACT IMMEDIATE
FORMAT:	SUBI NUMBER, REG
OPERATION:	REG ← REG - (COUNT +1)
FUNCTION:	The stored number plus one is subtracted from the destination register.

INDICATORS:	N = Set if bit 15 of the result is set Z = Set if the result = Ø V = Set if arithmetic underflow occurs; i.e. set if the operands were of opposite signs and the sign of the result is positive C = Set if a borrow was generate from bit 15 of the result

BICI — BIT CLEAR IMMEDIATE

FORMAT:	BICI NUMBER, REG
OPERATION:	REG ← REG Δ(COUNT + 1)
FUNCTION:	The stored number plus one is one's complemented and ANDED to the destination register

INDICATORS:	N = Set if bit 15 of the result is set
	Z = Set if the result = Ø
	V = Reset
	C = Unchanged

MOVI — MOVE IMMEDIATE

FORMAT:	MOVI NUMBER, REG
OPERATION:	REG ← COUNT + 1
FUNCTION:	The stored number plus one is placed in the destination register

INDICATORS:	N = Reset
	Z = Reset
	V = Reset
	C = Unchanged

SSRR — SHIFT SINGLE RIGHT ROTATE

FORMAT:	SSRR REG, COUNT
FUNCTION:	A 17-bit right rotate is done stored count+1 times on REG:C-Flag. The C-Flag is shifted into bit 15 of REG, and the C-Flag gets the last bit shifted out of REG bit Ø.

INDICATORS:	N = Set if bit 7 of REG is set
	Z = Set if REG = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the last bit shifted out of REG bit Ø

SSLR — SHIFT SINGLE LEFT ROUTINE

FORMAT:	SSLR REG, COUNT
FUNCTION:	A 17-bit left rotate is done stored count+1 times on C-Flag:REG . The C-Flag is shifted into bit Ø of REG and the C-Flag gets the last bit shifted out of REG bit 15.

INDICATORS:	N = Set if bit 15 of REG is set
	Z = Set if REG = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the last bit shifted out of REG bit 15.

SSRA — SHIFT SINGLE RIGHT ARITHMETIC

FORMAT: SSRA REG, COUNT

FUNCTION: A 17-bit right arithmetic shift is done stored count+1 times on REG:C-Flag. Bit 15 of REG is replicated. The C-Flag gets the last bit shifted out of REG bit Ø. Bits shifted out of the C-Flag are lost.

INDICATORS:
N = Set if bit 7 of REG is set
Z = Set if REG = Ø
V = Set to exclusive or of N and C flags
C = Set to the value of the last bit shifted out of REG bit Ø

SSLA — SHIFT SINGLE LEFT ARITHMETIC

FORMAT: SSLA REG, COUNT

FUNCTION: A 17-bit left arithmetic shift is done stored count+1 times on C-Flag:REG. Zeros are shifted into REG bit Ø, and the C-FLAG gets the last bit shifted out of REG bit 15. Bits shifted out of the C-Flag are lost.

INDICATORS:
N = Set if REG bit 15 is set
Z = Set if REG = Ø
V = Set to exclusive or of N and C flags
C = Set to the value of the last bit shifted out of REG bit 15

SDRR — SHIFT DOUBLE RIGHT ROTATE

FORMAT: SDRR REG, COUNT

FUNCTION: REG+1:REG:C-Flag is rotate right stored count+1 times. The C-Flag is shifted into REG+1 bit 15, REG+1 bit Ø is shifted into REG bit 15, and REG bit Ø is shifted into the C-Flag.

INDICATORS:
N = Set if bit 7 of REG is set
Z = Set if REG = Ø
V = Set to exclusive or of N and C flags
C = Set to the value of the last bit shifted out of REG bit Ø

SDLR — SHIFT DOUBLE LEFT ROTATE

FORMAT: SDLR REG, COUNT

FUNCTION: A 33 bit left rotate is done stored count+1 times on C-Flag:REG+1:REG. The C-Flag is shifted into REG bit Ø, REG bit 15 is shifted into REG+1 bit Ø, and REG+1 bit 15 is shifted into the C-Flag

INDICATORS:
N = Set if REG+1 bit 15 is set
Z = Set if REG+1 = Ø
V = Set to exclusive or of N and C flags
C = Set to the value of the last bit shifted out of REG+1 bit 15.

SDRA — SHIFT DOUBLE RIGHT ARITHMETIC

FORMAT:	SDRA REG, COUNT
FUNCTION:	A right arithmetic shift is done stored count+1 times on REG+1:REG:C-Flag, Bit 15 of REG+1 is replicated. Bit Ø of REG+1 is shifted to bit 15 of REG. Bit Ø of REG is shifted to the C-Flag. Bits shifted out of the C-Flag are lost.

INDICATORS:	N = Set if bit 7 of REG is set
	Z = Set if REG = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the last bit shifted out of REG bit Ø

SDLA — SHIFT DOUBLE LEFT ARITHMETIC

FORMAT:	SDLA REG, COUNT
FUNCTION:	A left arithmetic shift is done stored count+1 times on C-Flag:REG+1:REG. Zeros are shifted into REG bit Ø, REG bit 15 is shifted to REG+1 bit Ø. REG+1 bit 15 is shifted to the C-Flag. Bits shifted out of the C-Flag are lost.

INDICATORS:	N = Set if REG+1 bit 15 is set
	Z = Set if REG+1 = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the last bit shifted out of REG+1 bit 15

FORMAT 7 OP CODES

SINGLE OPS - ONE OR TWO WORDS - DMØ TO DM7

```
 15          6  5       3  2     0
|    OPC       |   MODE   |  REG  |
```

There are 32 op codes in this class representing op codes "ØAØØ" to "ØDFF" and "8AØØ" to "8DFF". All addressing modes from Ø to 7 are available with all registers available as index registers (see chapter two). A one word op code is generated for addressing modes Ø to 5. A two word op code is generated for addressing modes 6 and 7 with the offset value in word two. For DM6 and DM7 with PC as the index register PC is added to the offset from word two after the offset is fetched from memory. The offset is therefore relative to a PC that points to the op code that follows (i.e. current op code + 4). Codes "8AØØ" to "8CCØ" are BYTE ops.

BASE OP CODE	MNEMONIC	BASE OP CODE	MNEMONIC
ØAØØ	ROR	8AØØ	RORB
ØA4Ø	ROL	8A4Ø	ROLB
ØA8Ø	TST	8A8Ø	TSTB
ØACØ	ASL	8ACØ	ASLB
ØBØØ	SET	8BØØ	SETB
ØB4Ø	CLR	8B4Ø	CLRB
ØB8Ø	ASR	8B8Ø	ASRB
ØBCØ	SWAB	8BCØ	SWAD
ØCØØ	COM	8CØØ	COMB
ØC4Ø	NEG	8C4Ø	NEGB
ØC8Ø	INC	8C8Ø	INCB
ØCCØ	DEC	8CCØ	DECB
ØDØØ	IW2	8DØØ	LSTS
ØD4Ø	SXT	8D4Ø	SSTS
ØD8Ø	TCALL	8D8Ø	ADC
ØDCØ	TJMP	8DCØ	SBC

WORD OPS

ROR ROTATE RIGHT

FORMAT:	ROR DST
FUNCTION:	A 1-bit right rotate is done on (DST):C-Flag The C-Flag is shifted into (DST) bit 15, and (DST) bit Ø is shifted into the C-flag.

INDICATORS:	N = Set if bit 7 of (DST) is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of (DST)

ROL ROTATE LEFT

FORMAT:	ROL DST
FUNCTION:	A 1-bit left rotate is done on C-Flag:(DST). The

	C-Flag is shifted into (DST) bit Ø, and (DST) bit 15 is shifted into the C-Flag.
INDICATORS:	N = Set if bit 15 of (DST) is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of (DST)

TST TEST WORD

FORMAT:	TST DST
OPERATION:	(DST) ∆ (DST)
FUNCTION:	The indicators are set to reflect the destination operand status.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

ASL ARITHMETIC SHIFT LEFT

FORMAT:	ASL DST
FUNCTION:	A 1-bit left arithmetic shift is done on (DST). A zero is shifted into (DST) bit Ø, and (DST) bit 15 is shifted into the C-Flag.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of (DST)

SET SET TO ONES

FORMAT:	SET DST
OPERATION:	(DST) ← "FFFF"
FUNCTION:	The destination operand is set to all ones
INDICATORS:	N = Set
	Z = Reset
	V = Reset
	C = Unchanged

CLR CLEAR TO ZEROS

FORMAT:	CLR DST
OPERATION:	(DST) ← Ø
FUNCTION:	The destination operand is cleared to all zeros
INDICATORS:	N = Reset
	Z = Set
	V = Reset
	C = Unchanged if DMØ. Reset if DM1-DM7.

ASR ARITHMETIC SHIFT RIGHT

FORMAT:	ASR DST
FUNCTION:	A 1-bit right arithmetic shift is done on (DST). Bit 15 of (DST) is replicated. Bit Ø of (DST) is shifted into the C-Flag.

INDICATORS:	N = Set if (DST) bit 7 is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of (DST)

SWAB **SWAP BYTES**

FORMAT:	SWAB DST
OPERATION:	(DST) 15-8 ⇄ (DST) 7-Ø
FUNCTION:	The upper and lower bytes of (DST) are exhanged.
INDICATORS:	N = Set if (DST) bit 7 is set
	Z = Set if (DST) lower byte = Ø
	V = Reset
	C = Unchanged

COM **COMPLEMENT**

FORMAT:	COM DST
OPERATION	$(DST) \leftarrow \overline{(DST)}$
FUNCTION:	The destination operand is one's complemented.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Set

NEG **NEGATE**

FORMAT:	NEG DST
OPERATION:	(DST) ← -(DST)
FUNCTION:	The destination operand is two's complemented.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if (DST) = "8ØØØ"
	C = Reset if (DST) = Ø

INC **INCREMENT**

FORMAT:	INC DST
OPERATION:	(DST) ← (DST) + 1
FUNCTION:	The destination operand is incremented by one.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if (DST) = "8ØØØ"
	C = Set if a carry is generated from (DST) bit 15

DEC **DECREMENT**

FORMAT:	DEC DST
OPERATION:	(DST) ← (DST) - 1
FUNCTION:	The destination operand is decremented by one.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if (DST) = "7FFF"
	C = Set if a borrow is generated from (DST) bit 15

IW2 INCREMENT WORD BY TWO

FORMAT:	IW2 DST
OPERATION:	(DST) ← (DST) + 2
FUNCTION:	The destination operand is incremented by two.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if (DST) = "8ØØØ" or "8ØØ1"
	C = Set if a carry is generated from (DST) bit 15

SXT SIGN EXTEND

FORMAT:	SXT DST
OPERATION:	IF N = Ø, (DST) ← Ø
	IF N = 1, (DST) ← "FFFF"
FUNCTION:	The N-Flag status is replicated in the destination operand
INDICATORS:	Unchanged

TCALL TABLED SUBROUTINE CALL

FORMAT:	TCALL DST
OPERATION:	↓SP, @SP ← PC
	PC ← PC + (DST)
	PC ← PC + @PC
FUNCTION:	PC, which points to the op code that follows, is pushed onto the stack. The destination operand is added to PC. The contents of this intermediate table address is also added to PC to get the final destination address. Note that at least one op code must exist between the TCALL and the table for a subroutine return.
INDICATORS:	Unchanged

TJMP TABLED JUMP

FORMAT:	TJMP DST
OPERATION:	PC ← PC + (DST)
	PC ← PC + @PC
FUNCTION:	The destination operand is added to PC, and the contents of this intermediate location is also added to PC to get the final destination address.
INDICATORS:	Unchanged

LSTS LOAD PROCESSOR STATUS

FORMAT:	LSTS DST
FUNCTION:	The four indicators and the interrupt enable (I2) are loaded from the destination operand.
INDICATORS:	Set to the status of (DST) bits Ø - 3

SSTS STORE PROCESSOR STATUS

FORMAT:	SSTS DST
FUNCTION:	The processor status word is formed and stored in (DST).
INDICATORS:	Unchanged

ADC ADD CARRY

FORMAT:	ADC DST
OPERATION:	(DST) ← (DST) + C-flag
FUNCTION:	The carry flag is added to the destination operand
INDICATORS:	N= Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set if a carry is generated from (DST) bit 15

SBC SUBTRACT CARRY

FORMAT:	SBC DST
OPERATION:	(DST) ← (DST) - C-Flag
FUNCTION:	The Carry flag is subtracted from the destination operand
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set to exclusive or of N and C flags
	C = Set if a borrow is generated from (DST) bit 15

BYTE OPS

For DMØ addressing only the lower byte of the destination register is affected by a byte op code. For DM1-DM7 addressing only the specified memory byte is affected by a byte op. For even memory addresses the lower byte is altered, and for ddd memory addresses the upper byte is altered.

RORB ROTATE RIGHT BYTE

FORMAT:	RORB DST
FUNCTION:	A 1-bit right rotate is done on $(DST)_B$:C-Flag. Bit Ø of $(DST)_B$ is shifted into the C-Flag, and the C-Flag is shifted into $(DST)_B$ bit 7.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of $(DST)_B$ bit Ø

ROLB ROTATE LEFT BYTE

FORMAT:	ROLB DST
FUNCTION:	A 1-bit left rotate is done on C-flag :$(DST)_B$. Bit 7 of $(DST)_B$ is shifted into the C-flag, and the C-flag is shifted into $(DST)_B$ bit Ø
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of $(DST)_B$ bit 7

TSTB TEST BYTE

FORMAT:	TSTB DST
OPERATION:	$(DST)_B \Delta (DST)_B$

FUNCTION:	The destination operand status sets the indicators.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Reset
	C = Unchanged

ASLB ARITHMETIC SHIFT LEFT BYTE

FORMAT:	ASLB DST
FUNCTION:	A 1-bit left arithmetic shift is done on C-Flag:$(DST)_B$ A zero is shifted into $(DST)_B$ bit Ø, and $(DST)_B$ bit 7 is shifted into the C-flag.
INDICATORS:	N = set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of $(DST)_B$ bit 7

SETB SET BYTE TO ONES

FORMAT:	SETB DST
OPERATION:	$(DST)_B \leftarrow$ "FF"
FUNCTION:	The destination byte operand is set to all ones
INDICATORS:	N = Set
	Z = Reset
	V = Reset
	C = Unchanged

CLRB CLEAR BYTE TO ZEROS

FORMAT:	CLRB DST
OPERATION:	$(DST)_B \leftarrow$ Ø
FUNCTION:	The destination byte operand is cleared to all zeros.
INDICATORS:	N = Reset
	Z = Set
	V = Reset
	C = Reset

ASRB ARITHMETIC SHIFT RIGHT BYTE

FORMAT:	ASRB DST
FUNCTION:	A 1-bit right arithmetic shift is done on $(DST)_B$: C-flag. Bit 7 of $(DST)_B$ is replicated. Bit Ø of $(DST)_B$ is shifted into the C-flag.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set to exclusive or of N and C flags
	C = Set to the value of the bit shifted out of $(DST)_B$ bit Ø

SWAD SWAP DIGITS

FORMAT:	SWAD DST
FUNCTION:	The two hex digits in the destination byte operand are exchanged with each other.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set if $(DST)_B$ bit 7 is set
	C = Reset

COMB COMPLEMENT BYTE

FORMAT:	COMB DST
OPERATION:	$(DST)_B \leftarrow \overline{(DST)}_B$
FUNCTION:	The destination byte operand is one's complemented
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Reset
	C = Set

NEGB NEGATE BYTE

FORMAT:	NEGB DST
OPERATION:	$(DST)_B \leftarrow -(DST)_B$
FUNCTION:	The destination byte operand is two's complemented
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set if $(DST)_B$ = "8ØØØ"
	C = Reset if $(DST)_B$ = Ø

INCB INCREMENT BYTE

FORMAT:	INCB DST
OPERATION:	$(DST)_B \leftarrow (DST)_B + 1$
FUNCTION:	The destination byte operand is incremented by one
INDICATORS:	N = Set if $(DST)_B$ is set
	Z = Set if $(DST)_B$ = Ø
	V = Set if $(DST)_B$ = "8ØØØ"
	C = Set if a carry is generated from $(DST)_B$ bit 7

DECB DECREMENT BYTE

FORMAT:	DECB DST
OPERATION:	$(DST)_B \leftarrow (DST)_B - 1$
FUNCTION:	The destination byte operand is decremented by one
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Set if $(DST)_B$ = "7FFF"
	C = Set if a borrow is generated from $(DST)_B$ bit 7

FORMAT 8 OP CODES*

DOUBLE OPS - SINGLE WORD - SMØ AND DMØ ONLY

```
15             6  5      3  2      Ø
[     OPC       | S REG    | D REG  ]
```

There are 8 op codes in this class representing op codes "ØEØØ" to "ØFFF". Only addressing mode Ø is allowed for both the source and destination. All are one word op codes, and all are block move instructions. The last 4 can be used as pseudo DMA ops in some hardware configurations. In all cases the source register contains the address of the first word or byte of memory to be moved, and the destination register contains the address of the first word or byte of memory to receive the data being moved. The number of words or bytes being moved is contained in RØ. The count ranges from 1-65536 (Ø = 65536) words or bytes. The count in RØ is an unsigned positive integer. None of the indicators are altered by these op codes.

Each of these op codes is interruptable at the end of each word or byte transfer. If no interrupt requests are active the transfers continue. PC is not incremented to the next op code until the op code is completed. This allows for complete interruptability as long as register integrity is maintained during the interrupt.

BASE OP CODE	MNEMONIC
ØEØØ	MBWU
ØE4Ø	MBWD
ØE8Ø	MBBU
ØECØ	MBBD
ØFØØ	MBWA
ØF4Ø	MBBA
ØF8Ø	MABW
ØFCØ	MABB

* NOTE: These op codes are all in the third microm.

MBWU — MOVE BLOCK OF WORDS UP

FORMAT: MBWU SRC, DST

FUNCTION: The word string beginning with the word addressed by the source register is moved to successively increasing word addresses as specified by the destination register. The source and destination registers are each incremented by two after each word is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

MBWD — MOVE BLOCK OF WORDS DOWN

FORMAT: MBWD SRC, DST

FUNCTION: The word string beginning with the word addressed by the source register is moved to successively decreasing word addresses as specified by the destination register. The source and destination registers are each decremented by two after each word is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

INDICATORS: Unchanged

MBBU — MOVE BLOCK OF BYTES UP

FORMAT: MBBU SRC, DST

FUNCTION: The byte string beginning with the byte addressed by the source register is moved to successively increasing byte addresses as specified by the destination register. The source and destination registers are each incremented by one after each byte is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

INDICATORS: Unchanged.

MBBD — MOVE BLOCK OF BYTES DOWN

FORMAT: MBBD SRC, DST

FUNCTION: The byte string beginning with the byte addressed by the source register is moved to successively decreasing byte addresses as specified by the destination register. The source register, destination register, and RØ, are each decremented by one after each byte is transferred. Transfers continue until RØ = Ø.

INDICATORS: Unchanged

MBWA — MOVE BLOCK OF WORDS TO ADDRESS

FORMAT: MBWA SRC, DST

FUNCTION: Same as MBWU except that the destination register is never incremented.

INDICATORS: Unchanged

MBBA — MOVE BLOCK OF BYTES TO ADDRESS

FORMAT: MBBA SRC, DST

FUNCTION: Same as MBBU except that the destination register is never incremented.

INDICATORS: Unchanged

MABW — MOVE ADDRESS TO BLOCK OF WORDS

FORMAT: MABW SRC, DST

FUNCTION: Same as MBWU except that the source register is never incremented.

INDICATORS: Unchanged

MABB — MOVE ADDRESS TO BLOCK OF BYTES

FORMAT: MABB SRC, DST

FUNCTION: Same as MBBU except that the source register is never incremented.

INDICATORS: Unchanged

FORMAT 9 OP CODES

DOUBLE OPS - ONE OR TWO WORDS - SMØ, DMØ to DM7

15 9	8 6	5 3	2 Ø
OPC	S REG	D MODE	D REG

There are 8 op codes in this class representing op codes "7ØØØ" to "7FFF". Source mode Ø addressing only is allowed, but destination modes Ø - 7 are allowed for all op codes except 3: JSR and LEA with DMØ will cause an illegal instruction format trap (see chapter 2), and SOB is a special format unique to itself. It is included here only because its destination field is 6 bits long. SOB is a branch instruction. Its 6 bit destination field is a positive word offset from PC, which points to the op code that follows, backwards to the desired address. Forward branching is not allowed. SOB is always a one word op code, and it is used for fast loop control. All other op codes are one word long for DMØ to DM5 addressing and two words long for DM6 or DM7 addressing. The rules for PC relative addressing with DM6 or DM7 are the same as they are for the format 7 op codes. Preliminary decoding of all these op codes except SOB presets the indicator flags as follows: N = 1, Z = Ø, V = Ø, C = 1.

BASE OP CODE	MNEMONIC
7ØØØ	JSR
72ØØ	LEA
74ØØ	ASH
76ØØ	SOB
78ØØ	XCH
7AØØ	ASHC
7CØØ	MUL
7EØØ	DIV

JSR JUMP TO SUBROUTINE

FORMAT:	JSR REG, DST

OPERATION:	↓SP, @SP ← REG
	REG ← PC
	PC ← DST

FUNCTION:	The linkage register is pushed onto the stack; PC, which points to the op code that follows, is placed in the linkage register; and the destination address is placed in PC. DMØ is illegal. The assembler recognizes the format "CALL DST" as being equivalent to "JSR PC, DST".
INDICATORS:	Preset

LEA LOAD EFFECTIVE ADDRESS

FORMAT:	LEA REG, DST
OPERATION:	REG ← DST

FUNCTION:	The destination address is placed into the source register. DMØ is illegal. The assembler recognizes the format "JMP DST" as being equivalent to "LEA PC,DST".
INDICATORS:	Preset

XCH EXCHANGE

FORMAT:	XCH REG, DST
OPERATION:	REG $\rightleftarrows$ (DST)
FUNCTION:	The source register and destination contents are exchanged with each other.
INDICATORS:	Preset

SOB SUBTRACT ONE AND BRANCH (IF ≠ Ø)

FORMAT:	SOB REG, DST

OPERATION:	REG ← REG - 1
	IF REG ≠ Ø, PC ← PC -(OFFSET *2)

FUNCTION:	The source register is decremented by one. If the result is not zero then twice the value of the destination offset is subtracted from PC.
INDICATORS:	Unchanged

ASH ARITHMETIC SHIFT

FORMAT:	ASH REG, DST
FUNCTION:	The source register is shifted arithmetically with the number of bits and direction specified by the destination operand. If (DST) = Ø no shifting occurs. If (DST) = -X then REG is shifted right arithmetically X bits as in an SSRA. If (DST) = +X then REG is shifted left arithmetically X bits as in an SSLA. Only an 8 bit destination operand is used. Thus, DST is a byte address. For DMØ only the lower byte of the destination register is used.
INDICATORS:	Preset if (DST) = Ø . Otherwise: N = Set if REG bit 15 is set Z = Set if REG = Ø V = Set to exclusive or of N and C flags C = Set to the value of the last bit shifted out of REG

ASHC ARITHMETIC SHIFT COMBINED

FORMAT:	ASHC REG, DST
FUNCTION:	Exactly the same as ASH except that the shift is done on REG+1:REG. All other comments apply.

INDICATORS:	Preset if (DST) = Ø. Otherwise:
	N = Set if REG+1 bit 15 is set
	Z = Set if REG+1: REG = Ø
	V = Reset
	C = Set to the value of the last bit shifted out

MUL MULTIPLY

FORMAT: MUL REG, DST

OPERATION: REG+1:REG ← REG *(DST)

FUNCTION: An unsigned multiply is performed on the source register and the destination operand. The unsigned 32 bit result is placed in REG+1:REG.

INDICATORS:
N = Set if REG+1 bit 15 is set
Z = Set if REG+1:REG = Ø
V = Reset
C = Indeterminate

DIV DIVIDE

FORMAT: DIV REG, DST

OPERATION: REG ←[REG+1:REG/(DST)]
REG+1 ← REMAINDER

FUNCTION: An unsigned divide is performed on the 32 bit source operand REG+1:REG and the destination operand. The unsigned result is placed in REG, and the unsigned remainder is placed in REG+1.No divide occurs and the V-flag is set if REG+1 is greater than or equal to (DST) since the result will not fit into 16 bits. If the divisor is zero both the V and C flags are set.

INDICATORS:
If no division error:
N = Set if REG bit 15 is set
Z = Set if REG = Ø
V = Reset
C = Indeterminate
If division error:
N = Reset
Z = Reset
V = Set
C = set if (DST) = Ø

FORMAT 10 OP CODES

DOUBLE OPS - ONE TO THREE WORDS - SMØ TO SM7, DMØ TO DM7.

15 — 12	11 — 9	8 — 6	5 — 3	2 — Ø
OPC	S MODE	S REG	D MODE	D REG

There are 12 op codes in this class representing op codes "1ØØØ" to "6FFF" and "9ØØØ" to "EFFF". Nine of the op codes are word ops. Three are byte ops. Full source and destination mode addressing with any register is allowed. A one word op code is generated for SMØ-SM5 and DMØ-DM5 addressing. A two word op code is generated for either SM6-SM7 or DM6-DM7 addressing, but not both. For both SM6-SM7 and DM6-DM7 addressing a three word op code is generated. For a two word op code with word #1 at location X: X + 2 contains the source or destination offset and PC = X + 4 if PC is the register that applies to the offset in location X + 2. For a three word op code with word #1 at location X: X + 2 contains the source offset and X + 4 contains the destination offset. If the source register is PC then PC = X + 4 when added to the offset to compute the source address. If the destination register is PC then PC = X + 6 when added to the offset to compute the destination address.

BASE OP CODE	MNEMONIC
1ØØØ	ADD
2ØØØ	SUB
3ØØØ	AND
4ØØØ	BIC
5ØØØ	BIS
6ØØØ	XOR
9ØØØ	CMP
AØØØ	BIT
BØØØ	MOV
CØØØ	CMPB
DØØØ	MOVB
EØØØ	BISB

WORD OPS

ADD	ADD
FORMAT:	ADD SRC, DST
OPERATION:	(DST) ← (SRC) + (DST)
FUNCTION:	The source and destination operands are added together, and the sum is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if both operands were of the same sign and the result was of the opposite sign
	C = Set if a carry is generated from bit 15 of the result

SUB SUBTRACT

FORMAT:	SUB SRC, DST
OPERATION:	(DST) $\leftarrow$ (DST) - (SRC)
FUNCTION:	The two's complement of the source operand is added to the destination operand, and the sum is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Set if operands were of different signs and the sign of the result is the same as the sign of the source operand
	C = Set if a borrow is generated from bit 15 of the result

AND AND

FORMAT:	AND SRC, DST
OPERATION:	(DST) $\leftarrow$ (SRC) Δ (DST)
FUNCTION:	The source and destination operands are logically ANDED together, and the result is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

BIC BIT CLEAR

FORMAT:	BIC SRC, DST
OPERATION:	(DST) $\leftarrow$ $\overline{(SRC)}$ Δ (DST)
FUNCTION:	The one's complement of the source operand is logically ANDED with the destination operand, and the result is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

BIS BIT SET

FORMAT:	BIS SRC, DST
OPERATION:	(DST) $\leftarrow$ (SRC) $\vee$ (DST)
FUNCTION:	The source and destination operands are logically ORED, and the result is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

XOR EXCLUSIVE OR

FORMAT:	XOR SRC, DST
OPERATION:	(DST) $\leftarrow$ (SRC) $\veebar$ (DST)
FUNCTION:	The source and destination operands are logically EXCLUSIVE ORED, and the result is placed in the destination.

INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

CMP COMPARE

FORMAT:	CMP SRC, DST
OPERATION:	(SRC) - (DST)
FUNCTION:	The destination operand is subtracted from the source operand, and the result sets the indicators. Neither operand is altered.
INDICATORS:	N = Set if result bit 15 is set Z = Set if result = Ø V = Set if operands were of opposite sign and the sign of the result is the same as the sign of (DST) C = Set if a borrow is generated from bit 15 of the result

BIT BIT TEST

FORMAT:	BIT SRC, DST
OPERATION:	(SRC) $\wedge$ (DST)
FUNCTION:	The source and destination operands are logically ANDED, and the result sets the indicators. Neither operand is altered.

INDICATORS:	N = Set if result bit 15 is set
	Z = Set if result = Ø
	V = Reset
	C = Unchanged

MOV MOVE

FORMAT:	MOV SRC, DST
OPERATION:	(DST) $\leftarrow$ (SRC)
FUNCTION:	The destination operand is replaced with the source operand.

INDICATORS:	N = Set if (DST) bit 15 is set
	Z = Set if (DST) = Ø
	V = Reset
	C = Unchanged

BYTE OPS

For SMØ addressing only the lower byte of the source register is used as an operand. For SM1-SM7 addressing only the addressed memory byte is used as an operand. For DMØ addressing only the lower byte of the destination register is used as an operand with one exception: MOVB will extend the sign through bit 15. For DM1-DM7 addressing only the addressed memory byte is used as an operand.

CMPB COMPARE BYTE

FORMAT:	CMPB SRC, DST
OPERATION:	$(SRC)_B$ - $(DST)_B$

FUNCTION:	The destination operand is subtracted from the source operand, and the result sets the indicators. Neither operand is altered.
INDICATORS:	N = Set if result bit 7 is set
	Z = Set if result = Ø
	V = Set if operands were of different signs and the sign of the result is the same as the sign of $(DST)_B$.
	C = Set if a borrow is generated from result bit 7

MOVB — **MOVE BYTE**

FORMAT:	MOVB SRC, DST
OPERATION:	$(DST)_B \leftarrow (SRC)_B$
FUNCTION:	The destination operand is replaced with the source operand. If DMØ the sign bit (bit 7) is replicated through bit 15.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Reset
	C = Unchanged

BISB — **BIT SET BYTE**

FORMAT:	BISB SRC, DST
OPERATION:	$(DST)_B \leftarrow (SRC)_B \vee (DST)_B$
FUNCTION:	The source and destination operands are logically ORED, and the result is placed in the destination.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set
	Z = Set if $(DST)_B$ = Ø
	V = Reset
	C = Unchanged

When using auto increments or decrements in either the source or destination (or both) fields the user must remember the following rule: All increments or decrements in the source are fully completed before any destination decoding begins even if the same index register is used in both the source and destination. The two fields are totally independent.

FORMAT 11 OP CODES

DOUBLE OPS - ONE WORD - FLOATING POINT.

15 12	11 8	7	6 4	3	2 ∅
1111	OPC	I	SRC	I	DST

There are 16 OP Codes in this class representing OP Codes "F∅∅∅" to "FFFF". Only five are currently defined. They reside in the third microm along with the Format 8 OP Codes. The remaining 11 OP Codes are mapped to the fourth microm for future expansion or customized user OP Codes. All are one word long. Two source and destination addressing modes are available. These two modes, FP∅ and FP1, are unique to these OP Codes. Each consists of a 3-bit Register Designation and a 1 bit indirect flag preceeding the register designator. For FP∅ the indirect bit is ∅, and FP1 it is one. Both the source and destination fields have both addressing modes. The modes are defined as follows:

FP∅ The designated register contains the address of the operand.

FP1 The designated register contains the address of the address of the operand.

FP∅ is the same as standard addressing mode 1, and FP1 is the same as standard addressing mode 7 with an offset of zero.

The computed address is the address of the first word of a 3 word floating point operand. The first word contains the sign, exponent, and high byte of the mantissa. The next higher address contains the middle two bytes of the mantissa, and the next higher address after that contains the lowest two bytes of the mantissa. This format is half way between single and double precision floating point formats, and it represents the most efficient use of microprocessor ROM and register space. The complete format is as follows:

1. A 1 bit sign for the entire number which is zero for positive.
2. An 8-bit base-two exponent in excess-128 notation with a range of +127, -128. The only legal number with an exponent of -128 is true zero (all zeros).
3. A 40 bit mantissa with the MSB implied.

Since every operand is assumed to be normalized upon entry and every result is normalized before storage in the destination addresses, and since a normalized mantissa has a MSB equal to one, then only 39 bits need to be stored. The MSB is implied to be a one, and the bit position it normally occupies is taken over by the exponent to increase its range by a factor of two. The full format of a floating point operand is a follows:

	15	14 7	6 ∅
LOCATION X:	S	EXPONENT	MANTISSA (HIGH)

	15 8	7 ∅
LOCATION X+2:	MANTISSA	(MIDDLE)
LOCATION X+4:	MANTISSA	(LOW)

True zero is represented by a field of 48 zeroes. In effect, the CPU considers any number with an exponent of all zeroes (-128) to be a zero during multiplication and division. For add and subtract the only legal number with an exponent of -128 is true zero. All others cause erroneous results. No registers are modified by any Format 11 OP Code. However, to make room internally for computations 4 registers are saved in memory locations "30" - "38" during the exelution of FADD, FSUB, FMUL and FDIV. These registers are retrieved at the completion of the OP Codes. The registers saved are: the destination address, SP, PC and RØ. No Format 11 OP Code is interruptable (for obvious reasons). FMUL uses location "38" for temporary storage of partial results.

FLOATING POINT ERROR TRAPS

Location "3E" is defined as the floating point error trap PC. Whenever an overflow, underflow, or divide by zero occurs a standard trap call is executed with PS and PC pushed onto the stack, and PC fetched from location "3E". I2 is not altered. The remaining memory locations that are reserved for the floating point option ("3A and "3C") are not currently used. The status of the indicator flags and destination addresses during the 3 trap conditions are defined as follows:

FOR UNDERFLOW (FADD, FSUB, FMUL, FDIV)

N = 1	Destination contains all zeroes
Z = Ø	(true zero).
V = 1	
C = Ø	

FOR OVERFLOW (FADD, FSUB, FMUL)

N = Ø	Destination not altered in any way.
Z = Ø	
V = 1	
C = Ø	

FOR OVER FLOW (FDIV)

N = Ø	Destination not altered if overflow detected
Z = Ø	during exponent computation. Undefined
V = 1	otherwise. (Used to save unnormalized
C = Ø	partial results during a divide).

FOR DIVIDE BY ZERO (FDIV)

N = 1	Destination not altered in any way.
Z = Ø	
V = 1	
C = 1	

RESERVED TRAPS

If the third microm is in the system and the fourth is not then the last 11 floating point OP codes are the only ones that will cause a reserved OP code trap if executed. If the third microm is not in the system then all Format 8 and 11 OP Codes will cause a reserved OP code trap if executed. However, since the Format 8 OP Codes are interruptable the PC is not advance until the completion of the moves. In all other cases PC is advanced when the OP Code is fetched. For these reasons the PC that is saved onto the stack will point to the offending OP Code during a reserved OP Code trap if and only if the offending OP Code is a Format 8 OP Code. For the Format 11 OP Codes the saved PC will point to the OP Code that follows the offending OP Code. If the user wishes to identify which OP Code caused the reserved OP Code trap he must not preceed a Format 8 OP Code with a Format 11 OP Code or a literal that looks like a Format 11 OP Code.

BASE OP CODE	MNEMONIC
FØØØ	FADD
F1ØØ	FSUB
F2ØØ	FMUL
F3ØØ	FDIV
F4ØØ	FCMP
F5ØØ	
F6ØØ	
F7ØØ	
F8ØØ	
F9ØØ	
FAØØ	
FBØØ	
FCØØ	
FDØØ	
FEØØ	
FFØØ	

FADD — FLOATING POINT ADD

FORMAT: FADD SRC,DST

OPERATION: $(DST) \leftarrow (DST) + (SRC)$

FUNCTION: The source and destination operands are added together, normalized, and the result is stored in place of the destination operand.

INDICATORS: (if no errors)
N = Set if the result sign is negative (set).
Z = Set if the result is zero
V = Reset
C = Reset

FSUB — FLOATING POINT SUBTRACT

FORMAT: FSUB SRC, DST

OPERATION: $(DST) \leftarrow (DST) - (SRC)$

FUNCTION: The source operand is subtracted from the destination operand. The result is normalized and stored in place of the destination operand.

WARNING: THIS OP CODE COMPLEMENTS THE SIGN OF THE SOURCE OPERAND IN MEMORY AND DOES AN FADD.

INDICATORS: (if no errors)
N = Set if the result sign is negative (set)
Z = Set if the result is zero.

V = Reset
C = Reset

FMUL FLOATING POINT MULTIPLY

FORMAT: FMUL SRC, DST
OPERATION: (DST) ←(DST) *(SRC)
FUNCTION: The source and destination operands are multiplied together, normalized, and the result is stored in place of the destination operand.
INDICATORS: (if no errors)
N = Set if the sign of the result is negative (set).
Z = Set if the result is zero
V = Reset
C = Reset

FDIV FLOATING POINT DIVIDE

FORMAT: FDIV SRC, DST
OPERATION: (DST) ←(DST) / (SRC)
FUNCTION: The destination operand is divided by the source operand. The result is normalized and stored in place of the destination operand.
INDICATORS: (if no errors)
N = Set if the sign of the result is negative (set).
Z = Set if the result is zero
V = Reset
C = Reset

FCMP FLOATING POINT COMPARE

FORMAT: FCMP SRC, DST
OPERATION: (SRC) - (DST)
FUNCTION: The destination operand is compared to the source operand, and the indicators are set to allow a SIGNED conditional branch.
INDICATORS:
N = Set if result is negative
Z = Set if result is zero
V = Set if arithmetic underflow occurs.*
C = Set if a borrow is generated. *

*NOTE: True if first words of both operands are not equal.

CAUTION: The same physical operand may be used as both the source and destination operand for any of the above floating point OP Codes with no abnormal results except two. They are:
1) If an error trap occurs the operand will probably be altered.
2) An FSUB gives an answer of -2x, if x ≠ Ø, instead of Ø.

APPENDIX A

NUMERIC OP CODE TABLE

OP CODE	MNEMONIC
ØØØØ ØØØØ ØØØØ ØØØØ	NOP
ØØØØ ØØØØ ØØØØ ØØØ1	RESET
ØØØØ ØØØØ ØØØØ ØØ1Ø	IEN
ØØØØ ØØØØ ØØØØ ØØ11	IDS
ØØØØ ØØØØ ØØØØ Ø1ØØ	HALT
ØØØØ ØØØØ ØØØØ Ø1Ø1	XCT
ØØØØ ØØØØ ØØØØ Ø11Ø	BPT
ØØØØ ØØØØ ØØØØ Ø111	WFI
ØØØØ ØØØØ ØØØØ 1ØØØ	RSVC
ØØØØ ØØØØ ØØØØ 1ØØ1	RRTT
ØØØØ ØØØØ ØØØØ 1Ø1Ø	SAVE
ØØØØ ØØØØ ØØØØ 1Ø11	SAVS
ØØØØ ØØØØ ØØØØ 11ØØ	REST
ØØØØ ØØØØ ØØØØ 11Ø1	RRTN
ØØØØ ØØØØ ØØØØ 111Ø	RSTS
ØØØØ ØØØØ ØØØØ 1111	RTT
ØØØØ ØØØØ ØØØ1 ØREG	IAK
ØØØØ ØØØØ ØØØ1 1REG	RTN
ØØØØ ØØØØ ØØ1Ø ØREG	MSKO
ØØØØ ØØØØ ØØ1Ø 1REG	PRTN
ØØØØ ØØØØ ØØ11 ARGU	LCC
ØØØØ ØØØØ Ø1AR GUME	SVCA
ØØØØ ØØØØ 1ØAR GUME	SVCB
ØØØØ ØØØØ 11AR GUME	SVCC
ØØØØ ØØØ1 DISP LACE	BR
ØØØØ ØØ1Ø DISP LACE	BNE
ØØØØ ØØ11 DISP LACE	BEQ
ØØØØ Ø1ØØ DISP LACE	BGE
ØØØØ Ø1Ø1 DISP LACE	BLT
ØØØØ Ø11Ø DISP LACE	BGT
ØØØØ Ø111 DISP LACE	BLE
ØØØØ 1ØØR EGØØ VALU	ADDI
ØØØØ 1ØØR EGØ1 VALU	SUBI
ØØØØ 1ØØR EG1Ø VALU	BICI
ØØØØ 1ØØR EG11 VALU	MOVI
ØØØØ 1Ø1Ø ØØMO DREG	ROR
ØØØØ 1Ø1Ø Ø1MO DREG	ROL
ØØØØ 1Ø1Ø 1ØMO DREG	TST
ØØØØ 1Ø1Ø 11MO DREG	ASL
ØØØØ 1Ø11 ØØMO DREG	SET
ØØØØ 1Ø11 Ø1MO DREG	CLR
ØØØØ 1Ø11 1ØMO DREG	ASR
ØØØØ 1Ø11 11MO DREG	SWAB
ØØØØ 11ØØ ØØMO DREG	COM
ØØØØ 11ØØ Ø1MO DREG	NEG
ØØØØ 11ØØ 1ØMO DREG	INC
ØØØØ 11ØØ 11MO DREG	DEC

OP CODE				MNEMONIC
ØØØØ	11Ø1	ØØMO	DREG	IW2
ØØØØ	11Ø1	Ø1MO	DREG	SXT
ØØØØ	11Ø1	1ØMO	DREG	TCALL
ØØØØ	11Ø1	11MO	DREG	TJMP
ØØØØ	111Ø	ØØSR	CDST	MBWU
ØØØØ	111Ø	Ø1SR	CDST	MBWD
ØØØØ	111Ø	1ØSR	CDST	MBBU
ØØØØ	111Ø	11SR	CDST	MBBD
ØØØØ	1111	ØØSR	CDST	MBWA
ØØØØ	1111	Ø1SR	CDST	MBBA
ØØØØ	1111	1ØSR	CDST	MABW
ØØØØ	1111	11SR	CDST	MABB
ØØØ1	SRCR	EGDS	TREG	ADD
ØØ1Ø	SRCR	EGDS	TREG	SUB
ØØ11	SRCR	EGDS	TREG	AND
Ø1ØØ	SRCR	EGDT	TREG	BIC
Ø1Ø1	SRCR	EGDT	TREG	BIS
Ø11Ø	SRCR	EGDS	TREG	XOR
Ø111	ØØØR	RRDS	TREG	JSR
Ø111	ØØ1R	RRDS	TREG	LEA
Ø111	Ø1ØR	RRDS	TREG	ASH
Ø111	Ø11R	RROF	FSET	SOB
Ø111	1ØØR	RRDS	TREG	XCH
Ø111	1Ø1R	RRDS	TREG	ASHC
Ø111	11ØR	RRDS	TREG	MUL
Ø111	111R	RRDS	TREG	DIV
1ØØØ	ØØØØ	DISP	LACE	BPL
1ØØØ	ØØØ1	DISP	LACE	BMI
1ØØØ	ØØ1Ø	DISP	LACE	BHI
1ØØØ	ØØ11	DISP	LACE	BLOS
1ØØØ	Ø1ØØ	DISP	LACE	BVC
1ØØØ	Ø1Ø1	DISP	LACE	BVS
1ØØØ	Ø11Ø	DISP	LACE	BCC, BHIS
1ØØØ	Ø111	DISP	LACE	BCS, BLO
1ØØØ	1ØØR	EGØØ	VALU	SSRR
1ØØØ	1ØØR	EGØ1	VALU	SSLR
1ØØØ	1ØØR	EG1Ø	VALU	SSRA
1ØØØ	1ØØR	EG11	VALU	SSLA
1ØØØ	1Ø1Ø	ØØMO	DREG	RORB
1ØØØ	1Ø1Ø	Ø1MO	DREG	ROLB
1ØØØ	1Ø1Ø	1ØMO	DREG	TSTB
1ØØØ	1Ø1Ø	11MO	DREG	ASLB
1ØØØ	1Ø11	ØØMO	DREG	SETB
1ØØØ	1Ø11	Ø1MO	DREG	CLRB
1ØØØ	1Ø11	1ØMO	DREG	ASRB
1ØØØ	1Ø11	11MO	DREG	SWAD
1ØØØ	11ØØ	ØØMO	DREG	COMB
1ØØØ	11ØØ	Ø1MO	DREG	NEGB
1ØØØ	11ØØ	1ØMO	DREG	INCB
1ØØØ	11ØØ	11MO	DREG	DECB

OP CODE				MNEMONIC
1ØØØ	11Ø1	ØØMO	DREG	LSTS
1ØØØ	11Ø1	Ø1MO	DREG	SSTS
1ØØØ	11Ø1	1ØMO	DREG	ADC
1ØØØ	11Ø1	11MO	DREG	SBC
1ØØØ	111R	EGØØ	VALU	SDRR
1ØØØ	111R	EGØ1	VALU	SDLR
1ØØØ	111R	EG1Ø	VALU	SDRA
1ØØØ	111R	EG11	VALU	SDLA
1ØØ1	SRCR	EGDS	TREG	CMP
1Ø1Ø	SRCR	EGDS	TREG	BIT
1Ø11	SRCR	EGDS	TREG	MOV
11ØØ	SRCR	EGDS	TREG	CMPB
11Ø1	SRCR	EGDS	TREG	MOVB
111Ø	SRCR	EGDS	TREG	BISB
1111	ØØØØ	ISRC	IDST	FADD
1111	ØØØ1	ISRC	IDST	FSUB
1111	ØØ1Ø	ISRC	IDST	FMUL
1111	ØØ11	ISRC	IDST	FDIV
1111	Ø1ØØ	ISRC	IDST	FCMP
1111	Ø1Ø1	ISRC	IDST	
1111	Ø11Ø	ISRC	IDST	
1111	Ø111	ISRC	IDST	
1111	1ØØØ	ISRC	IDST	
1111	1ØØ1	ISRC	IDST	
1111	1Ø1Ø	ISRC	IDST	
1111	1Ø11	ISRC	IDST	
1111	11ØØ	ISRC	IDST	
1111	11Ø1	ISRC	IDST	
1111	111Ø	ISRC	IDST	
1111	1111	ISRC	IDST	

APPENDIX B

ASSEMBLER NOTES

FORMAT 1 OP CODES

All are one word op codes except SAVS which is a two word op code. The second word of the SAVS op code is an absolute value.

FORMAT 2 OP CODES

All are one word with a 3 bit register argument

FORMAT 3 OP CODE

A one word op code with a 4 bit numeric argument

FORMAT 4 OP CODES

All are one word with a 6 bit numeric argument

FORMAT 5 OP CODES

All are one word with an 8 bit signed PC relative word displacement. The displacement is relative to op code+2. Maximum displacement from the op code is +128, -127 words.

FORMAT 6 OP CODES

All are one word with a 3 bit register and a 4 bit numeric argument. The stored numeric argument is a positive number from Ø -"F" that equals the actual numeric argument (1-"1Ø") minus one.

FORMAT 7 OP CODES

All are one word op codes for DMØ - DM5 addressing and two word op codes for DM6 - DM7 addressing. For DM6- DM7 addressing the offset is in the second word. If the index register is PC with DM6 - DM7 the offset is relative to op code+4.

FORMAT 8 OP CODES

All are one word with a 3 bit source and a 3 bit destination register argument. The count register is implied to be RØ.

FORMAT 9 OP CODES

All have a 3 bit register argument with a 6 bit destination argument that allows DMØ - DM7 addressing. For DMØ - DM5 a one word op code is generated. For DM6 - DM7 a two word op code is generated with the offset in word two. If the index register is PC with DM6-DM7 then the offset is relative to op code+4.

FORMAT 10 OP CODES

All have a 6 bit source and a 6 bit destination argument that allow SMØ - SM7 and DMØ - DM7 addressing. For SMØ - SM5 and DMØ - DM5 combined addressing a one word op code is generated. For SM6-SM7 or DM6 - DM7 but not both a two word op code is generated with the offset in word two. If the field with mode 6 or 7 addressing uses PC as the index register then the offset is relative to the op code + 4. For SM6 - SM7 and DM6 - DM7 combined addressing a 3 word op code is generated. Word two contains the source offset, and word 3 contains the destination offset. For SM6 - SM7 with PC the offset is relative to the op code + 4. For DM6 - DM7 with PC the offset is relative to the op code + 6.

Any autoincrements/decrements in the source are fully completed before any destination decoding bègins.

FORMAT 11 OP CODES

All are one word op codes with a 4 bit source and a 4 bit destination argument. Each argument consists of a 3 bit register argument preceeded by a 1 bit indirect argument.

APPENDIX C

PROGRAMMING NOTES

Several of the op codes and addressing modes have personality peculiarities that the user should be aware of. Most of these can be put to good use in particular situations. This appendix attempts to list most of them.

IEN: This instruction allows one more instruction to begin execution before enabling I2.

IDS: This instruction allows one more instruction to begin execution before disabling I2. IDS is therefore interruptable. If such a situation occurs the status of I2 that is included in the pushed PC will equal Ø.

HALT: There is no halt in the microcode. A selection of options is therefore given that allows the user to define HALT for himself.

ADDRESSING MODES

In order to clarify the function of the various addressing modes several programming examples are given. In each case assume that the first word of the op code is at location X.

SET RØ

Register RØ is set to all ones.

CLR @R2

The memory location pointed to by R2 is cleared to zeros. If R2 contained a "Ø1ØØ" the memory word address "Ø1ØØ" would be cleared.

INC (R3)+

The memory location pointed to by R3 is incremented by one. R3 is then incremented by 2.

DEC (PC)+

Location X + 2 is decremented by one, and program control is advanced to location X + 4. This allows for in-line literals in a program, a method that saves a word of memory in most cases.

SWAB @(R4)+

If R4 contains a "Ø1ØØ" and location "Ø1ØØ" contains a "Ø2ØØ" then the two bytes in location "Ø2ØØ" are swapped and R4 is incremented to "Ø1Ø2".

COM -(R5)

R5 is decremented by two. The address specified by the altered R5 is one's complemented.

NEG -(PC)

A BOZO no-no since location X is the location negated and program control is again transferred to location X after the negation is completed.

TST @-(R1)

If R = "Ø1Ø4" and location "Ø1Ø2" contains a "1ØØØ" then the following sequence occurs: (1) R1 is decremented by 2 to "Ø1Ø2". (2) The contents of location "Ø1Ø2" (i.e. "1ØØØ") becomes the address of the operand to be tested.

ROR 4(R4)

The contents of memory location R4 + 4 is rotated right. R4 is not altered. Word two of this op code contains a 4. Program control is advanced to location X + 4 at the completion of the rotate.

ROL @6(SP)

The contents of memory location SP + 6 contains the address of the operand to be rotated. Word two of this op code contains a 6. Program control is advanced to location X + 4 at the completion of the rotate.

JSR PC,TAG

Location X + 2 contains the byte offset from location "TAG" to location X + 4. The address of location X + 4 is pushed onto the stack, and the address of location "TAG" is placed in PC.

JSR R5,TAG

Location X + 2 contains the byte offset from location "TAG" to location X + 4. The content of register R5 is pushed onto the stack, the address of location X + 4 is placed in R5, and the address of location "TAG" is placed in PC.

JSR PC,(R4)+

Location X + 2 is pushed onto the stack, R4 is moved to PC, and R4 is incremented by two.

JSR PC,@(SP)+

This is a co-routine call. Pay attention:
1) The contents of the location pointed to by SP is saved in CPU register "TMPA".

2) SP is incremented by two.
3) The address of location X + 2 is pushed onto the stack
4) CPU register "TMPA" is moved to PC

The effect of all this is to swap the top word on the stack with the address of location X + 2 without altering SP or stack size.

Consider the following routine.

```
SUBR:  JSR   PC,2(PC)
TAGA:  JSR   PC,@(PC)
TAGB:        .
             .
             .
       RTN   PC
```

The first JSR places the address of TAGA on the stack and executes the routine starting at TAGB. The RTN PC transfers control to location TAGA when it is executed. The second JSR places address TAGB onto the stack and into PC, effectively leaving PC unaltered. The second time the RTN PC is executed program control passes to location TAGB. The third time the RTN PC is executed program control passes back to the routine that call subroutine SUBR. Since TAGA and TAGB are never addressed explicitly both of the labels could be eliminated from the program. If left in then the "2(PC)" could be replaced with "TAGB".

<u>CMP (RØ)+, (RØ)+</u>

If RØ = "Ø1ØØ" then the contents of location "Ø1ØØ" is compared to the contents of location "Ø1Ø2" , and RØ is incremented to "Ø1Ø4". All source auto increments or decrements are completed before destination decoding begins.

<u>MOV @R2,-(R2)</u>

If R2 = "Ø1Ø6" then the contents of location "Ø1Ø6" is moved to location "Ø1Ø4", and R2 is decremented to "Ø1Ø4".

<u>BIT #2,@#4</u>

The contents of absolute memory location 4 is tested against the literal value 2. This is a three word op code with word two containing a 2 and word three containing a 4. This op code works on location 4 from anywhere in memory.

<u>CMP (PC)+,TAG</u>

This won't work. The assembler generates a two word op code for this with the destination offset in word two. The execution of the op code, however, uses word two as a literal and word three (which does not exist) as the destination offset. By swapping the source and destinations around then an in-line literal could be used for word three, and word two would contain a valid source offset.

JSR PC,(PC)+

The address of location X + 4 is pushed onto the stack, and PC gets the address of location X + 2.

JSR R5,(PC)+

The contents of R5 are pushed onto the stack, R5 gets the address of location X + 4, and PC gets the address of location X + 2.

MOVB (RØ)+,(RØ)+

If RØ = "Ø1Ø2" then the contents of memory byte location "Ø1Ø2" is moved to memory byte location "Ø1Ø3", and RØ is incremented to "Ø1Ø4".

MOVB (SP)+,R1

The contents of the memory byte addressed by SP is moved to the lower byte of R1, the sign bit (bit 7) is replicated through bit 15 of R1, and SP is incremented by 2. SP is always autoincremented or autodecremented by two.

CLRB (PC)+

The contents of the lower byte memory location X + 2 is cleared to zeros. The upper byte (X + 3) is not affected. PC is incremented by two. PC is always autoincremented or autodecremented by two.

BISB RØ,R1

The lower bytes of register RØ is logically ORED with the lower byte of register R1. The upper byte of R1 is not altered.

MOVB @(R2)+,@-(R3)

If R2 contains a "Ø1ØØ" and R3 contains a "Ø2ØØ" then location "Ø1ØØ" contains the byte address of the source operand and location "Ø1FE" contains the address of the destination byte that is to receive the source byte. R2 is incremented by two, and R3 is decremented by two since they point to addresses of (16 bit) addresses.

JSR SP,TAG

Not recommended since the value of the stack is lost. Perfectly legal however.

SAVS and RSTS

Although designed to be used for automatic register and I/O priority level saving and restoring, the lack of hardware priority masking does not alter the operation or the op codes. The SAVS op code is usually the first instruction executed in a device interrupt routine, and the RSTS is the last. The priority mask can use a one bit as an enable or disable with bit Ø the highest or lowest priority level. Such decisions are made by the hardware.

POWER FAIL

Two levels of power fail are provided for in the firmware. The hardware may use two, one, or no levels of power fail.The three modes are discussed in increasing order of complexity.

NO LEVELS: External address register bit 7 is hardwired to Ø, and a prayer is offered.

ONE LEVEL: The detection of a power fail sets bit 7 of the external status register and the CPU RESET line. When the power fail disappears the CPU RESET line is reset, but bit 7 of the external status register remains set. The Line Clock Clear State Code (see appendix D) clears bit 7 of the external status register (and bits 5, 6 if used). A system power up is then executed.

TWO LEVELS: This requires two hardware functions, AC LOW and DC LOW, plus two levels of power fail; AC and DC. It all works like this: If AC power begins to deteriorate AC LOW is set first. This sets bit 7 of the external status register and generates an interrupt via IØ or I1. If AC power does not deteriorate too far then nothing else happens except that bit 7 of the external status register is reset when power is restored. If AC power continues to deteriorate then eventually DC power will begin to deteriorate. When this happens DC LOW is set and DC LOW sets CPU RESET. AC LOW is still set and it maintains bit 7 of the external status register. When power is restored DC LOW is reset. This resets CPU RESET. A power up sequence is initiated, and the Line Clock Clear State (see appendix D) clears The External Status Register bit 7 (plus 5 and 6 if they are used). If the user wishes to be able to execute a programmed power fail routine even during a sudden and complete power failure then the DC power supply must be strong enough to run the CPU and MEMORY for at least 2 milliseconds. The power fail interrupt must also be programmed, and the interrupts enabled.

The use of the Line Clock Clear State Code to clear bits 5-7 on a CPU RESET function (plus the line clock of course) should have no effect on normal system operation. Should an error occur during a non-vectored interrupt the error would be cleared momentarily and then set again as CPU RESET obviously could not have been generated. If it had been then the system could not be in the non-vectored interrupt routine.

PARITY AND BUS ERRORS

These functions are also part of the CPU RESET function along with power fail/up. In order to get only one or the other then bit 7 of the external status register must be reset when the CPU RESET function

APPENDIX D

MICROM STATE CODE FUNCTIONS

Below is a list of MICROM STATE CODE FUNCTIONS for the WD1600 with a brief description of what each does. More elaborate descriptions, where necessary, follow the table.

CODE	MNEMONIC	FUNCTION
ØØØ1	PMSK	Priority mask out
ØØ1Ø	RUN	Macro instruction fetch
ØØ11	IORST	I/O reset
Ø1ØØ	INTEN	I2 set
Ø1Ø1	INTDS	I2 reset
Ø11Ø	ESRR	External status register request
Ø111	SRS	System reset
1ØØØ	BYTE	Read byte operation
1ØØ1	RMWW	Read-modify-write word
1Ø1Ø	RMWB	Read-modify-write byte
1Ø11	RLCI	Reset line clock interrupt
11ØØ	EARR	External address register request
11Ø1		Duplicate of "BYTE"
111Ø		Duplicate of "RMWW"
1111		Duplicate of "RMWB"

PMSK: The state code is generated on an OUTPUT WORD instruction when a new mask is written into location "2E". It signals the I/O devices that a new interrupt mask is on the DAL.

RUN: Generated during macro instruction fetch for a run light.

IORST: Generated during a RESET macro op code to reset I/O devices to some preset state.

INTEN: Enables the interrupt enable line -I2.

INTDS: Disables the interrupt enable line -I2.

ESRR: Generated during an INPUT STATUS BYTE micro op code to indicate that the external status register is being requested. See note 1.

SRS: Generated during a power up for a master system reset. This code is followed by a 300 cycle wait to allow time for any reset functions the hardware generates to be completed before any DAL requests are generated.

BYTE: Generated during an INPUT BYTE micro op code to indicate a read byte operation without a read-modify-write.

RMWW: Generated during an INPUT WORD micro op code with RMW active to indicate a read-modify-write word sequence.

RMWB: Generated during an INPUT BYTE micro op code with RMW active to indicate a read-modify-write byte sequence.

RLCI: Generated during a CPU RESET or a non-vectored interrupt without a power fail to clear both the line clock interrupt and external status register bits 5-7.

EARR: Generated during an INPUT STATUS BYTE micro op code to indicate a request for the external address register during the user bootstrap routine.

CODES "D" - "F": Duplicates of codes "8" - "A" respectively except that these codes appear as a part of the READ micro op codes instead of as a part of the INPUT micro op codes. Either or both may be used by the hardware as is convenient. These codes preceed the others. They are generated only once, however, instead of repeating in the event of a wait state as the others do.

NOTE 1: INPUT STATUS BYTE is not a function of reply and does not generate a SYNC. For these reasons the DAL must be tri-stated if a DMA device also exists. The data is always gated onto the lower byte. The upper byte is ignored.

NOTE 2: Lack of state codes "8" - "A" or "D" - "F" during a READ - INPUT sequence implies a read word operation without read-modify-write.

APPENDIX E

OP CODE TIMINGS

All times are in cycles. Timings include all OP Code fetches, memory reads, and memory writes applicable to each. Timings assume that the memory is running with full speed with respect to the CPU. This requires a 16 Bit access time = 1 CPU cycle, and a 16 Bit memory read/write cycle time = 2 CPU cycles. One CPU cycle = 300 NS @ 3.3 MHZ, UØØ NS @ 2.5 MHZ, and 500 NS @ 2 MHZ clock rates. Timings are included for SMØ and DMØ as basic with additions as necessary in tables that follow the OP Codes for SM1-7 and DM1-7 timings.

FORMAT ONE OP CODES

OP CODE	# CYCLES
NOP	1Ø
RESET	1Ø
IEN	1Ø
IDS	1Ø
HALT	16+
XCT	44 + OP CODE EXECUTED
BPT	24
WFI	16+
RSVC	62
RRTT	60
SAVE	46
SAVS	65
REST	48
RRTN	52
RSTS	64
RTT	13

FORMAT TWO-FOUR OP CODES

OP CODE	# CYCLES
IAK	1Ø
RTN	12
MSKO	1Ø
PRTN	22
LCC	7
SVCA	37
SVCB	73
SVCC	71

FORMAT FIVE OP CODES

All branches = 9 cycles if branch occurs or not.

FORMAT SIX OP CODES

OP CODE	# CYCLES
ADDI	9
SUBI	9
BICI	9
MOVI	9
SSRR	8 + (5 X # bits shifted)
SSLR	8 + (5 X # bits shifted)
SSRA	8 + (7 X # bits shifted)
SSLA	8 + (5 X # bits shifted)
SDRR	20 + (7 X # bits shifted)
SDLR	20 + (7 X # bits shifted)
SDRA	20 + (9 X # bits shifted)
SDLA	20 + (7 X 3 bits shifted)

FORMAT 7 OP CODES - DMØ

OP CODES	# CYCLES	OP CODES	# CYCLES
ROR	1Ø	RORB	9
ROL	1Ø	ROLB	9
TST	1Ø	TSTB	9
ASL	1Ø	ASLB	9
SET	1Ø	SETB	1Ø
CLR	1Ø	CLRB	9
ASR	12	ASRB	11
SWAB	1Ø	SWAD	21
COM	1Ø	COMB	9
NEG	1Ø	NEGB	9
INC	1Ø	INCB	9
DEC	1Ø	DECB	9
IW2	1Ø	LSTS	15
SXT	12	SSTS	1Ø
TCALL	21	ADC	11
TJMP	16	SBC	11

FOR WORD OPS AND:			FOR BYTE OPS AND:		
DM1	ADD	4	DM1	ADD	3
DM2	ADD	4	DM2	ADD	3 *
DM3	ADD	8	DM3	ADD	7
DM4	ADD	6	DM4	ADD	5 *
DM5	ADD	1Ø	DM5	ADD	9
DM6	ADD	1Ø	DM6	ADD	9
DM7	ADD	14	DM7	ADD	13

For DM1 - DM7 and:

CLR subtract 1 cycle
SWAB subtract 1 cycle

*NOTE: Add 2 more if SP or PC.

FORMAT 8 OP CODES

OP CODE	# CYCLES (ASSUMES NO INTERRUPTS)
MBWU	17 + (16 X # words moved)
MBWD	15 + (16 X # words moved)
MBBU	17 + (15 X # bytes moved)
MBBD	15 + (15 X # bytes moved)
MBWA	19 + (16 X # words moved)
MBBA	19 + (15 X # bytes moved)
MABW	19 + (16 X # words moved)
MABB	19 + (15 X # bytes moved)

FORMAT 9 OP CODES - DMØ

OP CODE	# CYCLES
JSR*	22
LEA*	15
ASH	19 if DST = Ø; 22 + (5 X count) if DST>Ø; 25+(7 X count) if DST < Ø.
SOB	1Ø if no branch, 13 if brancn
XCH	23
ASHC	19 if DST = Ø; 38 + (7 X count) if DST>Ø; 38+(9 X count) if DST < Ø
MUL	183
DIV	29 if divisor error, 2Ø2 if no divisor error

*NOTE: DMØ illegal. Used as base figure only.

FOR ALL OP CODES EXCEPT SOB AND:

DM1 add Ø
DM2 add 2
DM3 add 2
DM4 add 2
DM5 add 4
DM6 add 4
DM7 add 8

FORMAT 1Ø OP CODES - SMØ AMD DMØ

OP CODE	# CYCLES
ADD	11
SUB	11
AND	11
BIC	11
BIS	11
XOR	11
CMP	11
BIT	11
MOV	11
CMPB	11
MOVB	12
BISB	11

For SM1: add 3 for word ops, 1 for byte ops.
For SM2: add 4 for word ops, 2 for byte ops. *
For SM3; add 7 for word ops, 5 for byte ops.
For SM4; add 5 for word ops, 3 for byte ops. *
For SM5; add 9 for word ops, 7 for byte ops.
For SM6; add 9 for word ops, 7 for byte ops.
For SM7; add 13 for word ops, 11 for byte ops.

For DM1;add 4 for word ops, 3 for byte ops.
For DM2;add 4 for word ops, 3 for byte ops. *
For DM3;add 8 for word ops, 7 for byte ops.
For DM4;add 6 for word ops, 5 for byte ops. *
For DM5;add 1Øfor word ops, 9 for byte ops.
For DM6;add 1Øfor word ops, 9 for byte ops.
For DM7;add 14for word ops, 13 for byte ops.

For MOVB and DM1-DM7 subtract 1 cycle.

*NOTE: Add 2 if SP or PC

FORMAT 11 OP CODES - ALL ADDRESSING MODES

FADD:	If exponent difference > 39	: 138-145
	Worst Case	: 638
	Typical	: 180-420
FSUB:	If exponent difference > 39	: 141-148
	Worst Case	: 641
	Typical	: 190-430
FMUL:	If either operand = Ø	: 108-111
	Worst Case	: 805
	Typical	: 590-780
FDIV:	If divide by Ø	: 96
	If divide into Ø	: 118
	Worst Case	: 1596
	Typical	: 280-1210
FCMP:		: 49-86