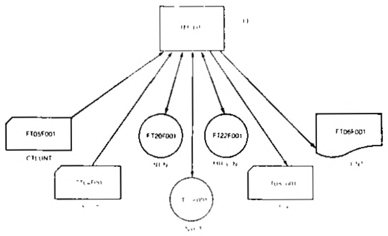
INPUT AND OUTPUT WIDTHS FOR FORMATS

FORMAT#	FORMAT	WIDTH (BITS)
1	Hexadecimal #1 (PROM)	8
2	Hexadecimal #2 (ROM)	8
3	BNPF	2 or 4 or 8 or 16
4	271 & 371 ROM/HILO of Prototyping System	4 or 8
5	TMS8080/TMS1000 Absolute Object from Loader / Simulator	8
6	TMS1000 Object from Assembler	8
7	TMS1000 Listed Absolute Object	8
8	TMS1000 OPLA Data	8 or 16
9	TMS9900 Standard Absolute Object	16
10	TMS9900 Compressed Absolute Object	16
11	TI4700 ROM	8
12	TI4800 ROM	4 or 8

FILES DEFINITIONS & DESCRIPTIONS



CTLUNT	—	Input file for control cards
INUNT	—	Input file for data
INTIN	—	Intermediate file for storage of input data. It must be a rewindable file with a logical record length of 80 bytes.
INTOT	—	Intermediate file for storage of internal data. It must be a rewindable file with a logical record length of 80 bytes
OTUNT	—	Output file for translated data
LSTUNT	—	Print file for listing of data and error messages
MRGUNT	—	Intermediate file for storage of internal data It must be a rewindable file with a logical record length of 80 bytes

TEXAS INSTRUMENTS

Cross Support Reference Data

Microprocessor Series™

The Cross Assembler data base which is assigned to PUNIT, is read by the FORTRAN program as the first file at execution time. It is the actual Cross Assembler program written in internal code, and it is suggested that it be assigned to a permanent disk file

[illegible]	[illegible]	DEVICE TYPE		FUNCTION
IUNIT	5	CR,CS MT,DF	80	TMS 9900 Source Input
LUNIT	6	CS,MT	80	Listing Output
OUNIT	7	CS,MT	80	TMS9900 Object Dutput
SUNIT	10	MT,DF	80	Assembly Scratch
PUNIT	11	CR,CS	80	Data Base INPUT

CR—CARD READER, CS—(E TAPE, MT—MAGNETIC TAPE, DF—DISKFILE, CP—CARD PUNCH, LP—LINE P

CROSS ASSEMBLER SYSTEM FILES

AORG places the expression value in the location counter, and defines the succeeding locations as absolute

ABSOLUTE ORIGIN

AORG

Syntax Definition

[< label -]ƀ AORGƀ · wd-exp ·ƀ [comment ·]

RORG places the expression value in the location counter and defines the succeeding locations as relocatable

RELOCATABLE ORIGIN

RORG

Syntax Definition

[<label>ƀ RORGƀ [<exp>]ƀ [<comment>]

DORG places the expression value in the location counter, and defines the succeeding locations as a dummy section No object code is generated in a dummy section

DUMMY ORIGIN

DORG

Syntax Definition

<label>ƀ . DORGƀ . <exp>ƀ .[<comment>]

BSS first assigns the label, if present, and increments the location counter by the value of the expression

BLOCK STARTING WITH SYMBOL

BSS

Syntax Definition

[<label>]ƀ . BSSƀ <wd-exp>ƀ [<comment>]

BSS first increments the location counter by the value of the expression, and then assigns the label, if present

BLOCK ENDING WITH SYMBOL

BES

Syntax Definition

[<label>]ƀ BESƀ <wd-exp>ƀ [<comment>]

EQU assigns an assembly-time constant to the label

DEFINE ASSEMBLY-TIME CONSTANT

EQU

Syntax Definition

<label>ƀ EQUƀ <exp>ƀ [<comment>]

EVEN first assigns the label, if present, and then aligns the location counter on a word boundary (even address)

WORD BOUNDARY

EVEN

Syntax Definition

[<label>]ƀ EVENƀ [<comment>]

OPTIONS allows cross referencing when XREF is specified, and allows printing of the symbol table when SYMT is present

OUTPUT OPTIONS

OPTION

Syntax Definition

ƀ OPTIONƀ <keyword>[,<keyword>] ƀ [<comment>]

IDT assigns a name to the program, and must precede any code-generating directive or instruction

PROGRAM IDENTIFIER

IDT

Syntax Definition:

[<label>]ƀ . . IDTƀ <string>ƀ [<comment>]

TITL supplies a string to be printed at the top of each subsequent source listing page

PAGE TITLE

TITL

Syntax Definition.

[<label>]ƀ TITLƀ <string>ƀ [<comment>]

LIST restores printing of the source listing

LIST SOURCE

LIST

Syntax Definition.

[<label>]ƀ LISTƀ [<comment>]

UNL inhibits printing of the source listing

NO SOURCE LIST

UNL

Syntax Definition.

[<label>]ƀ . . UNLƀ . . . [<comment>]

PAGE directs the assembler to continue the source listing on the next page

PAGE EJECT

PAGE

Syntax Definition

[· label ·]ƀ PAGEƀ [comment ·]

BYTE places expressions in successive bytes, optionally assigning the label the aodress of the first byte

INITIALIZE BYTE

BYTE

Syntax Definition

[<label>]ƀ . . . BYTEƀ . . <exp>[,<exp>] ƀ [<comment>]

DATA places expressions in successive words, optionally assigning the label the address of the first word

INITIALIZE WORD

DATA

Syntax Definition

[<label>]ƀ DATAƀ . <exp>[,<exp>] ƀ [<comment>]

TEXT places characters in successive bytes, arithmetically negating the last character, and optionally assigns the label the address of the first character

INITIALIZE TEXT

TEXT

Syntax Definition

[<label>]ƀ TEXTƀ [–]<string>ƀ [<comment>]

DEF makes symbols available to other programs as external references

EXTERNAL DEFINITION

DEF

Syntax Definition

[<label>]ƀ DEFƀ <symbol>[,<symbol>] . ƀ [<comment>]

REF directs the assembler to look externally for symbols

EXTERNAL REFERENCE

REF

Syntax Definition

[<label>]ƀ REFƀ <symbol>[,<symbol>] ƀ [<comment>]

DXOP assigns an extended operation to a symbol

DEFINE EXTENDED OPERATIONS

DXOP

Syntax Definition

[· label ·]ƀ DXOPƀ · symbol term ·ƀ [· comment]

END terminates the assembly

PROGRAM END

END

Syntax Definition

[<label>]ƀ ENDƀ [<symbol>]ƀ [<comment>]

NOP places a no-operation code in the object file

NO OPERATION

NOP

Syntax Definition.

[<label>]ƀ NOPƀ [<comment>]

RT assembles as a return from subroutine by substituting a branch through register 11

RETURN

RT

Syntax Definition

[<label>]ƀ RTƀ [<comment>]

SIMULATOR FILES

INTERNAL NAME	DEFAULT UNIT	DEVICE TYPE		FUNCTION	
INCOPY	4	MT,DF	80	Batch copy file	C
INCOM	5	TE,CR MT DF	80	Simulation command	C
OUTPRT OUTTRC	6	MT,DF TE,CR	80 or 136	Listing output	L,C R
INLOD	10	TE,CR MT,DF	80	Linker commands	L
OUTCOM	11	TE,LP	80 or 136	Prompts and error msg for linker output	L
OUTSAV	17	MT,CP DF	80	Absolute object	L,S
INSCR	20	MT,DF	136	Input scratch file	C,R,S
OUTSCR	21	MT,DF	136	Output scratch file	L,C,R

Device type legend
TE—terminal, CR—card reader, MT—magnetic tape, DF—disk file, CP—card punch

Where used legend
L—link processor, C—command processor; R—run processor, S-save processor

In to the above unit number assignments, the user must also assign unique FC logical unit numbers to each TMS9900 object code module to be included in the rocessor

SIMULATOR DIRECTIVES

ORIGIN COMMAND The "ORIGIN" command can be used to specify where relocatable code is to be loaded.

ORIGIN hex-number

INCLUDE COMMAND The "INCLUDE" command directs the loader to load an object module from a data set (e.g , card reader, disc, tape) The data set must be a sequential d may contain one or more object modules At least one "INCLUDE" hould be used in the LINK processor command stream The format for the s as follows

INCLUDE n

ENTRY COMMAND The "ENTRY" command specifies the program entry point to the loader. The format for the command is as follows

ENTRY name

SUMMARY OF CONTROL LANGUAGE STATEMENTS

The formats of the control statements for the "COMMAND" processor are shown below, with a brief description following

[label] $\begin{Bmatrix}\text{R}\\\text{RUN}\end{Bmatrix}$ [*] $\begin{Bmatrix}\text{F}\\\text{FOR}\end{Bmatrix}$ n $\left[\begin{Bmatrix}\text{FR}\\\text{FROM}\end{Bmatrix} \text{i1}\right]\left[\begin{Bmatrix}\text{T}\\\text{TO}\end{Bmatrix}\right]$i2 [,label]	Specifies where to start and stop simulation Control passes to statement at label operand when a breakpoint occurs
[label] $\begin{Bmatrix}\text{T}\\\text{TRACE}\end{Bmatrix}$ [list]	Specifies locations to be traced
[label] $\begin{Bmatrix}\text{NOT}\\\text{NOTRACE}\end{Bmatrix}$ [list]	Disables trace for specified locations
[label] $\begin{Bmatrix}\text{RE}\\\text{REFER}\end{Bmatrix}$ [list]	Specifies locations for reference breakpoint
[label] . =ER [list]	Disables reference breakpoint at specified locations
[label] $\begin{Bmatrix}\text{A}\\\text{ALTER}\end{Bmatrix}$ [list]	Specifies locations for alteration breakpoint
[label] $\begin{Bmatrix}\text{NOA}\\\text{NOALTER}\end{Bmatrix}$ [list]	Disables alteration breakpoint at specified locations
[label] $\begin{Bmatrix}\text{P}\\\text{PROTECT}\end{Bmatrix}$ [list]	Specifies areas for memory protection.
[label] IF (logical expression) label	Conditional transfer of control program
[label] $\begin{Bmatrix}\text{J}\\\text{JUMP}\end{Bmatrix}$ label	Unconditional transfer of control program
[label] $\begin{Bmatrix}\text{TI}\\\text{TIME}\end{Bmatrix}$ [n]	Prints the value of 9900 time and optionally sets a new value
[label] $\begin{Bmatrix}\text{D}\\\text{DISPLAY}\end{Bmatrix}$ [D] $\begin{Bmatrix}\text{CP}\\\text{CPU}\end{Bmatrix}$ [register list]	Prints contents of registers
[label] $\begin{Bmatrix}\text{D}\\\text{DISPLAY}\end{Bmatrix}\begin{bmatrix}\text{D}\\\text{C}\end{bmatrix}\begin{Bmatrix}\text{M}\\\text{MEMORY}\end{Bmatrix}$ list	Prints contents of memory
[label] $\begin{Bmatrix}\text{D}\\\text{DISPLAY}\end{Bmatrix}\begin{Bmatrix}\text{S}\\\text{SYMBOL}\end{Bmatrix}\begin{bmatrix}\text{\$}\\\text{symbol}\\\text{number}\end{bmatrix}$	Prints values from symbol table
[label] $\begin{Bmatrix}\text{D}\\\text{DISPLAY}\end{Bmatrix}\begin{Bmatrix}\text{CR}\\\text{CRU}\end{Bmatrix}\begin{Bmatrix}\text{I}\\\text{INPUT}\\\text{O}\\\text{OUTPUT}\end{Bmatrix}$ list	Prints CRU values
[label] $\begin{Bmatrix}\text{S}\\\text{SET}\end{Bmatrix}\begin{Bmatrix}\text{C}\\\text{CPU}\end{Bmatrix}$ register-value list	Places values into registers
[label] $\begin{Bmatrix}\text{S}\\\text{SET}\end{Bmatrix}\begin{Bmatrix}\text{M}\\\text{MEMORY}\end{Bmatrix}$ location-value list	Places values into memory
[label] $\begin{Bmatrix}\text{S}\\\text{SET}\end{Bmatrix}\begin{Bmatrix}\text{I}\\\text{INT}\end{Bmatrix}$ level, n_1 [,n_2,n_3]	Sets up one or more interrupts
[label] $\begin{Bmatrix}\text{E}\\\text{END}\end{Bmatrix}$	Disables breakpoints and traces, and initializes simulation Passes control to next control statement
[label $\begin{Bmatrix}\text{I}\\\text{INPUT}\end{Bmatrix}\begin{Bmatrix}n_1 \overset{n}{\text{TO}} n_2\end{Bmatrix}\begin{bmatrix}\text{F}\\\text{FIRST}\\\text{L}\\\text{LAST}\\\text{A}\\\text{ALL}\end{bmatrix}$ [data]	Defines input lines and fields, and supplies data for program being simulated
[label] $\begin{Bmatrix}\text{O}\\\text{OUTPUT}\end{Bmatrix}\begin{Bmatrix}n_1 \overset{n}{\text{TO}} n_2\end{Bmatrix}$	Defines output lines and fields, or prints output of program being simulated
[label] ECT list	Connects input CRU lines to output CRU lines
[label] $\begin{Bmatrix}\text{C}\\\text{CONVERT}\end{Bmatrix}$ expression list	Evaluates and prints values of expressions in decimal and hexadecimal form
$\begin{Bmatrix}\text{B}\\\text{BATCH}\end{Bmatrix}$	Specifies batch mode
[label] $\begin{Bmatrix}\text{L}\\\text{LDAD}\end{Bmatrix}$	Loads Wp and PC from locations $FFFC_{16}$ and $FFFE_{16}$
[label] $\begin{Bmatrix}\text{CL}\\\text{CLOCK}\end{Bmatrix}$ t	Specify clock period
[label] $\begin{Bmatrix}\text{M}\\\text{MEMORY}\end{Bmatrix}\begin{Bmatrix}\text{RA}\\\text{RAM}\\\text{RO}\\\text{ROM}\end{Bmatrix}\begin{Bmatrix}\text{R}\\\text{READ}\end{Bmatrix} = n_1 \left[\begin{Bmatrix}\text{W}\\\text{WRITE}\end{Bmatrix} = n_2\right]$ list	Define available memory Default is 32K RAM
[label] $\begin{Bmatrix}\text{SA}\\\text{SAVE}\end{Bmatrix}$	Create absolute object module
(label] $\begin{Bmatrix}\text{W}\\\text{WIDTH}\end{Bmatrix}$ n	Specifies number of columns available for printing

MONITOR COMPLETION CODES

The simulator signals completion by executing and writing an appropriate STOP I statement, where I takes on one of the following values

CODE	MEANING
0	Normal completion
1	Abnormal completion from LNKPRC
2	Premature EOF —If this error occurs it indicates that a premature EOF was encountered while attempting to reposition the BATCH command file
3	Internal error; invalid label value
4	Roll memory overflow
5	Loader error —If this error occurs it means an attempt was made to load an object file into simulated memory and it failed causing termination of the link processor
8	Abnormal completion from LOADER
9	Abnormal completion from CMDPRC
99	Internal error —Illegal completion from CMDPRC Internal error
999	Internal error —Illegal parameter passed to WRITER

If an error of 99 or 999 results, an internal error has occurred and the error should be reported to TEXAS INSTRUMENTS INC

LINK PROCESSOR ERRORS

CODE	MESSAGE
L01	Load not completed
L02	Multiply defined external symbol (name)
L03	Empty object file on unit
L04	Attempt to load undefined memory
L05	Tag D follows tag 0
L06	Invalid tag character
L09	Undefined external memory
L13	Empty memory on save
L14	(name) not in external symbol table
L18	Maximum memory size exceeded
L19	Missing end
L21	Checksum error (computed value)
L22	Odd origin value specified—even value used
L24	Ref chain loop
L25	Object module does not start with tag 0
L26	Odd value (value) specified for tag (tag) even value used
L27	Missing F tag in record (number)
L28	Bad REF chain for (name)
L29	Bad object format in object module
L30	Illegal hex digit in field (digit)

COMMAND PROCESSOR ERRORS

CODE			CODE		
NUMBER	NAME	MESSAGE	NUMBER	NAME	MESSAGE
1	BADCHR	Bad character	18	RANGE	Range error
2	BADCMD	Unrecognizable command	19	SYNTAX	Syntax error
3	BADIGT	Bad digit	20	TOOMNY	Too many values
4	BADMOD	Bad module name	21	UNDEF	Undefined symbol
5	BADREG	Bad register mnemonic			
6	BADVAL	Bad value			
7	CRUSPC	CRU specification error			
8	FLDCNT	Too few/many fields			
9	HITEDF	Hit EOF			
10	HITEOL	Hit end-of-line			
11	MEMDEF	Undefined			
12	MISSEQ	Missing equal sign			
13	NODATA	No data found			
14	NORDL	No data rolls available			
15	NOSET	Set not performed			
16	NOTIMP	Command not implemented			
17	ORDER	Command out of order			

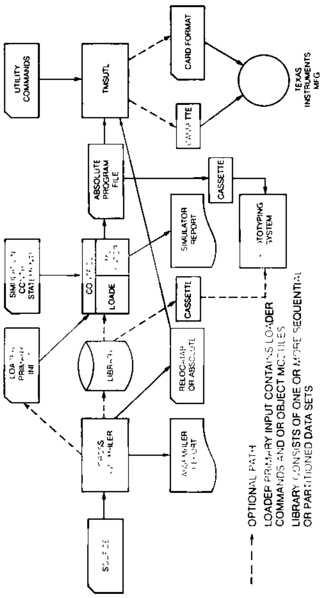
RUN PROCESSOR ERRORS

CODE	MESSAGE
1	PC interrupt vector entry in undefined memory
2	WP interrupt vector entry in undefined memory
3	Register out of address space (WP 65502)
4	Registers in undefined memory
5	Registers in ROM
6	PC interrupt vector refer breakpoint
7	WP interrupt vector refer breakpoint
8	Register alter breakpoint
9	Register protect breakpoint
10	Register refer breakpoint
11	Undefined opcode
12	Undefined memory reference
13 14	Unused
15	PC refer breakpoint
16	Unimplemented opcode
17,18,19	Unused
20	Destination address in undefined memory
21	Destination refer breakpoint
22	Destination alter breakpoint
23	Destination ROM breakpoint
24	Unused
25	Source address in undefined memory
26	Source refer breakpoint
27	Source alter breakpoint
28	Source ROM breakpoint

TMSUTL

CONCEPT

TMSUTL is a general purpose utility program that accepts as input TI microprocessor object format, PROM manufacturing formats, or ROM manufacturing formats This data is syntax checked, output options are gathered, the input data converted and an output file is produced



INPUT, OUTPUT CONTROL CARD FORMATS

GENERAL DESCRIPTION

INPUT frmt [addr1 addr2] [WIDTH = x] [PARTITION = y]

frmt	—	is tne format number (integer 1-12)
addr1	—	is the starting address where input data is to be stored
addr2	—	is the maximum address where data is to be stored
x	—	is the bit width of the input words
y	—	is the number of input data set partitions 1 Y 4

OUTPUT num addr1 addr2 WIDTH = x PARTITION = y

num	—	is the format number (integer 1-12)
addr1	—	is the minimum address to be output
addr2	—	is the maximum address to be output
x	—	is the bit width of an output word
y	—	

EOF—End of COMMAND FILE indicator

AVAILABLE FORMATS

FORMAT #	FORMAT	INPUT	OUTPUT
1	Hexadecimal #1 (PROM)	X	X
2	Hexadecimal #2 (ROM)	X	X
3	BNPF	X	X
4	271 & 371 ROM/HILO of prototyping System	X	X
5	• TMS [illegible] bsolute Object from SIM1 [illegible] ader/Simulator	X	X
6	TMS1000 Absolute ROM Object from Assembler	X	X
7	TMS1000 Listed Absolute Object	X	X
8	TMS1000 OPLA Data	X	
9	TMS9900 Standard Absolute Object of Cross Support System (Assembler or Loader/Simulator) & Prototyping System	X	X
10	TMS9900 Compressed Absolute Object of Prototyping System	X	X
11	TI4700 ROM	X	X
12	TI4800 ROM	X	X

TMSUTL FORMAT PATHS

	Output Format ⟶	1	2	3	4	5	6	7	8	9	10	11	12
1)	Hexadecimal #2 (PROM)	YES	YES	YES	YES	NO	NO	YES	NO	NO	NO	YES	YES
2)	[illegible] lecimal #2	YES	YES	YES	YES	NO	NO	YES	NO	NO	NO	YES	YES
3)	[illegible]	YES	YES	YES	YES	YES	YES	YES	NO	YES	YES	YES	YES
4)	271 & 371 ROM/ HILO of Prototyping	YES	YES	YES	YES	NO	NO	YES	NO	NO	NO	YES	YES
5)	TMS1000 / TMS8080 Absolute Object from L[illegible] nulator	YES	YES	YES	YES	YES	YES	YES	NO	NO	NO	YES	YES
6)	T[illegible] Absolute ROM Objects from Assembler for [illegible]	YES	YES	YES	YES	YES	YES	YES	NO	NO	NO	YES	YES
7)	T[illegible] Listed A[illegible]	YES	YES	YES	YES	YES	YES	YES	NO	NO	NO	YES	YES
8)	T[illegible] Data	YES	YES	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
9)	TMS9900 Standard Absolute Object of Cross Support System (Assembler or Loader/Simulator) & Prototyping System	YES	YES	YES	YES	NO	NO	NO	NO	YES	YES	YES	YES
10)	TMS9900 Compressed Absolute Object of Protoyping	YES	YES	YES	YES	NO	NO	NO	NO	YES	YES	YES	YES
11)	[illegible] ROM	YES	YES	YES	YES	YES	NO	YES	NO	NO	NO	YES	YES
12)	TI4800 ROM	YES	YES	YES	YES	YES	NO	YES	NO	NO	NO	YES	YES

DATA DELIMITERS

The following is a table of data delimiters or end-of-module records for Input Data

FORMAT #	TYPES
1. Hex format 1	End of file record (:00)
2. Hex format 2	Trailer record — "END OF TEXT" (hollerith code 12-9-3) character followed by 79 non-blank characters (without asterisks)
3 BNPF	End of file record ($ in column 1)
4. 271/371 ROM and HILO of Prototyping System	End of file record ($END)
5. TMS8080/TMS1000 Absolute Object from Loader/Simulator	End record (+ END)
6 TMS1000 Absolute ROM Object	End of file record ($END)
7 TMS1000 Listed Absolute Dbject	End of file record ($END)
8 TMS1000 OPLA Data	End of file record ($END)
9. TMS9900 Standard Absolute Object	End of module record ( )
10 TMS9900 Binary Compressed Absolute Object	End of file record ($END)
11 TI4700 ROM	End of file record ($END)
12 TI4800 ROM	End of file record ($END)

ADDRESS RANGES FOR FORMATS

FORMAT#	FORMAT	ADDRESS RANGE
1	Hexadecimal # 1 (PROM)	$(0\text{-FFFF})_H$
2	Hexadecimal # 2 (ROM)	None
3	BNPF	None
4	271 & 371 ROM/HILO of Prototyping System	None
5	TMS8080/TMS1000 Absolute Object from Loader/Simulator	(0-255)
6	TMS1000 Absolute ROM Object	$(0\text{-}800)_H$
7	TMS1000 Listed Absolute Object	$(0\text{-1 Chapter 0-15 page 0-3F location})_H$
8	TMS1000 OPLA Data	$(0\text{-1F})_H$
9	TMS9900 Standard Absolute Object	$(0\text{-FFFF})_H$
10	TMS9900 Compressed Absolute Object	$(0\text{-FFFF})_H$
11	TI4700 ROM	$(0\text{-}400)_H$
12	TI4800 ROM	$(0\text{-}400)_H$