DataBoard

070-8000

Assembler, ESTAB, TRACE. Development Kit

ASSEMBLER working under the operating system OS.8MT, ia s powerful language equally suitable for both technical and administrative applications i.e. calculation, process control (the code can be placed in PROM), data management, etc. ESTAB is a powerful linker with capabilities for linking modules written in different languages. TRACE provides sophisticated debugging with many options.

OS.8MT — ORIENTATION

Multi-user Several users working at the same time is a function in OS.8MT. The users can work with both program development and application/ execution at the same time.

Multitasking Supervised by OS.8MT several programs can run at the same time. More than one task can be started from the same terminal device. This makes it possible to, for example, control a machine at the same time as running a data base.

Real time All execution is done in a real time environment, with the possibilities of both time-sharing and giving tasks different priorities. A calender and a clock are included in OS.8MT. Tasks can also be time-synchronised.

INPUT/OUTPUT

Peripherals All I/O is totally independent of the physical characteristics of each device. Any combination of devices can be used as every type is controlled by driver routines contained in OS.8MT.
The most common types of peripherals are:

Terminal devices:	Intelligent and non-intelligent.
Printers:	Character and line oriented.
Mass-storage:	Floppy discs, winchesters, hard discs and magnetic tape
Modems:	Synchronous and asynchronous.
Process control:	Analogous and digital inputs and outputs.

Filing system Totally independent of a chosen combination of mass-storage devices i.e. floppy disc, winchester, etc. Every disc has its own name, independent of the drives. One master file directory and any number of element file directories exist on each volume.
Files and space for files are allocated and deallocated in a totally dynamic way.

Files Every file-type can be reached from ASSEMBLER.
The record length can be fixed or variable between 1 and 65535 characters and the files can consist of 1 to 16777215 records. The files can be continuous with fixed size or indexed with variable size. Three access methods exist: on byte, record or sector level.

Data bases Data bases are managed by an ISAM program, which permits several users to work with one or more data bases at the same time. Index-sequential access method with fixed record length is used. Up to 10 indices are allowed in the same data base. While every record can be read, protection is made centrally. As all indices and keys are updated when writing, no sorting is needed. The data bases can be moved to different mass-storage devices without any limitations.

Networks Up to 32 systems can be used together as well as central peripherals.

PROGRAM DEVELOPMENT

Source code ASMZ source code is prepared with a powerful editor in ASCII-code.

Assembly The assembler works in two passes, it: 1. Reads the source code and sets up a reference table. 2. Generates the relocatable object code while updating the object library.

Listing A powerful program and error listing function speeds up program development.

Linking Object code modules written in ASSEMBLER or any compiling language can be linked together and organised with the help of a powerful linker — ESTAB.

PROMation The result can be placed in PROM.

Segmentation Program modules can be segmented.

Error handling As every type of error can be handled without stopping the program, very stable application programs can be written.

Debugging Debugging is made very easy by using TRACE and single step (CTRL-S) functions.

PROGRAMMING TOOLS

Opcodes ASMZ gives the choice of using either DataBoard or Zilog mnemonics.

Symbols Global and local symbols.

Expressions +, —, *, /, OR, AND, XOR and shift.

Constants Decimal, hexadecimal, octal and character.

Pseudo instructions Over 40 pseudo instructions to control the assembly process, define symbols and generate data.

Macro processing Provides the user with the possibility to define own opcodes or redefine existing ones. Parameters may change the macro in each call.

Conditional assembly Conditional assembly, with many options, of segments in programs.

Data structures Data structures are easily defined.

PLC 32 individual program location counters.

ADVANCED PROGRAMMING

ISAM With the aid of special instructions, advanced, multi-user data base systems can be built.

Supervisor Call Every system function can be reached with supervisorcalls. Parameters are transferred using standardised parameter-blocks.

External programs Programs written in arbitrarily chosen languages can be loaded, started and manipulated from an ASSEMBLER-program.

DATA INDUSTRIER AB

Box 2029, 18302 TÄBY. Tel 08-768 06 60. Telex 10978.

SYMBOL DEFINITION INSTRUCTIONS

EQU	Equates a symbol to the value of an expression.
LET	Equates a symbol to the value of an expression or redefines a symbol value.
XLET	Equates a symbol to the value of an expression and redefines the symbol as external.
ENTRY	Identifies symbols, defined in this program, that may be used in another program. This can also be accomplished by following the symbol with a "*" character when it is defined.
EXTERN	Identifies symbols that are defined in another program but referenced in this program.
CEXTR	Has the same function as EXTERN but do not include the symbol in the global symbol table.
SELECT	Defines symbols to be included in the global symbol table.

DATA DEFINITION INSTRUCTIONS

DA	Defines and generates a 16-bit constant.
DB	Defines and generates an 8-bit constant.
DS	Reservs a number of bytes.
DMA	Defines and generates a specified number of 16-bit constants.
DMB	Defines and generates a specified number of 8-bit constants.

STRUCTURE DEFINITION INSTRUCTIONS

STRUC	Defines a data structure.
ENDS	Defines the end of a data structure.

LIST CONTROL INSTRUCTIONS

PROG	Defines a program name in the object label and establishes the main heading for each page of the list.
ZPROG	The same function as PROG, but indicates that Zilog mnemonics instead of DataBoard mnemonics are used.
TITLE	Defines an assembly listing sub-title.
EJECT	Generates a form-feed on the listing device.
EJCTL	Conditionally generates a form feed on the listing device.
LIST ON/OFF	Controls the listing output from the assembler. Lines containing errors are however always listed.
LIST DATA/ NODATA	Controls if the data exceeding the space of one line will be listed or not.
LIST IF/NOIF	Controls if unassembled conditional statements are listed or not.
LIST EDIT/ NOEDIT	Enables or disables the list output editing function.
LIST MACRO/ *NOMACRO*	Enables or disables the listing of expanded macro code.
CROSS	Causes ASMZ to generate a cross-reference symbol table following the assembly listing.
NCROS	Causes ASMZ not to generate the symbol table.
LCNT	Specifies the number of printed source statement lines per page at the assembly listing.
RADIX	Sets the output RADIX to hexadecimal or octal.
ERROR	Generates an error message which is shown in error field of the listing and counted in the error count.

LOCATION COUNTER CONTROL INSTRUCTIONS

ORG	Causes the location counter to be set to a the value of an expression.
PLC	Controls which of the 32 program location counters to be used.
ALIGN	Sets the current PLC to the next available address modulo the value of an expression.
TP	The current PLC will be set to the top of the next memory page modulo 256.
TPL	Conditionally turns a memory page.

ASSEMBLER CONTROL INSTRUCTIONS

END	Terminates the assembly of the program and can define the place where control is transferred to, when the program is loaded.
COPY	Obtaines code from a source library and includes it in the program currently being assembled.
TARGT	Defines the target machine for the assembly to Z-80 or 8080.

CONDITIONAL ASSEMBLY INSTRUCTIONS

IF	Provides conditional assembly capability of a sequentional set of instructions using either arithmetical or logical statements in the test.
ELSE	Complements the currently active IF-block. It is used to construct IF — ELSE — ENDF blocks.
ENDF	Terminates the presently active conditional assembly.
DO	Causes a statement to be processed a specified number of times.
IFB, IFNB	Conditional macroparameter test.
IFDEF, IFUND	Tests if a user symbol is defined or not.

EXTENDED MACHINE INSTRUCTIONS

SVC	Executes a supervisor call.
SINT	Simulates an interrupt on a specified level.

MACRO INSTRUCTIONS

MACRO	Declares the start of a macro definition.
ENDM	Declares the end of a macro definition.
%	A symbol preceeded by a "%" character will be replaced with the decimal value of that symbol at the macro call.
	Concatinates two symbol names to one symbol name.

ESTAB FEATURES

Multi language	Modules written in any compiling language can be linked together to a relocatable or absolute task.
Segmentation	The tasks can be segmented giving possibilities of reentable code and switching segments as different routines are executed.
Task initialisation	The initial options, priorities, resources etc of the tasks can be specified.
Listing	Several listing options.
Output file	The user has full control of the organisation of the resulting task.

ESTAB PROGRAM COMMANDS

END	Ends command sequence and produces an output file and listing.
PAUSE	Puts ESTAB in pause state.
ABORT	Aborts program execution and deletes the output file.
REMOTE	Causes the linking to be aborted if an error is detected.
CHAIN	Re-assigns the command file.

ESTAB TASK COMMANDS

TASK	Specifies the output task file.
VERSION	Sets the version number in the task file.
ABSOLUTE	Generates an absolute task.
RELOCATABLE	Generates a relocatable task.
EXPAND	Adds additional memory to a task.
MAXLU	Sets the upper limit of the number of task files that the task can use.
MAXNODE	Sets the number of nodes generated to the task at load time.
DEFSIZE	Defines a default size for the task.

STACKLIMIT Sets the stack size at load time.
PRIORITY Sets the default priority to a task.
OPTION Sets task optins at load time.

ESTAB LIST COMMANDS

CHECK Lists references on the console.
LIST/NOLIST Enables/disables the listing of symbols of different types.
LINECOUNT Defines the number of lines to print on each page of the listing.
PRINT Specifies where the printing list should be written.
RADIX Selects the base when printing values to hex or octal.
PLCLIST Selects the PLCs to be printed in the linking list.
LOG Directs a log of the commands and error messages to a log file.

ESTAB SYMBOL COMMANDS

EQU Gives a number of symbols a value and includes them in the symbol table.
CEQ Same as EQU but does not change the value of a symbol that already has a value.
CEXTERNAL Gives a value to certain CEXTERN symbols.
UNSOLVED Gives a value to symbols that are referenced but not given any value during the linking.
REDEFINE Changes any already defined symbol to a value.
DATE Defines symbols which can be used to generate an ASCII date and time within a user program.

ESTAB MODULE COMMANDS

INCLUDE Includes a file or a module into the linking.
LIBRARY Includes a module if any ENTRY that corresponds to an undefined symbol exist.
OBJECT Renames a temporary file and saves it.

ESTAB OUTPUT FILE COMMANDS

AUTORST Used when generating segmented tasks.
SEGMENT Defines both the resolution of the memory mapping unit and the PLCs that will be included in the different segments of the task.
LOCAL Cause entries to be defined as local to the current segment.
PURENAME Defines a four character ASCII string which is used to name a pure task.
OPTION PURE Establishes the pure code segment of a task.
IMPURE Terminates the current pure-segment.
PURETOP Sets the lowest load address of an impure segment.
PLCORDER Specifies the order in which the PLC-segments are organised in the memory.
PLCBASE Specifies the first PLC.
PLCCOMMON Selects the PLC-numbers under which any common areas will be allocated.
PLCEND Generates symbols that will be given the first free location after each PLC. The symbols may be referenced in the user program.
PLCSTART Generates symbols which will be given the start value of each PLC.
PLCNR Selects a PLC for a ORG command.
ORG Orgins the PLC given in a previous ORG command.

TRACE FEATURES

Breakpoints Several breakpoint conditions like changes in memory locations, program flow and CPU-registers.
Modes TRACE can either work in command mode or trace mode.
Command mode Full control over the execution, memory and register contents, I/O ports. Parts of the programs can be run in real-time and any symbols can be specified in the form of relocation registers.
Trace mode Single step, interpretating and free run. Subroutines already debugged can be suspended from the debugging.

TRACE TASK COMMANDS

END Terminates the debugging.
LOAD Loads a task.
PAUSE Pauses the user program.
START Starts the debugging of a specified task.

TRACE GENERAL COMMANDS

CONVERT Converts a value to decimal, octal and hex values.
COPY ON/OFF Enables and disables copying of all commands and information displayed on the screen to a specified file.
RADIX Selects the default base for numeric values.
SET Sets a symbolic relocation register to a value.
SVC OFF/ON Disables and enables the interpretation of RST 7 as an SVC.

TRACE MEMORY COMMANDS

EXAMINE Displays a specified memory content in ASCII, dec, oct or hex notation.
FILL Fills a specified memory area with a value.
MODIFY Displays 16 bytes.

TRACE CPU COMMANDS

DISABLE/ ENABLE Disables and enables the interrupt logic of the CPU.
INPUT Displays the content of an I/O port.
NMI Enables TRACE to use NMIs as breakpoints.
OUT Sends data to an I/O port.
REGISTER Displays the content of the current or a specified CPU register.

TRACE BREAKPOINT COMMANDS

HALT Stops the program execution.
JTABLE Specifies a base address for 256 byte jump table.
LIST Displays the current trace conditions and their addresses.
REMOVE Removes all trace conditions or a specified breakpoint.
SUBROUTINE Defines a subroutine not to be traced.
TRACE, (switch) Puts a trace condition at a specified address.
TRACE Enters the single step mode.
TRACE, D (isable) Enters the free run mode until a new breakpoint is encountered.
TRACE, J (ump) Enters the interpretating mode. Only executed JMP, CALL and RET instructions are printed.
TRACE, M (emory) Enters the interpretating mode and prints the change in a memory cell specified by WATCH.
TRACE, S (can) Enters the free run mode and prints the change in a memory cell specified by WATCH.
TRACE, R (egister) Enters the free run mode and prints changes in CPU registers.
WATCH Defines an address in the memory to be watched for any changes.

TRACE DEBUG COMMANDS

DEBUG Starts debugging at a specified address.
PROCEED Continues the debugging.

TRACE MODE COMMANDS

SPACE Step to next instruction in single step mode.
D (isable) Enters free run mode.
J (ump) Sets a breakpoint at the target address location.
K (ill) Removes a breakpoint at the current address location.

L (oop)	Executes the current DJNZ until end.
R (egister)	Prints the CPU registers.
T (race)	Specifies a single step breakpoint at the current address location.
W (atch)	Specifies the current address argument to be watched for changes.
X (ecute)	Supresses the currently entered subroutine.
Any other key	Exits trace mode and enters command mode.