Microsoft CREF-80 CROSS REFERENCE FACILITY

CP/M® Version

Software Reference Manual

for HEATH/ZENITH 8-bit digital computer systems

Table of Contents

CREF-80, Cross Reference Facility

CREF-80
CROSS REFERENCE FACILITY

OVERVIEW

The following section contains reference information about the CREF-80 Cross Reference Facility. The cross reference facility will generate a special listing that can be an important diagnostic tool. Assume, for example, that a program uses a field called FIELD1, and that program testing reveals an error in the manipulating of this field. The cross reference listing can be used to check every instruction that references this field.

USING THE CROSS REFERENCE FACILITY

The Cross Reference Facility is invoked by typing CREF80. To generate a cross reference listing, the Assembler must output a special listing file with embedded control characters. The MACRO-80 command string tells the assembler to output this special listing file. /C is the cross reference switch. When the /C switch is encountered in a MACRO-80 command string, the Assembler opens a .CRF file instead of a .PRN file.

Example:

(NOTE: The asterisk represents the prompt from the Assembler.)

*=TEST/C	Assemble file TEST/MAC and create object file TEST/REL and cross reference file TEST.CRF.
*T,U=TEST/C	Assemble file TEST/MAC and create object file T/REL and cross reference file U.CRF.

When the Assembler is finished, exit to CP/M with CTRL-C. Then call the Cross Reference Facility by typing CREF.

The command string is:

***listing file=source file**

The default extension for the source file is .CRF. the /L switch is ignored, and any other switch will cause an error message to be sent to the terminal.

Possible command strings are:

*=TEST	Examine file TEST.CRF and generate a cross reference listing file TEST.PRN.
*T=TEST	Examine file TEST.CRF and generate a cross reference listing file T.PRN.

Cross Reference listing files differ from ordinary listing files in that:

1. Each source statement is numbered with a Cross Reference number.

2. At the end of the listing, variable names appear in alphabetic order along with the numbers of the lines on which they are referenced or defined. Line numbers on which the symbol is defined are flagged with '#'.

The following example uses the macro assembler, M80, with the cross reference switch. A printout of the cross reference listing is also shown.

```
A>TYPE TEST. MAC
        ORG     100H
ABLE    EQU     10
BAKER   EQU     20
        XRA     A
        LXI     H,STORE
        MVI     B,5
LOOP:   ADD     M
        CPI     ABLE
        JNC     LOOP
        HLT
STORE:  DB      1,2,3,4,5
        END

A>M80

*=TEST/C

No  Fatal error(s)

*↑C
A>CREF

*=TEST

*↑C
A>TYPE TEST.PRN


        MACRO-80 3.4    26-Nov-80       PAGE    1

    1                                           ORG     100H
    2       000A                        ABLE    EQU     10
    3       0014                        BAKER   EQU     20
    4       0100'   AF                          XRA     A
    5       0101'   21 010D'                    LXI     H,STORE
    6       0104'   06 05                       MVI     B,5
    7       0106'   86                  LOOP:   ADD     M
    8       0107'   FE 0A                        CPI     ABLE
    9       0109'   D2 0106'                     JNC     LOOP
   10       010C'   76                          HLT
   11       010D'   01 02 03 04        STORE:  DB      1,2,3,4,5
   12       0111'   05
   13                                           END
```

```
        MACRO-80 3.4    26-Nov-80       PAGE    S

Macros:

Symbols:
ABLE    000A    BAKER   0014    LOOP    0106'   STORE   010D'

No  Fatal error(s)

ABLE        2#      8
BAKER       3#
LOOP        7#      9
STORE       5      11#
```

Appendix A

8080 Op-Codes

INSTRUCTION SET

Mnemonic	Description	Mnemonic	Description
ACI	Add immediate to A with carry	DAA	Decimal adjust A
ADC M	Add memory to A with carry	DAD B	Add B & C to H & L
ADC r	Add register to A with carry	DAD D	Add D & E to H & L
ADD M	Add memory to A	DAD H	Add H & to L to H & L
ADD r	Add register to A	DAD SP	Add stack pointer to H & L
ADI	Add immediate to A	DCR M	Decrement memory
ANA M	And memory with A	DCR r	Decrement register
ANA r	And register with A	DCX B	Decrement B & C
ANI	And immediate with A	DCX D	Decrement D & E
CALL	Call unconditional	DCX H	Decrement H & L
CC	Call on carry	DCX SP	Decrement stack pointer
CM	Call on minus	DI	Disable Interrupt
CMA	Complement A	EI	Enable Interrupts
CMC	Complement carry	HLT	Halt
CMP M	Compare memory with A	IN	Input
CMP r	Compare register with A	INR M	Increment memory
CNC	Call on no carry	INR r	Increment register
CNZ	Call on no zero	INX B	Increment B & C registers
CP	Call on positive	INX D	Increment D & E registers
CPE	Call on parity even	INX H	Increment H & L registers
CPI	Compare immediate with A	INX SP	Increment stack pointer
CPO	Call on parity odd	JC	Jump on carry
CZ	Call on zero	JM	Jump on minus

Mnemonic	Description
JMP	Jump unconditional
JNC	Jump on no carry
JNZ	Jump on no zero
JP	Jump on positive
JPE	Jump on parity even
JPO	Jump on parity odd
JZ	Jump on zero
LDA	Load A direct
LDAX B	Load A indirect
LDAX D	Load A indirect
LHLD	Load H & L direct
LXI B	Load immediate register Pair B & C
LXI D	Load immediate register Pair D & E
LXI H	Load immediate register Pair H & L
LXI SP	Load immediate stack pointer
MVI M	Move immediate memory
MVI r	Move immediate register
MOV M, r	Move register to memory
MOV r,M	Move memory to register
MOV r1, r2	Move register to register
NOP	No-operation
ORA M	Or memory with A
ORA r	Or register with A
ORI	Or immediate with A
OUT	Output
PCHL	H & L to program counter
POP B	Pop register pair B & C off stack
POP D	Pop register pair D & E off stack
POP H	Pop register pair H & L off stack
POP PSW	Pop A and Flags off stack
PUSH B	Push register B & C on stack
PUSH D	Push register pair D & E on stack
PUSH H	Push register pair H & L on stack
PUSH PSW	Push A and Flags on stack

Mnemonic	Description
RAL	Rotate A left through carry
RAR	Rotate A right through carry
RC	Return on carry
RET	Return
RLC	Rotate A left
RM	Return on minus
RNC	Return on no carry
RNZ	Return on no zero
RP	Return on positive
RPE	Return on parity even
RPO	Return on parity odd
RRC	Rotate A right
RST	Restart
RZ	Return on zero
SBB M	Subtract memory from A with borrow
SBB r	Subtract register from A with borrow
SBI	Subtract immediate from A with borrow
SHLD	Store H & L direct
SPHL	H & L to stack pointer
STA	Store A direct
STAX B	Store A indirect
STAX D	Store A indirect
STC	Set carry
SUB M	Subtract memory from A
SUB r	Subtract register from A
SUI	Subtract immediate from A
XCHG	Exchange D & E, H & L Registers
XRA M	Exclusive Or memory with A
XRA r	Exclusive Or register with A
XRI	Exclusive Or immediate with A
XTHL	Exchange top of stack, H & L

Appendix B

Z80 Op-Codes

INSTRUCTION SET

Mnemonic	Description
ADC HL, ss	Add with Carry Reg. pair ss to HL
ADC A, s	Add with carry operand s to Acc.
ADD A, n	Add value n to Acc.
ADD A, r	Add Reg. r to Acc.
ADD A, (HL)	Add location (HL) to Acc.
ADD A, (IX+d)	Add location (IX+d) to Acc.
ADD A, (IY+d)	Add location (IY+d) to Acc.
ADD HL, ss	Add Reg. pair ss to HL
ADD IX, pp	Add Reg. pair pp to IX
ADD IY, rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b, (HL)	Test BIT b of location (HL)
BIT b, (IX+d)	Test BIT b of location (IX+d)
BIT b, (IY+d)	Test BIT b of location (IY+d)
BIT b, r	Test BIT b of Reg. r
CALL cc, nn	Call subroutine at location nn if condition cc is true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B=0
EI	Enable interrupts
EX (SP), HL	Exchange the location (SP) and HL
EX (SP), IX	Exchange the location (SP) and IX
EX (SP) IY	Exchange the location (SP) and IY
EX AF, AF	Exchange the contents of AF and AF'
EX DE, HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
HALT	HALT (wait for interrupt or reset)

Mnemonic	Description
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A, (n)	Load the Acc. with input from device n
IN r, (C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C), and increment HL and decrement B.
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditional Jump to (IY)
JP cc, nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JP C, e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JP NC, e	Jump relative to PC+e if carry=0
JR NZ, e	Jump relative to PC+e if non zero (Z=0)
JR Z, e	Jump relative to PC+e if zero (Z=1)

Mnemonic	Description
LD A, (BC)	Load Acc. with location (BC)
LD A, (DE)	Load Acc. with location (DE)
LD A, I	Load Acc. with I
LD A, (nn)	Load Acc. with location nn
LD A, R	Load Acc. with Reg. R
LD (BC), A	Load location (BC) with Acc.
LD (DE), A	Load location (DE) with Acc.
LD (HL), n	Load location (HL) with value n
LD dd, nn	Load Reg. pair dd with value nn
LD HL, (nn)	Load HL with location (nn)
LD (HL), r	Load location (HL) with Reg. r
LD I, A	Load I with Acc.
LF IX, nn	Load IX with value nn
LD IX, (nn)	Load IX with location (nn)
LD (IX+d), n	Load location (IX+d) with value n
LD (IX+d), r	Load location (IX+d) with Reg. r
LD IY, nn	Load IY with value nn
LD IY, (nn)	Load IY with location (nn)
LD (IY+d), n	Load location (IY+d) with value n
LD (IY+d), r	Load location (IY+d) with Reg. r
LD (nn), A	Load location (nn) with Acc.
LD (nn), dd	Load location (nn) with Reg. pair dd
LD (nn), HL	Load location (nn) with HL
LD (nn), IX	Load location (nn) with IX
LD (nn), IY	Load location (nn) with IY
LD R, A	Load R with Acc.
LD r, (HL)	Load Reg. r with location (HL)
LD r, (IX+d)	Load Reg. r with location (IX+d)
LD r, (IY+d)	Load Reg. r with location (IY+d)
LD r, n	Load Reg. r with value n
LD r, r'	Load Reg, r with Reg. r'
LD SP, HL	Load SP with HL
LD SP, IX	Load SP with IX
LD SP, IY	Load SP with IY

Mnemonic	Description
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC, repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' or operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C), r	Load output port (C) with Reg. r
OUT (n), A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B
POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b, m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circular
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A, s	Subtract operand s from Acc. with carry
SBC HL, ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b, (HL)	Set Bit b of location (HL)
SET b, (IX+d)	Set Bit b of location (IX+d)
SET b, (IY+d)	Set Bit b of location (IY+d)
SET b, r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

Appendix C

ASCII Codes

DECIMAL TO OCTAL TO HEX TO ASCII CONVERSION

I				II				III				IV			
DEC	OCT	HEX	ASCII	DEC	OCT	HEX	ASCII	DEC	OCT	HEX	ASCII	DEC	OCT	HEX	ASCII
0	000	00	NUL	32	040	20	SPACE	64	100	40	@	96	140	60	'
1	001	01	SOH	33	041	21	!	65	101	41	A	97	141	61	a
2	002	02	STX	34	042	22	"	66	102	42	B	98	142	62	b
3	003	03	ETX	35	043	23	#	67	103	43	C	99	143	63	c
4	004	04	EOT	36	044	24	$	68	104	44	D	100	144	64	d
5	005	05	ENQ	37	045	25	%	69	105	45	E	101	145	65	e
6	006	06	ACK	38	046	26	&	70	106	46	F	102	146	66	f
7	007	07	BEL	39	047	27	'	71	107	47	G	103	147	67	g
8	010	08	BS	40	050	28	(	72	110	48	H	104	150	68	h
9	011	09	HT	41	051	29	)	73	111	49	I	105	151	69	i
10	012	0A	LF	42	052	2A	*	74	112	4A	J	106	152	6A	j
11	013	0B	VT	43	053	2B	+	75	113	4B	K	107	153	6B	k
12	014	0C	FF	44	054	2C	,	76	114	4C	L	108	154	6C	l
13	015	0D	CR	45	055	2D	-	77	115	4D	M	109	155	6D	m
14	016	0E	SO	46	056	2E	PERIOD	78	116	4E	N	110	156	6E	n
15	017	0F	SI	47	057	2F	/	79	117	4F	O	111	157	6F	o
16	020	10	DLE	48	060	30	0	80	120	50	P	112	160	70	p
17	021	11	DC1	49	061	31	1	81	121	51	Q	113	161	71	q
18	022	12	DC2	50	062	32	2	82	122	52	R	114	162	72	r
19	023	13	DC3	51	063	33	3	83	123	53	S	115	163	73	s
20	024	14	DC4	52	064	34	4	84	124	54	T	116	164	74	t
21	025	15	NAK	53	065	35	5	85	125	55	U	117	165	75	u
22	026	16	SYN	54	066	36	6	86	126	56	V	118	166	76	v
23	027	17	ETB	55	067	37	7	87	127	57	W	119	167	77	w
24	030	18	CAN	56	070	38	8	88	130	58	X	120	170	78	x
25	031	19	EM	57	071	39	9	89	131	59	Y	121	171	79	y
26	032	1A	SUB	58	072	3A	:	90	132	5A	Z	122	172	7A	z
27	033	1B	ESC	59	073	3B	;	91	133	5B	[	123	173	7B	{
28	034	1C	FS	60	074	3C	<	92	134	5C	\	124	174	7C	|
29	035	1D	GS	61	075	3D	=	93	135	5D	]	125	175	7D	}
30	036	1E	RS	62	076	3E	>	94	136	5E	^	126	176	7E	~
31	037	1F	US	63	077	3F	?	95	137	5F	_	127	177	7F	DELETE

CONTROL CHARACTER DEFINITIONS

NUL	Null: Tape feed,
SOH	Start of Heading; Start of Message
STX	Start of Text; End of Address
ETX	End of Text; End of Message
EOT	End of Transmission; Shuts off TWX machines
ENQ	Enquiry; WRU
ACK	Acknowledge; RU
BEL	Rings Bell
BS	Backspace
HT	Horizontal TAB
LF	Line Feed or Space (New Line)
VT	Vertical TAB
FF	Form Feed (PAGE)
CR	Carriage Return
SO	Shift Out
SI	Shift In
DLE	Data Link Escape
DC1	Device Control 1; Reader on
DC2	Device Control 2; Punch on
DC3	Device Control 3; Reader off
DC4	Device Control 4; Punch off
NAK	Negative Acknowledge; Error
SYN	Synchronous Idle (SYNC)
ETB	End of Transmission Block; Logical End of Medium
CAN	Cancel (CANCL)
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	Unit Separator

Refer to the chart on page C-1. Note that any print control character defined above and listed in column I of the chart can be produced from the combination of CTRL and the alphabetical character in column III or IV which is on the same line and to the right of the print control character. That is, DLE is CTRL-P or ↑P, BEL is CTRL-G or ↑G, and so on.

Appendix D

Error Messages

MACRO-80 ERROR MESSAGES

MACRO-80 errors are indicated by a one-character flag in column one of the listing file. If a listing file is not being printed on the terminal, each erroneous line is also printed or displayed on the terminal. Below is a list of the MACRO-80 Error Codes:

Error Codes

A Argument error —
Argument to pseudo-op is not in correct format or is out of range (.PAGE 1; .RADIX 1; PUBLIC 1; STAX H; MOV M,N; INX C).

C Conditional nesting error —
ELSE without IF, ENDIF without IF, two ELSEs on one IF.

D Double Defined symbol —
Reference to a symbol which is multiply defined.

E External error —
Use of an external illegal in context (e.g., FOO SET NAME ;
MVI A,2-NAME).

M Multiply-Defined symbol —
Definition of a symbol which is multiply-defined.

N Number error —
Error in a number, usually a bad digit (e.g., 8Q).

O Bad opcode or objectionable syntax —
ENDM, LOCAL outside a block; SET, EQU or MACRO without a name; bad syntax in an opcode (MOV A:); or bad syntax in an expression (mismatched parenthesis, quotes, consecutive operators, etc.).

P Phase error —
Value of a label or EQU name is different on pass 2.

Q Questionable —
Usually means a line is not terminated properly. This is a warning error (e.g., MOV A,B,).

R Relocation —
Illegal use of relocation in expression, such as abs-rel. Data, code and COMMON areas are relocatable.

U Undefined symbol —
A symbol referenced in an expression is not defined. (For certain pseudo-ops, a V error is printed on pass 1 and a U on pass 2.)

V Value error —
On pass 1 a pseudo-op which must have its value known on pass 1 (e.g., .RADIX, .PAGE, DS, IF, IFE, etc.), has a value which is undefined later in the program, a U error will not appear on the pass 2 listing.

Error Messages:

'No end statement encountered on input file'
No END statement: either it is missing or it is not parsed due to being in a false conditional, unterminated IRP/IRPC/REPT block or terminated macro.

'Unterminated conditional'
At least one conditional is unterminated at the end of the file.

'Unterminated REPT/IRP/IRPC/MACRO'
At least one block is unterminated.

[xx] [No] Fatal error(s) [,xx warnings]
The number of fatal errors and warnings. The message is listed on the console and in the list file.

LINK-80 ERROR MESSAGES

?No Start Address

A /G switch was issued, but no main program had been loaded.

?Loading Error

The last file given for input was not a properly formatted LINK-80 object file.

?Out of Memory

Not enough memory to load program.

(A minimum of 40K RAM is required.)

?Command Error

Unrecognizable LINK-80 command string.

?<file> Not Found

<file>, as given in the command string, did not exist.

%2nd COMMON Larger /XXXXXX/

The first definition of COMMON block /XXXXXX/ was not the largest definition. Re-order module loading sequence or change COMMON block definitions. (See Chapter 9 in the FORTRAN Reference Manual for more information on the COMMON statement.)

%Mult. Def. Global YYYYYY

More than one definition for the global (internal) symbol YYYYYY was encountered during the loading process.

%Overlaying Program Area
Data

A /D or /P will cause already loaded data to be destroyed.

?Intersecting Program Area
Data

The program and data area intersect and an address or external chain entry is in this intersection. The final value cannot be converted to a current value since it is in the area intersection.

?Start Symbol — <name> — Undefined

After a /E: or /G: is given, the symbol specified was not defined.

Origin Above Loader Memory, Move Anyway (Y or N)?
Below

After a /E or /G was given, either the data or program area has an origin or top which lies outside loader memory. If a Y <cr> is given, LINK-80 will move the area and continue. If anything else is given, LINK-80 will exit.

In either case, if a /N was given, the image will already have been saved.

?Can't Save Object File

A disk error occurred when the file was being saved. Usually this occurs when there is no more room left on the disk.

?Nothing Loaded

A <filename>/S or /E or /G was given but no object file was loaded. That is, an attempt was made to search a library, exit the Linker, or execute a program, when in fact nothing had been loaded.