HP Series 64800
Cross Compilers
Pascal and C Languages

HEWLETT PACKARD

Technical Data January 1985

Table of Contents

Microprocessor Specific Cross Compiler Information

Introduction

Software development is a highly labor-intensive process. Development managers recognize that the productivity of software engineers can be significantly improved when software is designed, coded, debugged, and maintained in a high-level programming language.

Supported Microprocessors

The Hewlett-Packard 64000 Logic Development System offers both Pascal and C Cross Compilers for the following microprocessors:

- Motorola 68000, 68008, and 68010
- Motorola 6800 family
- Motorola 6809
- Intel 8086, 8088, 80186, 80188
- Intel 8080, 8085
- Zilog Z80
- Zilog Z8001, Z8002
- National Semiconductor NSC800
- Hitachi 6301

Operating Environment

These software development tools are available for the following host environments:

- HP 64100A and 64110A development stations
- HP 9000/series 500 HP-UX computer systems
- Digital Equipment Corporation VAX/VMS computer systems

The HP 64000 Pascal and C Cross Compilers are much more than language translators. Not only are the language implementations optimized for microprocessor-based design applications, but the compilers are integrated into the HP 64000 Logic Development System and generate the database needed to operate HP's powerful emulator-based analysis tools.

C Language Description

The HP 64000 C Cross Compilers provide a full implementation of the C programming language as defined in *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie (Prentice-Hall, 1978) and the supplement published November 15, 1978.

Hewlett-Packard has also extended the standard C language to improve its utility as a tool for microprocessor system programming and to make optimum use of the HP 64000 System software development tools. For applications requiring portability to other C Compilers, these extensions may be ignored.

The "standard I/O library" functions (printf, getchar, etc) are not included. Register variables are treated as an automatic storage class and are not specially optimized.

Additional language implementation details are in the REFERENCE INFORMATION section under "C Language Extensions and Restrictions".

Pascal Language Description

The HP 64000 Pascal Cross Compilers implement a dialect of the Pascal programming language which is specifically designed and optimized for microprocessor development applications. Although closely aligned with the Pascal language as defined in *Pascal User Manual and Report — Second Edition* by Kathleen Jensen and Niklaus Wirth (Springer-Verlag, 1976), the HP 64000 System Pascal language contains most features of and many extensions to the Jensen and Wirth "standard".

Specific language implementation details may be found in the REFERENCE INFORMATION section under "Pascal Language Extensions and Restrictions".

Selecting a Programming Language

There are three primary reasons for programming in a high-level language such as Pascal or C, rather than a low-level assembly language:

1. **Reduce Development Time** It is easier to write programs in a high-level language. Programmers can express alogrithms in a more natural notation than the underlying hardware allows. With high-level languages, software can be written and debugged in less time.
2. **Improve Program Readability** It is easier to read and understand a program written in a high-level language. Since professional programmers spend much of their time revising programs, written by themselves or others, this issue has a major economic impact in many development organizations.
3. **Improve Program Portability** High-level language source code is machine independent. A single program can be compiled to execute on several different processors.

Most users select a programming language based upon organizational standards, past experience of the design team, or application requirements. When the correct choice is not readily apparent, the following comparison of language benefits may help:

Pascal was developed for teaching students how to program. Many language features are included to help the programmer quickly identify and avoid common errors, reducing the time required to obtain functionally correct programs.

Pascal's strong type checking and highly structured syntax encourage source code that is easier to read and understand. For this reason, maintenance costs for software written in Pascal tends to be lower than equivalent software written in C.

C was developed for writing operating systems by highly experienced programmers at AT&T Bell Laboratories. C provides the programmer with a powerful language in which programs can be tersely worded. The C language contains very few features which protect the programmer from programming errors and oversights.

The standard C language provides the low level facilities needed to support microprocessor development applications. For this reason, the HP 64000 System C language is highly standard and software written in C can be moved to and from the HP 64000 System development environment at lower cost than equivalent software written in Pascal.

Host Computer Selection

Software development strategies are based on organizational structure and microprocessor applications. To best meet YOUR needs, the HP 64000 System Pascal and C Cross Compiler products operate on several host computers.

Each of the cross compiler products will execute on the HP 64100A or 64110A development stations. In this environment, compilation speeds are very fast, typically in excess of 600 lines per minute.

Each of the Pascal and C Cross Compilers are also available to execute on the HP-UX 9000/500 or VAX/VMS computer systems. In a host computer environment, the cross compiler can be accessed from a time-sharing terminal. This is a particularly effective environment for microprocessor applications involving large software development teams. When the time needed to complete an compile-link iteration is longer than convenient, the program can be executed as a background process, allowing the programmer to continue working on other tasks. Typical compiler speeds in various host environments are documented in technical data supplements, HP P/N 5953-9252 and HP P/N 5953-9254.

Pascal and C, native-mode compilers are available for the HP-UX and VAX/VMS computer systems. This allows testing Pascal or C source code using a host computer as an execution environment. The relative difficulty of implementing this software development strategy depends upon both the programming language selected and the microprocessor application.

HP 64000 System C is a highly standardized language. C programs can be compiled and tested in the host environment, and later moved into the HP 64000 System environment with relatively little change. Changes required include: addition of "C" and "target microprocessor" directives, deletion of standard I/O function calls, deletion of references to HP-UX or VMS execution environment resources, revisions to development environment-unique file naming conventions, and revisions to any inherently machine-dependent code. Moving software from the host to the development station or visa versa is very fast using the high-speed link software.

HP 64000 System Pascal is an extension to most native-mode Pascal compiler languages. Pascal programs that use only the common-denominator subset of both HP 64000 System and the native-mode Pascal can be compiled and tested in the host environment, and later moved into the HP 64000 system environment with relatively little change. Other Pascal programs may require such extensive changes as to make the implementation of this strategy impractical.

Strong Linkage between Language Systems and HP 64000 Test and Analysis Tools

Symbolic interface to measurement tools is important for programming productivity. All the HP 64000 language-linked analysis tools support symbolic interface. These tools support code execution and module test prior to the existence of target hardware, as well as real-time system analysis using in-circuit emulation for software/hardware integration. The HP 64000 emulator-based analyzers and other logic analysis tools can be used independently or interactively for different levels of testing and system integration. Transition from "functional testing" (without target hardware) to "system testing" (with target hardware) is straightforward since common tools and environments are used for both phases.

Emulator analysis in combination with the HP 64000 networking to host computer systems significantly reduces the equipment cost per engineer. Many users can access this test environment via remote control feature through the host computer. This emulation analysis arrangement provides many advantages over traditional "simulation" products:

HP's emulators can be operated out-of-circuit — Code can be fully written and functionally debugged prior to the availability of any target hardware.

Execution is real time — resulting in maximally short test and debug time. The host computer is not loaded down when testing instruction intensive code. The quick turnaround in the find-fix-verify cycle encourages good code documentation and insightful testing.

Execution is real, not modeled — Emulation does not require the inevitable compromises between accuracy and the speed of the simulator.

All of HP's powerful, emulator-based analysis tools are available to evaluate program execution — The programmer can locate faults quickly. Testing can be accelerated even further through the HP 64000 logic analyzers, powerful tools that show the bus activity while the designed system is running at operational speed.

Multiple emulators can be operated together — System test in a true multiprocessor execution environment can be very complex. However, the HP 64000 system is a "universal system"; processors from different families and different manufacturers can be emulated. The HP 64000 emulators simplify emulation in a multiprocessor design.

The transition from functional test to system test is 'graceful' — As target hardware is completed, it can be added to the software test bed. There is no dramatic shift from a "simulated" to "true" execution environment with the attendant subtle surprises.

Source Code Generation

Pascal and C source code can be entered in a variety of ways. On the HP 64100A/64110A development stations, a softkey-driven editor provides a user-friendly means of writing and editing Pascal, C or assembly language source code. In the HP-UX or VMS environment, the programmer can select any of several editors for generating source code.

Optimization of Object Code

Software development often involves trade-offs between the labor investment to develop a program, the execution efficiency of the program, and the subsequent maintainability of the code.

Software developed with the HP 64000 System Pascal and C Cross Compilers typically generates less efficient code — in terms of both memory utilization and execution speed — than the same program written in assembly language. The labor, however, required to write, debug, and subsequently maintain the program will be significantly lower; many users report 5 to 10 times labor savings due to the use of high-level programming languages such as Pascal and C.

The HP 64000 System Pascal and C Cross Compilers include optimization algorithms to generate efficient code. Often, however, the compiler designer had to choose between tight, high-speed code and code which will execute safely in all possible situations. In HP 64000 System, the trade-offs are always made to achieve safe, reliable code. Compared to a native-mode compiler, the optimization of cross compiler code is exacerbated by the extremely wide range of possible execution situations. Assumptions cannot be made about the target operating system, e.g., external variables may really be an I/O port, necessitating reloading for each reference, etc.

A programmer, writing software for a specific microprocessor application, frequently identifies more efficient generated code constructs by simply inspecting an assembly language listing. It is important to identify those few key program areas where optimizing code will have maximum impact on the overall software system performance.

The HP 64000 System supports a 4-step methodology and provides a set of tools to achieve high-leverage code optimization:

1. Write and functionally debug as much of the program as possible in C or Pascal.
2. Use the HP Model 64310A Software Performance Analyzer to locate and flag those areas of the code which primarily affect program performance.
3. Use the ASM_FILE compiler directive to generate assembly code for all modules flagged by the Software Performance Analyzer.
4. Edit or recode the assembly language files for maximum performance.

Linking C, Pascal, and Assembly Language Programs

Relocatable object code modules generated by the Pascal and C Cross Compilers and the Cross Assembler (for a given target microprocessor) are all compatible and processed by the same HP 64000 System Linker.

Each Pascal and C Cross Compiler has a $LIST_CODE$ directive which generates a symbolic assembly language listing of the compiled object code. By turning LIST_CODE on when an assembly language routine is called, the assembly language programmer obtains a "roadmap" for establishing the proper linkages.

Three product notes address the issues of linking code modules from cross compilers and cross assemblers: *Communicating Between 68000 Pascal Modules and 68000 C Language Modules* (HP P/N 5957- 7339); *Communication Between 68000 C Language Modules and 68000 Assembly Modules* (HP P/N 5957-7340); and *Communication Between 68000 Pascal Modules and 68000 Assembly Modules* (HP P/N 5957- 7341). Although the 68000 microprocessor is used as the example, these product notes provide information about the general subject of linkage between assembly language, C, and Pascal program modules.

Linking Relocatable Code and Compatibility with Other Execution Environments

The HP 64000 System Pascal and C Cross Compilers generate relocatable object code. The HP 64000 System Linker combines multiple relocatable modules into a single, absolute load file. This file can then be used by the HP 64000 system for execution in an emulation environment, on a target system, or programming PROMs (Programmable Read Only Memory).

Please refer to the HP 64000 System Cross Assembler/Linker Technical Data Sheet (HP P/N 5953-9251) for more information about the HP 64000 System Linker.

By reformatting, HP absolute object file can be loaded to other emulation or target systems. Complete information is contained in the *HP 64000 Logic Development System File Format Reference Manual* (HP P/N 64980-90933), *HP-UX File Format Manual* (HP P/N 64880-90903), and *VAX/VMS Hosted File Format Manual* (HP P/N 64882-90903).

HP 64000 Cross Compiler Operations (Pascal and C)

Operation

The compiler is invoked by using the "compile" softkey on a HP 64000 development station and by executing the "comp" command in HP-UX and in VMS. The first line of the C language source code must be "C" in quotes. The second line of the C language source code and first line of Pascal language source code must be the processor directive, e.g., "80186", "68010", "6809", "BZ80", etc. The compile syntax is in UNIX* format in the HP-UX environment and is in VMS* format in the VMS environment.

*UNIX is a trademark of AT&T Bell Laboratories.
VMS is a trademark of Digital Equipment Corporation.

Options

nolist	No listing except error messages. All $LIST$ directives in the source program are ignored.
list	Lists source program without expansions. All $NOLIST$ directives in the source program are ignored.
expand	Specifies a list of all source lines with an expansion of the assembly language. Also shows INCLUDE files and expanded MACROs if used. All $LIST_CODE$ and $FULL_LIST$ directives in the source program are ignored.
nocode	Suppresses the generation of object code. Only the source code will be listed in pass 2.
xref	Specifies a symbol cross-reference listing for the source file. All $XREF$ directives in the source file are ignored.

listfile On HP 64000 development stations, the assembly output listing is generated and stored in the specified destination. On HP-UX, the assembly output listing will go to "stdout" and is subject for redirection. On VAX/VMS, if the output file name is given, the listing is stored in the specified file; otherwise, the source file name will be used for the listing file with extension .LIS.

comp_sym Generates an entire compiler symbol table for use in creating the comp_db file by the linker.

Reference Information

Information common to all HP 64000 cross compilers is discussed in this section. Information about specific cross compilers is discussed in the section on that product.

Index to General Pascal and C Programming Information

Compiler Directives

Common to All HP 64000 System Pascal and C Cross Compilers

$AMNESIA$ causes the compiler to forget register contents after each use of the register. May be used to ensure that variables representing memory mapped I/O ports are reloaded each time they are needed.

ASM_FILE creates an assembly language source file intermixed with Pascal or C source lines displayed as assembler comments. This file will be accepted by the assembler for the selected microprocessor. Critical modules can be modified and reassembled for better performance.

$ASMB_SYM OFF$ causes the compiler to omit the generation of an assembler symbol file (used by the analysis tools during emulation); results in shorter compile time.

$DEBUG$ checks for run-time overflow in addition, subtraction, negation, multiplication and absolute value operations. Also checks for division by zero in division and modulus operations.

$EMIT_CODE OFF$ causes the compiler to omit the generation of the relocatable object code.

$FULL_LIST$ causes INCLUDE files to be listed and MACROs to be expanded in the compiler listing file. A complete listing for documentation and debugging is generated.

$LINE_NUMBERS OFF$ halts generation of symbols (for inclusion in the assembler symbol file) corresponding to the Pascal or C source code line numbers for use by the analysis tools during emulation and results in shorter compile time.

$LIST OFF$ halts generation of the compiler listing file. This allows users to selectively list segments of a program.

$LIST_CODE$ generates a listing file containing an expanded assembly language listing intermixed with the Pascal or C source code. This allows users to selectively list segments of a program with expanded assembly instructions.

$LIST_OBJ$ specifies that the listing file contain the relocatable object code (hexadecimal representation) when the $LIST_CODE$ directive or expand option is used.

ORG and END_ORG allow variables to be allocated at sequential absolute memory addresses beginning at a specified location.

$PAGE$ generates a form feed output to the compiler listing file.

$RECURSIVE$ causes local data and temporaries to be allocated on the stack. Memory requirements for the generated code may be greater and execution speed may be slower than with $RECURSIVE OFF$.

$SEPARATE$ causes code related to program and constants to be separated from data. The specific implementation will vary depending on the target microprocessor.

$TITLE$ prints a specified header line at the top of each subsequent page of the compiler listing file.

$WARN OFF$ prints error messages, but inhibits warning messages in the compiler listing file.

$WIDTH$ specifies the number of significant characters in the source line to be compiled; additional characters are ignored.

C Compiler Directives

The following compiler directives are available for C only.

$ENTRY OFF$ disables the automatic use of the 'main' function as the program entry point and allows the user to establish an application-unique program initialization and entry procedure.

$EXTENSIONS$ allows the use of HP 64000 System extensions to the C language. Specifically permits use of HP 64000 format for defining binary, octal, decimal, and hexadecimal constants.

$FIXED_PARAMETERS$ specifies that subsequent functions will always pass the same types and number of parameters. May result in more efficient code and ensures compatibility with the parameter passing conventions of HP 64000 System Pascal procedures and functions.

$INIT_ZEROES OFF$ inhibits initialization to zero of all static and external variables not explicitly initialized.

$LONG_NAMES OFF$ causes identifiers to be truncated at eight characters instead of fifteen.

$SHORT_ARITH$ causes arithmetic with 'short' and 'char' variables to occur without conversion to type 'int', and arithmetic with 'float' to occur without conversion to type 'double'.

$STANDARD$ causes warnings to be printed in the compiler listing file whenever any "non-standard C" feature (as defined by Kernighan and Ritchie) is used.

$UPPER_KEYS$ instructs the compiler to recognize upper-case keywords instead of lower case.

$USER_DEFINED$ permits redefinition of certain operators in the language and results in calls to user provided run-time routines. Allowed operators: +, −, *, /, %, ==, !=, ‹=, ›=, ›, and ‹.

Pascal Compiler Directives

The following compiler directives are available for Pascal only.

$ANSI$ causes a warning message to be issued when any HP 64000 System Pascal feature is used which is not part of 'standard' Pascal. This directive is useful for identifying program areas which are likely to create transportability problems.

$EXTENSIONS$ allows programmers to use the microprocessor oriented language extensions. These include: functional type changing, the address function, the BYTE data type, built-in functions, SHIFT and ROTATE, and nondecimal constant representations.

$EXTVAR$ causes subsequently defined variables to be declared EXTERNAL, allowing data communication among modules. A variable declared EXTERNAL is imported from another program.

$GLOBPROC$ causes subsequent main block procedures to be declared GLOBAL and exported for use in other programs.

$GLOBVAR$ causes subsequent main block variables to be declared GLOBAL and exported for use in other programs.

$IOCHECK OFF$ causes input/output procedures and functions to return with an error code in the event of an I/O error. With $IOCHECK ON$, an I/O error causes all input/output activity and program execution to terminate.

$RANGE ON$ causes the compiler to generate code to check array indices, value parameters, variable set elements, and subrange assignments for legal values.

$USER_DEFINED$ permits redefinition of certain operators in the language and therefore results in calls to user provided run-time routines. Allowed operators: +, -, *, /, DIV, MOD, =, ◊, ‹=, ›=, ‹, and ›.

C Cross Compiler Information

Compiler Operation

A four-pass compilation process translates C source programs into relocatable code for the target microprocessor. In the first pass, the compiler reads the C source code, checks for errors, and creates an intermediate language file. The next pass inserts parameters related to the newly created file. The third pass is microprocessor dependent, and the intermediate language file is translated into a tokenized assembler file. In the final pass, the compiler generates a relocatable object file. A listing file, an assembler source file, an assembler symbol file, and a compiler symbol file may also be generated during the final pass. An additional preprocessor pass may be executed to process macros, include files, and conditional compilation.

Language Extensions

Program code and constants may be compiled to a relocatable area separate from data and variables, allowing the design of ROM and RAM memory systems.

Variables can be assigned to absolute memory locations permitting easy access to memory I/O addresses and other fixed memory location entities.

The first fifteen characters of an identifier are significant, permitting more meaningful names to be used. Truncation to eight characters can also be forced.

Arithmetic may be done with 'short' variables without conversion to type 'int', or with 'float' without conversion to type 'double'.

The keywords may be defined to be upper case instead of the standard lower case at the user's preference.

Binary, decimal, hexadecimal, and octal constants in HP 64000 format (B, D, H, and O or Q suffixes) may be used in addition to the standard C forms.

Structures may be assigned, compared for equality and inequality, passed as parameters, or returned from functions. This simplifies operations on variables of structure types.

The enumeration type is supported, allowing the use of symbolic identifiers instead of integers for classifying data.

A shift by a negative value is allowed (equivalent to a shift in the opposite direction by a positive value, i.e., a»b and a«-b are equivalent if b is a signed type).

Large constants are optimized and those which match in full or in part may be in the same memory location.

The user may selectively redefine the following operators: +, −, *, /, %, ==, !=, ‹, ›, ‹=, and ›=. This allows users to extend the meaning of an operator to the operands of a user defined type.

Dynamic Memory Allocation Extensions

Because library routines are common between the Pascal and C Cross Compilers, the dynamic memory allocation routines required for Pascal programming may also be used in C programs as external functions.

INITHEAP to initialize a block of memory as a memory pool or 'heap'.

NEW and DISPOSE to allocate and deallocate blocks of memory from the 'heap'.

MARK and RELEASE to simplify 'heap' management for short term memory usage. The state of the heap is contained in a pointer variable.

Language Restrictions

The "standard I/O library" functions are not included.

Register variables are treated as an automatic storage class and are not specially optimized.

The HP 64000 System C Cross Compilers do not presume any specific host operating system environment; therefore the C preprocessor #include ‹filename› is not available. (However, the #include "filename" refers to any user-defined file, and is available.)

A string constant may not be extended across multiple source code lines, and its maximum length is therefore limited to 238 characters.

The C preprocessor '#line' instruction is not implemented.

Intrinsic Data Types

short	An 8-bit signed integer
unsigned	An 8-bit unsigned integer
int	A 16-bit signed integer
unsigned	A 16-bit unsigned integer
long	A 32-bit signed integer (implemented as 16-bit on 6800 and 6809)
unsigned long	A 32-bit unsigned integer (implemented as 16-bit on 6800 and 6809)
char	An 8-bit value defined by the ASCII character set. Equivalent to an unsigned short.
float	A 32-bit binary floating point number in the IEEE single-precision format.
double	A 64-bit binary floating point number in the IEEE double-precision format.

Derived Data Types

The following derived data types have an implementation representation that is target microprocessor dependent. They may contain holes (unused bytes) due to memory alignment requirements. For statically allocated variables, these holes are filled with zeroes.

struct	structures
union	unions
typedef	type definitions
enum	enumeration type

Pascal Cross Compiler Information

Compiler Operation

A three-pass compilation process translates Pascal source programs into relocatable code for the target microprocessor. In the first pass, the compiler reads the Pascal source code, checks for errors, and creates an intermediate language file. The second pass is microprocessor-dependent, and the intermediate language file is translated into a tokenized assembler file. In the final pass, the compiler generates a relocatable object file. A listing file, an assembler source file, an assembler symbol file and a compiler symbol file may also be generated during the final pass. An additional special preprocessor pass may be executed to process MACROs, INCLUDE files, and conditional compilation.

Language Extensions

Full facilities for separate compilation are provided. Procedures and variables may be declared EXTERNAL (imported from another program) and GLOBAL (exported to other programs) for use by separately compiled or assembled program modules. The HP 64000 System Linker satisfies these external references between separate modules, or with relocatable library files.

Program code and constants may be compiled to a relocatable area separate from data and variables, allowing the design of ROM and RAM memory systems.

Variables may be assigned to absolute memory locations, permitting easy access to memory mapped I/O locations and other fixed memory location entities. These absolute variable assignments can be made in one program module and externally linked in all other modules.

The direct access file I/O procedures READDIR and WRITEDIR, and the string I/O procedures STRREAD and STRWRITE are implemented in addition to the standard READ, READLN, WRITE, and WRITELN. The auxiliary functions and procedures, LINEPOS, POSITION, MAXPOS, REWRITE, APPEND, RESET, PUT, GET, CLOSE, OPEN, SEEK, PROMPT, PAGE, OVERPRINT, EOF and EOLN are also implemented.

Since the I/O hardware on the target system is undefined, the libraries supplied with the Pascal Cross Compilers will cause I/O to be implemented via simulated I/O using the HP 64000 System printer and disc files, HP 64100 development station keyboard, display, and RS-232C serial interface port. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

The CASE statement may contain an OTHERWISE clause to specify action when the CASE expression fails to match any of the listed values.

The CONST, TYPE, and VAR declarations may be made in any order, and more than one declaration is allowed. This extension improves program readability.

Constant-valued expressions are allowed wherever constants are allowed in "standard" Pascal.

The first 15 characters of an identifier and letter case are significant ("standard" Pascal uses only 8), permitting more meaningful names to be used.

MARK and RELEASE procedures enhance the "standard" Pascal facilities for dynamic memory management by simplifying the 'heap' management for short term memory usage.

The built-in function ADDR returns the address of any variable, providing users with greater power in manipulating data structures. The result is a pointer that is compatible with any pointer type. "Standard" Pascal only allows pointers to dynamically allocated memory space.

Predefined integer types of SIGNED_8, SIGNED_16, SIGNED_32, UNSIGNED_8, UNSIGNED_16, and UNSIGNED_32 permit selection of a variable's memory implementation independent of the target microprocessor.

The built-in functions SHIFT and ROTATE, permit bit-level data manipulation via logical and circular shifting of data.

Expressions and variables are permitted to have their implicit type changed by being included as the parameters of a "function" call of any named TYPE. "Standard" Pascal allows this by using variant type records.

With the $USER_DEFINED$ option, the meaning of the arithmetic operators (+,−,*,/,DIV,=,‹,›,‹=, and ›=) may be selectively redefined. This allows users to extend the meaning of an operator to the operands of a user defined type.

A string data type is defined with a maximum length of 255 characters. The length of a string variable is dynamic and can be changed during program execution.

An additional special preprocessor pass provides the ability to have INCLUDE files, conditional compilation, and MACRO expansion.

Language Restrictions

HP 64000 System Pascal is a subset of the Jensen and Wirth "standard" Pascal with the following constraints:

Procedures and functions are not allowed as parameters.

Set subranges are not implemented. Also, sets are limited to 256 elements or less of any ordinal type.

The standard function SQR is not implemented.

Packing of data is not carried below the byte level. The PACKED key word is ignored by the compiler, except when defining a string, and the standard procedures PACK and UNPACK are not implemented. Bit packing in 8-bit, 16-bit, or 32-bit data item can be performed using the SET construct "*"(AND), "+"(OR), and "−"(set difference) operators, and the predefined SHIFT and ROTATE functions in conjunction with the functional type change operations.

Program parameters are not implemented since the compiler does not presume the existence of a host operating system. The predefined variables, INPUT and OUTPUT, may be used without mentioning them in a program parameter list.

Intrinsic Data Types

BYTE An 8-bit signed integer.

CHAR An 8-bit value defined by the ASCII character set. Equivalent to the UNSIGNED_8.

INTEGER A 32-bit signed integer. (except, implemented as a 16-bit signed integer on 6800 and 6809)

REAL A 32-bit binary floating point number in the IEEE single-precision format.

LONGREAL A 64-bit floating point number in the IEEE double-precision format.

STRING A 256-byte array equivalent to PACKED ARRAY [0..255] of CHAR. Byte 0 contains the run-time string length.

TEXT Type FILE variables of type TEXT are called 'textfiles'. The components of a textfile, of type CHAR, are further structured into 'lines' separated by 'line markers'. Line markers are written to a textfile using WRITELN. When reading a textfile, line markers are detected by using EOLN.

SIGNED_8 An 8-bit integer.

UNSIGNED_8 An 8-bit unsigned integer

SIGNED_16 A 16-bit integer.

UNSIGNED_16 A 16-bit unsigned integer

SIGNED_32 A 32-bit integer. (Not implemented on 6800 and 6809.)

UNSIGNED_32 A 32-bit unsigned integer. (Not implemented on 6800 and 6809.)

User Defined Types

Enumerated Type Identifiers are assigned ordinal values based upon their enumerated order. Each enumerated type variable occupies one byte of memory.

Subrange Type Compiler generates run-time range checking to verify that values of the variable are restricted to a sequential subset range of the ordinal base type.

Structured Types

The structured types ARRAY, RECORD, SET and FILE are characterized by component type and structuring method. A structured type definition may contain the term PACKED as an indication to the compiler that memory is to be economized at the expense of execution speed.

ARRAY An ARRAY is made up of a fixed number of components, each of which can be accessed via the index. The index specifies the array size and must be an ordinal type. The components can be any type, but all components must be of the same type. Both single- and multi-dimensional arrays can be used.

SET A SET is the power set (set of all subsets) of any ordinal base type. The base type is limited to 256 or fewer elements. Subsets are allowed, but the value of the maximum element of the SET may not exceed 255.

RECORD A RECORD, like an ARRAY, is a structured variable with several components. The components of a RECORD may have different types; however, they are accessed by name, not by index value.

FILE A FILE consists of a sequence of components which are all of the same type. Only one component in a FILE is accessible at a time. FILEs are usually associated with peripheral storage devices and their length is not specified in the program.

Pointer Type

In 'standard' Pascal, dynamic variables can be generated by using the standard procedure NEW(p). New memory space is allocated for the new dynamic variable, and the pointer variable '^' holds the address of the new dynamic variable. With the ADDR(p) function, HP 64000 System Pascal permits a pointer variable to hold the address of any variable. The memory representation for a pointer type is microprocessor dependent.

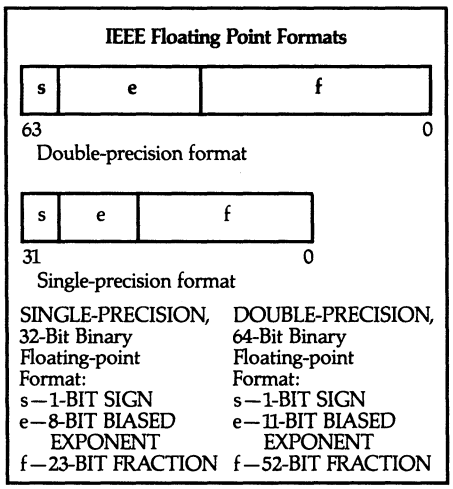
Run-Time Libraries

Run-time libraries are supplied with each compiler as relocatable code modules that are linked with the relocatable program modules. These libraries perform commonly encountered tasks, such as the real number operations, moving and comparing structured data, and error handling. Both the Pascal and C Cross Compilers use common microprocessor-specific run-time library routines, which may be shared in programs composed of modules from both languages.

If the libraries as supplied do not fully conform to the needs of the program being developed, they may be replaced with user supplied routines by simply specifying the new modules at link time.

Real Number Operations

An integral part of the compiler and library routines are real number operations. HP 64000 System Pascal and C both use the IEEE floating point formats with the Pascal type 'real' and C type 'float' implemented as an IEEE single-precision (32-bit) format and the Pascal type 'longreal' and C type 'double' implemented as an IEEE double-precision (64-bit) format.



The Pascal and C Cross Compilers handle operations on single and double precision real numbers by generating calls to value-returning functions from the run-time libraries. Calls are generated automatically whenever floating point operations, as listed, are encountered in the source code.

Both single and double precision versions of the following routines are included in the run-time libraries.

General Operators:
ADD (+) SUBTRACT (−) MULTIPLY (*) DIVIDE (/)
NEGATE (−) [C only]

Value Comparisons:
EQUAL (=) NOT EQUAL (◊ or !=) LESS THAN (‹)
GREATER THAN (›) LESS THAN OR EQUAL (‹=) GREATER THAN OR EQUAL (›=)

Predefined Functions:
ABS (absolute value) ARCTAN (arctangent) COS (cosine)
SIN (sine) EXP (exponent) LN (natural logarithm)
SQRT (square root)

Data Conversions:
Single- and double-precision real numbers may be converted to and from 8-, 16- and 32-bit integers. Language access differs between Pascal and C. Implementation specifics vary with the target microprocessor.

Number Ranges

8-bit	signed	−128 to 127
	unsigned	0 to 255
16-bit	signed	−32,767 to 32,768
	unsigned	0 to 65,535
32-bit	signed	−2,147,483,648 to 2,147,483,647
	unsigned	0 to 4,294,967,295
32-bit	real or float	−10 E 38 to 10 E 38
64-bit	longreal or double	−10 E 308 to 10 E 308

Character Set

Alphabetic Characters — All upper and lower case characters ('A' through 'Z' and 'a' through 'z').

Numeric Characters — Digits '0' through '9' for decimal numbers and including 'A' through 'F' (and also 'a' through 'f' for C language) for hexadecimal numbers.

Special Characters — blank, dollar sign, apostrophe, left and right parentheses, comma, plus, minus, equals, less than, greater than, decimal point, slash, colon, semicolon, left and right brackets, left and right braces, caret (^), asterisk (*), and underscore (_). The C language also includes: ampersand (&), exclamation point, quotation mark, pound sign(#), percent sign (%), tilde (~), backslash, and question mark.

6800 Family and 6301

The HP 64811 Pascal Cross Compiler or HP 64821 C Cross Compiler provides high-level language software development support for the 6800 microprocessor. The generated 6800 code will also execute on the 6801, 6802, 6803, 6808, and the Hitachi 6301 microprocessors.

Models 64811AF and 64821AF compilers are hosted on the HP 64100A or 64110A development stations.

Models 64811S Pascal Language System or Model 64821S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the HP 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute code for a given source program.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities such as source lines display and module and global variable tracing in terms of the language being used to develop the software.

Pascal Cross Compiler

Model 64811 Pascal cross compiler provides a complete implementation of the HP 64000 System Pascal Language (with 16-bit integer). Code for procedures and functions can be generated for recursive execution. Pascal routines can be written to handle interrupts from external signals or software programs. An extensive Pascal I/O facility simplifies software debug. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64821 C Cross Compiler provides a complete implementation of the HP 64000 System C Language (with 16-bit integer). Code for functions can be generated for recursive execution. C functions can be written to handle interrupts from external signals or software programs.

Operation

The compiler directive "6800" appearing in the first line invokes the 6800 Pascal cross compiler. If the directive is preceded by line "C", the 6800 C cross compiler will be invoked.

The 6800 compilers generate code for the 6800 microprocessor. The code will also execute on 6801, 6802, 6803, 6808, and 6301.

Reference Information

Pointers are represented as 16-bit values. There are five run-time libraries supplied with the 6800 compilers for real number operations, error handling, and Pascal I/O functions.

Compiler Directives

$OPTIMIZE$ causes the compiler to generate short jump instructions for all the forward branches, providing a savings of three bytes for each forward branch operation. An error message will be displayed to inform the user to turn 'optimize' off at the given location where a forward branch is out of range.

$DEBUG$ causes arithmetic error checking such as overflow, underflow, or divide by zero error conditions to be performed using subroutine calls to the routines in the debug library.

$RANGE$ used to check array index expressions, value parameters, and variable assignments for correct subrange values before performing the operation. Range-checking code will also be produced if the variable is a scalar data type or is a subrange data type. (Pascal Only)

$PASCALvarPARAMS$ used with $FIXED_PARAMETERS$ to modify the standard parameter passing technique in C to support $USER_DEFINED$ routines written in Pascal. (C only)

Ordering Information

Model	Description
64811AF	Pascal Cross Compiler for 6800 Family
64811AR	License to use a second copy of 64811AF
64811AX	One-Time Update to current revision for 64811AF
64811S	Pascal Language System for 6800 Family
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64811SR	License to use a second copy of 64811S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64811SX	One-Time Update to current revision for 64811S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64821AF	C Cross Compiler for 6800 Family
64821AR	License to use a second copy of 64821AF
64821AX	One-Time Update to current revision for 64821AF
64821S	C Language System for 6800 Family
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64821SR	License to use a second copy of 64821S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64821SX	One-Time Update to current revision for 64821S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

HP 64813 Pascal Cross Compiler or HP 64822 C Cross Compiler provides high-level language software development support for the 6809 microprocessor.

Models 64813AF and 64822AF compilers are hosted on the HP 64100A or 64110A development stations.

Models 64813S Pascal Language System or Model 64822S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the HP 64100/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute codes from a given source program.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities such as source lines display and module and variable tracing in terms of the language being used to develop the software.

Pascal Cross Compiler

Model 64813 Pascal cross compiler provides a complete implementation of the HP 64000 System Pascal Language. Code for procedures and functions can be generated for recursive execution. Pascal routines can be written to handle interrupts from external signals or software programs. An extensive Pascal I/O facility simplifies software debug. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64822 C compiler provides a complete implementation of the HP 64000 System C Language. Code for functions can be generated for recursive execution. C functions can be written to handle interrupts from external signals or software programs.

Operation

The compiler directive "6809" appearing in the first line of the program invokes the 6809 Pascal cross compiler. If the directive is preceded by line "C", the 6809 C cross compiler is invoked.

Reference Information

Other Implementation Specifics

Pointers are represented as 16-bit values. The default size of the run-time stack is 512 bytes. If the debug library is linked, the stack is assigned in the program area of the linked modules. If the LIB_6809 library is linked, the stack will be assigned in the data area of the linked modules. The stack size can easily be changed to other sizes.

All procedures and function parameters are passed on the stack. The addresses of reference parameters in Pascal and structure parameters and parameters with '&' operator in C, are pushed onto the stack. For value parameters in Pascal and C, the values of the parameters are pushed onto the stack.

There are four run-time libraries supplied with the 6809 compiler for real number operations, error handling, and Pascal I/O functions.

Compiler Directives

$OPTIMIZE$ causes the compiler to generate short jump instructions for all forward branches to save memory space. The compiler also heuristically selects one of the three stack offset sizes (5 bits, 8 bits, and 16 bits) for parameter addressing in a recursive routine instead of the default offset size (16 bits).

$DEBUG$ causes arithmetic error checking, such as overflow, underflow, or divide by zero, to be performed using subroutine calls to the routines in the debug library.

$INTERRUPT$ insures that the subsequent Pascal main-level procedure or C function will be suitable for use as an interrupt routine. Nothing special is done upon entry. On exit, a return from interrupt (RTI) is generated to return control instead of the normal RTS instruction.

Ordering Information

Model	Description
64813AF	Pascal Cross Compiler for 6809
64813AR	License to use a second copy of 64813AF
64813AX	One-Time Update to current revision for 64813AF
64813S	Pascal Language System for 6809
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64813SR	License to use a second copy of 64813S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64813SX	One-Time Update to current revision for 64813S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64822AF	C Cross Compiler for 6809
64822AR	License to use a second copy of 64822AF
64822AX	One-Time Update to current revision for 64822AF
64822S	C Language System for 6809
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64822SR	License to use a second copy of 64822S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64822SX	One-Time Update to current revision for 64822S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

The HP 64814 Pascal Cross Compiler or HP 64818 C Cross Compiler provides high-level language software development support for four microprocessors: 8086, 8088, 80186, and 80188. Both compilers will optionally generate library calls to utilize an 8087 numeric coprocessor for floating point operations. Both 8086 and 80186 code will execute on an 80286 operating in 'compatibility mode'.

Models 64814AF and 64818AF compilers are hosted on the HP 64100A or 64110A development stations.

Model 64814S Pascal Language System or Model 64818S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the Model 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute codes from a given source program.

All code generated for procedures and functions is suitable for recursive or reentrant execution.

Executable code and constants are compiled to a separate relocatable area, apart from data and variable areas.

The 8086 address space is divided into 64k-byte segments. More efficient code is generated if only a single segment is used. Different compiler directives are provided for optimal management of multiple code segment routine calls and multiple data segments variable accesses.

Compiler directives are available to include constants in PROG segments, implement multiple DATA segments, implement multiple PROG segments, place the stack in a separate DATA segment, and implement one or more dynamic memory (heap) DATA segments for large applications.

The compiler can also generate position-independent code, permitting decisions regarding program location in memory to be deferred until run time.

Both C and Pascal cross compilers support interrupts from external control signals or from software programs. Trap-handling routines may be written to implement system calls.

Both compilers generate necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high-level debugging facilities such as source lines display and module and global variable tracing, in terms of the language being used.

Pascal Cross Compiler

Model 64814 Pascal compiler provides a complete implementation of the HP 64000 System Pascal Language. An extensive Pascal I/O facility simplifies software debug. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64818 C compiler provides a complete implementation of the HP 64000 System C Language.

Operation

Four compiler selection directives are recognized: "8086", "8088", "80186" and "80188". Although the 8086/8088 and 80186/80188 have identical instruction sets, the "8088" and "80188" directives are required to inform the linker that the 8088 and 80188 use 8- instead of 16-bit memory and to correctly select the 8088/80188 library routines.

The 80186/80188 compilers generate identical code to that produced by the 8086/8088 compilers, except that tests are made to generate more optimized code for the 80186/80188 in six functional areas: The 80186/80188 instructions 'ENTER' and 'LEAVE' permit subroutine calls and exits to be made with less code and faster execution speed. The 80186/80188 instructions 'PUSH Immediate', 'ROTATE Immediate', 'SHIFT Immediate', and 'Integer MULtiply Immediate' are also used. The 80186/80188 compilers do not use the 'PUSH All', 'POP All' instructions, the string manipulation instructions 'INS' or 'OUTS', and the array checking 'BOUND' instruction. The run-time libraries for the 80186/80188 are the same as for the 8086/8088.

Reference Information

Other Implementation Specifics

The default of the run-time stack is 512 bytes. The stack will be allocated in the data area of the linked modules. The stack can easily be modified to other sizes.

Eight separate versions of the run-time libraries are supplied. One set of four supports the 8086/80186; the other set supports the 8088/80188. Selection of the appropriate library is determined by whether the PROG area is limited to one segment (NEAR) or occupies multiple segments (FAR), and whether the DATA area is limited to one segment (SHORT) or occupies multiple segments (LONG). The NEAR/FAR and SHORT/LONG selections are independent; thus, versions of the libraries are provided to support all four combinations.

Compiler Directives

$ALIGN$ causes data and temporaries to be aligned on word boundaries; NEW and DISPOSE will always be called with an even number of bytes; and constants are sized to occupy an even number of bytes. This improves memory access efficiency on the 8086 and 80186.

$DEBUG$ causes code to be generated to check for run-time arithmetic overflow, division by zero, and Pascal case errors.

$SEPARATE$ has no effect since PROG and DATA are always separate.

$INTERRUPT$ causes subsequent C functions and Pascal main-level procedures to be generated suitable for use as an interrupt routine. On entry, all registers will be pushed onto the stack. On exit, the registers will be restored and 'IRET' will be used to return instead of the normal 'RET'.

$INT n$ causes the subsequent C function or Pascal main-level procedure to be used as the interrupt service routine for interrupt number 'n', where 'n' is in the range of 0 to 255. A call to the function or procedure name, results in the generation of software interrupt number 'n'.

$RANGE$ generates code to check that values are within expected range and array indices are within bounds. Constants are tested at compile time and variables are tested at run time.

$FAR_EXTVARS$, $DS_EXTVARS$, $CS_EXTVARS$, $ES_EXTVARS$, and $SS_EXTVARS$ are mutually exclusive options for selecting the addressing mode used in the generated code to access external variables. Applies to external variables which are subsequently declared.

$FAR_EXTVARS$ causes the ES (extra segment) register to be loaded prior to accessing the variable. Thus, the variable may be located in any segment.

$DS_EXTVARS$ is the default selection and will cause external variables to be accessed via the current contents of the DS (data segment) register. If variables can be stored in multiple memory segments, users must be cognizant of the run-time segmentation environment.

$CS_EXTVARS$ causes external variables to be accessed via the current contents of the CS (code segment) register.

$SS_EXTVARS$ causes external variables to be accessed via the current contents of the SS (stack segment) register.

$ES_EXTVARS$ causes external variables to be accessed via the current contents of the ES register. Compiler assumes proper initialization of the contents of the ES (extra segment) register.

$SEPARATE_CONST OFF$ causes constants to be placed in the PROG relocatable area and accessed via the CS register. Otherwise, constants will be in DATA and accessed via the DS register. When constants are in PROG, 32-bit pointers should be used (except when PROG and DATA are located in the same 64k memory segment).

FAR_PROC causes subsequent procedures and functions to be declared "FAR" so that they can be called by programs located in other memory segments. If pointers are set to be 32-bit values, the contents of DS register will be saved prior to the call and restored after the return.

$POINTER_SIZE 32$ causes pointers to be 32-bit values. Otherwise, the pointer size is 16-bits, and the user is responsible for determining that memory addressing outside the current segment can be performed correctly.

$SHORT_LIBRARIES OFF$ causes libraries to be called which operate with 32-bit pointers. Otherwise, libraries are limited to 16-bit pointer operation.

$FAR_LIBRARIES$ causes libraries to be called "FAR" and be consistent with 32-bit pointers.

Ordering Information

Model	Description
64814AF	Pascal Cross Compiler for 8086/186/88/188
64814AR	License to use a second copy of 64814AF
64814AX	One-Time Update to current revision for 64814AF
64814S	Pascal Language System for 8086/186/88/188
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64814SR	License to use a second copy of 64814S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64814SX	One-Time Update to current revision for 64814S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64818AF	C Cross Compiler for 8086/186/88/188
64818AR	License to use a second copy of 64818AF
64818AX	One-Time Update to current revision for 64818AF
64818S	C Language System for 8086/186/88/188
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64818SR	License to use a second copy of 64818S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64818SX	One-Time Update to current revision for 64818S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

The HP 64815 Pascal Cross Compiler or HP 64819 C Cross Compiler provides high-level language software development support for three microprocessors: 68000, 68008 and 68010.

Models 64815AF and 64819AF compilers are hosted on the HP 64100A or 64110A development stations.

Models 64815S Pascal Language System or 64819S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on an HP 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a DEC VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute codes from a given source program.

All code generated for procedures and functions is suitable for recursive or reentrant execution. All executable code and constants are compiled to a separate relocatable area, apart from data and variable areas. Three addressing modes: absolute short, absolute long, and address register indirect with displacement are provided for optimal static variable access management. Variables may be accessed through different address modes in different modules. Provisions are made for the compiler to generate position-independent code, permitting decisions regarding program location in memory to be deferred until run time. Both C and Pascal cross compilers support interrupts from external control signals or from software programs. Trap-handling routines may be written to implement system calls.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities, such as source line display and module and variable tracing, in terms of the language being used to develop the software.

Pascal Cross Compiler

Model 64815 Pascal compiler provides a complete implementation of the HP 64000 System Pascal Language. An extensive Pascal I/O facility simplifies software debug. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64819 C compiler provides a complete implementation of the HP 64000 System C Language.

Operation

Two compiler selection directives are recognized: "68000" and "68008". The "68000" directive is used for both the 68000 and 68010 microprocessors. Although the 68008 and 68000 instruction sets are identical, the "68008" directive is required to inform the linker that the 68008 uses 8-bit instead of 16-bit memory and to correctly select the 68008 library routines.

Reference Information

Other Implementation Specifics

Pointers are allocated 4 bytes of memory.

The default size of INTEGER is 32 bits. Variables that required only 16 bits can be declared as type SIGNED_16. Significantly more efficient code is generated for multiplications and divisions.

The size of the run-time stack provided by the standard libraries is 256 words. It can easily be modified for other sizes.

Registers A5, A6, A7, and D7 have special uses. A7 is the stack pointer; it always points to the last item on the stack. A6 is the local frame pointer; it always points to the highest address of the data area for the currently executing procedure or function. A5 is the static data pointer; whenever $COMMON$ is ON in some part of a program, it is used to access external and other statically allocated variables. D7 is the function value return register.

All procedure and function parameters are passed on the stack. When a parameter is passed by reference, its address is pushed onto the stack. Only parameters of 4 bytes or less are passed by pushing their value onto the stack. Nested Pascal procedures access outer-level variables via a 'static link' parameter which is pushed onto the stack.

There are seven libraries provided for the 68000/68010 and seven libraries for the 68008 processor for real number operations, error handling, and Pascal I/O functions in various addressing modes.

Compiler Directives

$BASE_PAGE$ causes external variables to be accessed by the absolute short-addressing mode. This can be used to force absolute short-addressing mode for all program level variables.

$CALL_PC_SHORT OFF$ turns off the default program counter plus dispacement-addressing mode for calling Pascal procedures and functions and C functions.

$CALL_PC_LONG$ causes Pascal procedures and functions and C functions to be called using the program counter plus displacement plus index-addressing mode.

$CALL_ABS_LONG$ causes Pascal procedures and functions and C functions to be called using the absolute long-addressing mode.

$CALL_ABS_SHORT$ causes Pascal procedures and functions and C functions to be called using the absolute short-addressing mode.

$COMMON OFF$ turns off the default variable addressing mode of A5 register indirect plus displacement.

FAR causes external variables to be accessed by the absolute long-addressing mode. This can be used to force absolute long-addressing mode for all program level variables.

$INTERRUPT$ insures that the subsequent Pascal main-level procedure or C function will be suitable for use as an interrupt routine. On entry, the contents of all registers will be saved on the stack. On exit, the registers will be restored and RTE is used to return control instead of the normal RTS instruction.

LIB_PC_LONG causes predefined procedures and functions to be called using the program counter plus displacement plus index-addressing mode.

LIB_ABS_LONG causes predefined procedures and functions to be called using the absolute long-addressing mode.

LIB_ABS_SHORT causes predefined procedures and functions to be called using the absolute short-addressing mode.

$OPTIMIZE$ improves the efficiency of generated code by not reloading register contents following a store indirect or a store to an external variable. Forward jumps are assumed to be within a 128 byte range, providing a savings of two bytes of memory for each forward jump.

$SEPARATE$ has no effect since PROG and DATA are always separate.

$TRAP=n$ causes the subsequent Pascal main-level procedure or C function to be the handling routine for TRAP number 'n', when 'n' is in the range 0 to 15.

Ordering Information

Model	Description
64815AF	Pascal Cross Compiler for 68000/68008/68010
64815AR	License to use a second copy of 64815AF
64815AX	One-Time Update to current revision for 64815AF
64815S	Pascal Language System for 68000/68008/68010
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64815SR	License to use a second copy of 64815S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64815SX	One-Time Update to current revision for 64815S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64819AF	C Cross Compiler for 68000/68008/68010
64819AR	License to use a second copy of 64819AF
64819AX	One-Time Update to current revision for 64819AF
64819S	C Language System for 68000/68008/68010
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64819SR	License to use a second copy of 64819S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64819SX	One-Time Update to current revision for 64819S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

The HP 64816 Pascal Cross Compiler or HP 64820 C Cross Compiler provides high-level language software development support for the Z8001 and Z8002 microprocessors.

Model 64816AF and 64820AF compilers are hosted on the HP 64100A or 64110A development stations.

Model 64816S Pascal Language System or Model 64820S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the HP 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute code for a given source program.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities, such as source lines display and module and variable tracing, in terms of the language being used.

Pascal Cross Compiler

Model 64816 Pascal cross compiler provides a complete implementation of the HP 64000 System Pascal Language. All code generated for functions and procedures is suitable for recursive or reentrant execution. Pascal routines can be written to handle interrupts from external signals or software programs. An extensive Pascal I/O facility simplifies software debug. By rewriting the ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64820 C compiler provides a complete implementation of the HP 64000 System C Language. All code generated for functions and procedures is suitable for recursive or reentrant execution. C functions can be written to handle interrupts from external signals or software programs.

Operation

Two compiler selection directives are recognized: "Z8001" and "Z8002". "Z8002" directive is suggested when nonsegment mode is used.

Reference Information

Other Implementation Specifics

Pointers are represented as 32-bit values in Z8001 and are represented as 16-bit values in Z8002. The default size of the run time stack is 512 bytes. If the PLIB is linked, the stack will be assigned in the program area of the linked modules. For linking of other library files, the stack will be assigned in the data area of the linked modules.

There are three real number libraries and four unique run-time libraries supplied for use with Z8001 compilation and two unique run-time libraries supplied for use with Z8002 compilation for real number operations, error handling, and Pascal I/O functions.

Compiler Directives

$OPTIMIZE$ causes the compiler to remember the contents of registers and use the values when the corresponding variables are referenced. The compiler also generates short jump instructions for all forward branches to save memory space. An error message will be displayed to inform the user to turn 'optimize' off at the given location where a forward branch is out of range.

SC allows declaration of routines as System Call (SC) routines and results in the generation of SC instructions in place of call instruction.

$SEPARATE_CONST$ when separate addressing spaces are used for program and data, $SEPARATE_CONST ON$ causes all constants and case tables to be placed in the DATA section.

$SAME_SEGMENT$ (Z8001 only) indicates that the data is to be placed in the same segment as the program and can be accessed using PC relative addressing.

$SHORT$ (Z8001 only) all external variables defined while SHORT is on are accessed using short offset addressing.

Options that apply uniquely to the Z8001 are ignored during Z8002 compilations.

Ordering Information

Model	Description
64816AF	Pascal Cross Compiler for Z8001/Z8002
64816AR	License to use a second copy of 64816AF
64816AX	One-Time Update to current revision for 64816AF
64816S	Pascal Language System for Z8001/Z8002
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64816SR	License to use a second copy of 64816S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64816SX	One-Time Update to current revision for 64816S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64820AF	C Cross Compiler for Z8001/Z8002
64820AR	License to use a second copy of 64820AF
64820AX	One-Time Update to current revision for 64820AF
64820S	C Language System for Z8001/Z8002
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64820SR	License to use a second copy of 64820S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64820SX	One-Time Update to current revision for 64820S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

The HP 64823 Pascal Cross Compiler or HP 64824 C Cross Compiler provides high-level language software development support for the Z80 and NSC800 microprocessors.

Models 64823AF and 64824AF compilers are hosted on the HP 64100A/64110A development stations.

Models 64823S Pascal Language System or Model 64824S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the HP 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute code for a given source program.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities, such as source lines display and module and variable tracing, in terms of the language being used to develop the software.

Pascal Cross Compiler

Model 64823 Pascal compiler provides a complete implementation of the HP 64000 System Pascal Language. Code for functions and procedures can be generated for recursive or reentrant execution. Program code and constants may be separated from data in ROM-based applications. Pascal routines can be written to handle interrupts from external signals or software programs. An extensive Pascal I/O library simplifies software debug. By rewriting ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64824 C compiler provides a complete implementation of the HP 64000 System C Language. Code for functions and procedures can be generated for recursive or reentrant execution. Program code and constants may be separated from data in ROM-based applications. C functions can be written to handle interrupts from external signals or software programs.

Operation

The compiler selection directive is "BZ80" for Pascal and "Z80" for C. The "B" nomenclature in the Pascal directive is required to avoid conflict with an earlier Model 64812AF Pascal compiler. The NSC800 and Z80 have identical instruction set architectures and a unique NSC800 directive is not needed.

Reference Information

Pointers are represented as 16-bit values.

The default size of the run-time stack is 128 bytes in the data area of the linked modules. The stack size can easily be changed to other sizes.

The Z80/NSC800 compilers allow undisturbed access to the processor's alternate registers (A', B', C', D', E', F', H', and L'). The compiler generated code and run-time libraries use only the main registers (A, B, C, D, E, F, H, and L) and the index registers IX and IY. The main and alternate register swap instructions (EXX, EX, AF, and AF') are reserved for the user to use for interrupt processing or assembly language routines where rapid context switching is required.

Procedure and function parameters are passed by being pushed on the stack, and then accessed relative to IX or SP. If a function has a one-byte result, it will be returned in the A register; two-byte values are returned in the DE register; and the address of values larger than two bytes is returned on the stack.

Compatibility with Earlier Pascal Compilers

The HP 64823 Pascal Compiler use different code generation conventions than the earlier Model 64812AF Pascal Compiler, thus relocatable object code from the two compilers are not compatible. Pascal source code must be recompiled before it can be linked with other Model 64823 generated code. A user transporting Pascal source files from the 64812AF to 64823 compilers needs to be aware of six syntactical changes:

1. The compiler selection directive is "BZ80" instead of "Z80".
2. The default size of an INTEGER is 32 bits instead of 16 bits.
3. The built-in function ADDR now returns a type that is compatible with any pointer type, rather than being compatible with type INTEGER.
4. The default width of the source text is 240 characters instead of 120 characters.
5. The built-in function ROTATE replaces the earlier SHIFTC function, although SHIFTC will also remain available.
6. The default setting is $RECURSIVE OFF$ instead of $RECURSIVE ON$.

In addition to the syntactical changes, there may be semantic differences which will cause a program to function differently. Type changes, and expressions or assignments that mix operands of different sizes are potential sources for such differences.

Compiler Directives

$DEBUG$ causes code to be generated to test for run-time errors in arithmetic operations and size conversions.

$INTERRUPT$ causes subsequent C functions and Pascal main-level procedures to be generated suitable for use as an interrupt handling routine.

$OPTIMIZE$ causes the generated code to be more aggressively optimized. In particular, relative jumps to forward references will be used where the destination address may exceed the range of the relative offset. If such an error does occur, it will be flagged, and can be corrected by setting $OPTIMIZE OFF$ for the line causing the error.

$RECURSIVE$ causes local data and temporaries to be allocated on the stack. Memory requirements for the generated code will be greater and execution speed will be slower than with $RECURSIVE OFF$.

Ordering Information

Model	Description
64823AF	Pascal Cross Compiler for Z80/NSC800
64823AR	License to use a second copy of 64823AF
64823AX	One-Time Update to current revision for 64823AF
64823S	Pascal Language System for Z80/NSC800
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64823SR	License to use a second copy of 64823S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64823SX	One-Time Update to current revision for 64823S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64824AF	C Cross Compiler for Z80/NSC800
64824AR	License to use a second copy of 64824AF
64824AX	One-Time Update to current revision for 64824AF
64824S	C Language System for Z80/NSC800
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64824SR	License to use a second copy of 64824S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64824SX	One-Time Update to current revision for 64824S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

HP 64825 Pascal Cross Compiler or HP 64826 C Cross Compiler provides high-level language software development support for the 8085 and 8080 microprocessors.

Models 64825AF and 64826AF compilers are hosted on the HP 64100A or 64110A development stations.

Models 64825S Pascal Language System or 64826S C Language System each provides two cross compilers and two cross assembler/linkers. One of the cross compilers and one of the assembler/linkers executes on the HP 64100A/64110A development station. The other cross compiler and assembler/linker are hosted on an HP 9000/series 500 HP-UX or a VAX/VMS computer system, depending on the option selected.

Regardless of the host computer execution environment, the cross compilers and assembler/linker generate identical relocatable and absolute code for a given source program.

Both compilers generate the necessary information for symbolic debug in emulation. Programmers can troubleshoot the code using source program line numbers and global symbols, eliminating the task of looking up addresses. The compilers also generate files for use with other analyzers which provide high level debugging facilities, such as source lines display and module and variable tracing, in terms of the language being used to develop the software.

Pascal Cross Compiler

Model 64825 Pascal compiler provides a complete implementation of the HP 64000 System Pascal Language. Code for procedures and functions can be generated for recursive or reentrant execution. Program code and constants may be separated from data in ROM- based applications. Pascal routines can be written to handle interrupts from external signals or software programs. An extensive Pascal I/O library simplifies software debug. By rewriting ten primitive routines in the simulated I/O library, the user can redirect the HP 64000 System Pascal I/O facility to work with an application-specific target I/O system.

C Cross Compiler

Model 64826 C compiler provides a complete implementation of the HP 64000 System C Language. Code for functions and procedures can be generated for recursive or reentrant execution. Program code and constants may be separated from data in ROM-based applications. C functions can be written to handle interrupts from external signals or software programs.

Operation

The compiler selection directive is "B8085" for Pascal and "8085" for C. The "B" nomenclature in the Pascal directive is required to avoid conflict with an earlier Model 64810AF Pascal compiler. Generated code is compatible with the 8080 microprocessor.

Reference Information

Other Implementation Specifics

Pointers are represented as 16-bit values.

The default size of the run-time stack is 128 bytes in the data area of the linked modules. The stack size can easily be changed to other sizes.

The compiler generated code and run-time libraries make use of the registers A, B, C, D, E, F, H, and L .

Procedure and function parameters are passed by being pushed on the stack, and then accessed relative to SP. If a function has a one-byte result, it will be returned in the A register; two-byte values are returned in the DE register; and the address of values larger than two bytes is returned on the stack.

Compatibility with Earlier Pascal Compilers

HP 64825 Pascal Compiler use different code generation conventions than the earlier Model 64810AF Pascal Compiler, thus relocatable object codes from the two compilers are not compatible. Pascal source code must be recompiled before it can be linked with other Model 64825 generated code. A user transporting Pascal source files from HP 64810AF to HP 64825 compilers needs to be aware of six syntactical changes:

1. The compiler selection directive is "B8085" instead of "8080" or "8085".
2. The default size of an INTEGER is 32 bits instead of 16 bits.
3. The built-in function ADDR now returns a type that is compatible with any pointer type, rather than being compatible with type INTEGER.
4. The default width of the source text is 240 characters instead of 120 characters.
5. The built-in function ROTATE replaces the earlier SHIFTC function, although SHIFTC will also remain available.
6. The default setting is $RECURSIVE OFF$ instead of $RECURSIVE ON$.

In addition to the syntactical changes, there may be semantic differences that will cause a program to function differently. Type changes and expressions or assignments that mix operands of different sizes are potential sources for such differences.

Compiler Directives

$DEBUG$ causes code to be generated to test for run-time errors in arithmetic operations and size conversions.

$INTERRUPT$ causes subsequent C functions and Pascal main-level procedures to be generated suitable for use as an interrupt handling routine.

$OPTIMIZE$ causes the generated code to be more aggressively optimized. In particular, relative jumps to forward references will be used where the destination address may exceed the range of the relative offset. If such an error does occur, it will be flagged, and can be corrected by setting $OPTIMIZE OFF$ for the line causing the error.

$RECURSIVE$ causes local data and temporaries to be allocated on the stack. Memory requirements for the generated code will be greater and execution speed will be slower than with $RECURSIVE OFF$.

Ordering Information

Model	Description
64825AF	Pascal Cross Compiler for 8085
64825AF	License to use a second copy of 64825AF
64825AX	One-Time Update to current revision for 64825AF
64825S	Pascal Language System for 8085
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64825SF	License to use a second copy of 64825S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64825SX	One-Time Update to current revision for 64825S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64826AF	C Cross Compiler for 8085
64826AF	License to use a second copy of 64826AF
64826AX	One-Time Update to current revision for 64826AF
64826S	C Language System for 8085
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64826SF	License to use a second copy of 64826S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System
64826SX	One-Time Update to current revision for 64826S
Opt 001	Hosted on HP-UX 9000/500 Computer System
Opt 003	Hosted on VAX/VMS Computer System

Printed in U.S.A. 1/85
5953-9250

For more information, call your local HP Sales Office or nearest Regional Office: Eastern (301) 258-2000; Midwestern (312) 255-9800; Southern (404) 955-1500; Western (213) 877-1282; Canadian (416) 678-9430. Ask the operator for Instrument Sales. Or, write: Hewlett-Packard, 1501 Page Mill Road, Palo Alto, CA 94304. In Europe: Hewlett-Packard S.A., 7, rue du Bois-du-Lan, P.O. Box CH-1217 Meyrin 2, Geneva, Switzerland. In Japan: Yokogawa-Hewlett-Packard Ltd., 29-21, Takaido-Higashi 3-chome, Suginami-ku, Tokyo, 168.