1.00.00 INTRODUCTION

PL/9 has become a very popular language among programmers involved with the day to day writing of control programs for systems or products incorporating the Motorola MC6809.

PL/9 has been designed specifically for the Motorola MC6809 and as a result it takes full advantage of the architecture of this powerful processor. As no tradeoffs have been made to make the 'core' of PL/9 compatible with other processors there is no intention of porting PL/9 to other processors at this time although we are keeping a close eye on the MC68008 (the 8-bit version of the MC68000).

PL/9 is currently only being offered under the FLEX disk operating system as we feel that this single user operating system offers the most solid base for developing control oriented programs. FLEX imposes absolutely no restrictions on what you may or may not do with the MC6809 instruction set. The simplicity of a non-interrupt driven operating system is also highly desirable lest you find yourself spending more time accommodating the requirements of your operating system than you will writing your program. If your program will fit into 48K of memory the FLEX working environment will be hard to beat.

The popularity of PL/9 is, in our opinion, due to six factors:

1. The interactive working environment offered by the co-resident editor/compiler/tracer greatly assists the programmer when debugging, fine tuning, or adding 'bells and whistles' to programs.

   The traditionally lengthy edit-compile-debug, edit-compile- -debug, edit-compile-debug cycles ... with time consuming file loading and saving through the disk operating system intervening between each step of the process is now a thing of the past!
   past!

2. The syntax of PL/9 is close enough to Pascal and 'C ' that if you are are familiar with either of these languages you will not have any problems getting used to the syntax of PL/9.

3. The extremely fast compile time. A source file that produces 8K of binary takes <u>less than 60 seconds</u> to compile! This INCLUDES the load time of the compiler, the source, and the saving of the binary object file!

4. The overall performance of the product incode efficiency and execution times is VERY impressive; the SIEVE of Eratosthenes benchmark runs in 9.4 seconds, with a total program size of 192 bytes (678 bytes if all I/O routines are included).

5. NO RUN-TIME OVERHEADS; be they in the form of a license for an interpreter or a license for a mathematics package. The cost of PL/9 includes an un-limited and non-exclusive license to the PL/9 REAL math package.

6. The ease with which programs that start from system power-up can be developed. PL/9 has the built-in capability to intercept ALL of the MC6809 interrupts, including RESET. Gone are the days of generating system startup and interrupt handling procedures in assembly language. Now the ENTIRE program from system startup to system power down can handled by one language ... PL/9.

1.00.01 ACKNOWLEDGEMENTS

We sincerely thank Ron Anderson, author of the 'FLEX USERS NOTES' column in '68 Micro Journal for his authoritative constructive criticism of the earlier versions of this product. If Ron had not taken the time, trouble, and effort to enter into lengthy correspondence with us and offer many suggestions for improvement this product and its documentation would not be what they are today.

We would also like to acknowledge Ron Anderson as the author of algorithms used in the SCIPACK library and for his valuable assistance in improving the internal arithmetic package.

We would also like to acknowledge Matt Scudiere as the author of the coefficients used in the SINE and ARCTAN routines in the SCIPACK library.

Our thanks also to Neil Jarman for his efforts in improving the file handling capabilities and editor features in our assembler product MACE. These features have subsequently been incorporated into PL/9 by the author.

The information given to Windrush by the author was edited by Bill Dickinson to form this manual but he refuses to take the blame for it.

We also thank Motorola for developing THE most powerful 8-bit processor available, the MC6809, and for having the foresight to produce the MC68008. Keep 'em coming you guys!

Our thanks also to Ric Hammond of Smoke Signal Broadcasting and Richard Don of Gimix for having the foresight in seeing the potential of the SS-50C bus and the Motorola MC6809 and producing hardware that was reliable enough to withstand the demands of professional users. Without the reliability offered by the hardware designs and manufacturing standards of these two companies we would have never considered that the SS-50C bus and FLEX was worth supporting.

And last, but by no means least, we would like to thank John Alford of Alford and Associates for developing what we consider to be the most powerful word processing package available; SCREDITOR III. Without this beautiful piece of software this manual would NEVER have been completed.

Any references to SSB mean Smoke Signal Broadcasting Incorporated any references to GIMIX mean Gimix Incorporated and any references to SWTP mean South West Technical Products Incorporated.

1.00.02 HOW TO USE THIS MANUAL

This manual has been written as a tutorial on PL/9, aimed at the complete newcomer to structured programming languages in general, and PL/9 in particular. All of the important features of the language are described, with examples given where appropriate.

As the manual is organized as a tutorial the sequence of presentation had to be arranged to ensure that the topics contained in each section only made reference to preceding sections wherever possible. This has necessitated various trade-offs in the order of presentation.

To help compensate for the tutorial organization of the various sections in the manual the index to the keyword definitions is arranged in alphabetical order. This has been devised to assist the user when he needs to address a specific topic.

The information in the Language Reference Manual outlining the keywords is not meant to provide masses of detail; it is designed for quick reference. If the information in the Language Reference section does not supply the detail you require consult the equivalent section in the Users Guide. The Users Guide, which is bound in a separate manual, is arranged in roughly the same order and has, in most cases, identical section titles to assist in locating information.

The body of the reference manual can be considered to be in two main sections:

(1) The CONFIGURATION of PL/9 to match your system hardware environment and to suit your own programming requirements.

(2) The TECHNICAL REFERENCE section which covers the following topics:
- a. The PL/9 editor.
- b. The PL/9 compiler.
- c. The PL/9 tracer.
- d. The PL/9 language.
- e. The PL/9 libraries.

The body of the users guide can be considered to be in two main sections:

(1) The USERS GUIDE which covers six main areas in considerable detail. This section places heavy emphasis on working programs as examples, which, for the most part, are fully explained on a line by line basis:
- a. A brief description of data types and sizes.
- b. Program structures.
- c. A detailed description of PL/9 arithmetic, and data types.
- d. Advanced program structures.
- e. Bit operations and bit oriented program structures.
- f. Hardware oriented program structures.

(2) A selection of WORKING PL/9 programs.

We make no apologies for our frivolity from time to time in this manual. Technical documentation tends to be BORING. We feel that there is already an ample supply of boring manuals and books around. We do not intend to add this one to the collection.

S T O P - L O O K - L I S T E N

You are encouraged to read this manual from cover to cover before you sit down to write any serious programs. If you can't wait to get started refer to the section on configuring PL/9 to your system hardware. Further instructions for those of you without any patience will be found there.

If you encounter ANY difficulties read this entire manual starting from here!

1.00.03 HISTORY OF PL/9

PL/9 was written to meet a need that has been long felt without any real solution having previously been available. It is aimed primarily at the world of control systems, where programmers traditionally have a choice between BASIC and assembly language.

The former is simple to learn but results in slow and usually unreadable programs. Most BASICs also impose run-time interpreter license considerations that will, quite often, price a product out of the market.

Assembly language allows the programmer to get the utmost in performance out of his computer. Programming at this level has the disadvantage of taking a long time to learn, the programs are difficult to write, and can, at times, be almost impossible to debug!

It is against this background that languages such as PL/M (Programming Language for Microprocessors), originally written for the Intel 8080, but subsequently ported to the 8086 (amongst others), appeared a few years ago.

PL/M strikes a happy balance between code efficiency and ease of programming, while providing a degree of control over the hardware of the system that formal teaching languages such as Pascal, and mainframe languages such as 'C' cannot match.

The Motorola 6800 microprocessor was for many years poorly served for software of this type, except on expensive development systems. Then Tom Crosley wrote SPL/M, a subset of PL/M. This gave 6800 programmers the ability to write control programs, without spending more time trying to get around the limitations of the language than actually writing the program. The language was well-received by the 6800 fraternity and is still in widespread use. When the 6809 appeared, however, no comparable language was initially available (although SPL/M has subsequently been ported to the 6809).

1.00.04 EXISTING TOOLS

Pascal and 'C' are now available for use in microprocessors but due to their heritage are often cumbersome in handling the I/O, et. al., in microprocessor applications.

In addition most BASICs and Pascals require run-time interpreters; some of the 'native code' variety even charge you a license fee to use their math package. Most 'C' language compilers tend to require a very large overhead for the library modules which tend to be difficult to trim down. When these languages are used to generate code for small dedicated control systems it is like using a sledge hammer to crack a nut ... they'll do the job but they can also have undesirable side effects!

Although languages such as SPL/M, Pascal and C are a vast improvement on BASIC or assembly language, they can still be inconvenient to use.

The program development cycle consists of first using a text editor to create the source file, which is then written back to the disk at the end of the edit session. Next the compiler is called, giving the name of the source file to be compiled and any options required, such as whether an object file is wanted. Lastly, the object file is loaded and tested.

Few languages give the user much help in debugging his programs; it is usually a case of putting diagnostic print instructions into the code at crucial points. If an error is found, the whole edit-compile-load cycle must be repeated; this will take several minutes for all but the smallest of programs.

What is needed is some way of speeding up this process; BASIC for all of its deficiencies at least has the advantage that you can run the program without having to go through all that time consuming disc loading and saving, and it is this factor alone that keeps many programmers faithful to BASIC.

1.00.05 A 'BETTER MOUSETRAP'

PL/9 provides a solution to the problem of the edit-compile-load cycle by incorporating an editor into the compiler itself, making it unnecessary to exit the editor and load a separate compiler just to see if there are any syntax errors. This interactive approach is not new; one of the first assemblers for the 6800 was Motorola's CO-RES, which also included an editor and was very popular among small computer users. The inclusion of the editor does require a different approach to designing the compiler; IT IS NOW ESSENTIAL TO CONSERVE MEMORY since the source file is taking up a considerable amount of space. PL/9 is very frugal in its use of memory; each of its 18 symbol tables takes up only as much space as it actually needs (compare this with systems that use hash coding) and the source file is stored in a compacted form that allows the free use of spaces to improve readability.

PL/9 runs to just under 16K bytes leaving you 32K for your source and binary files. In addition a separate trace-debug facility, which loads into the FLEX transient command area, is provided to simplify the program testing cycle. At a command, PL/9 compiles the program into memory but puts extra information into the object code that allows it to retain control over the program while it is being tested. This means that the program can be stopped at any point or run one instruction at a time. Variables can be examined while the program is stopped or running, allowing the programmer the convenience of BASIC. When the program is working satisfactorily, it can be compiled to object code for use in ROM or RAM.

No understanding of 6809 assembly language programming is needed, although such understanding will help when interfacing to the system hardware.

PL/9 does not require a run-time package for its programs to work, and there are no license fees to pay for the use of compiled programs. All programs compiled by PL/9 can be placed in ROM and will run at any address in memory WITHOUT recompiling them. No memory addresses are reserved by PL/9 for special purposes, so you have complete flexibility.

Library routines are available to perform functions not provided directly by the language. Due to the architecture of PL/9 the users library functions actually appear to be an integral part of the language.

1.00.06 COMPATIBILITY WITH OTHER TEXT EDITORS

Even though PL/9 has its own built in editor there is nothing in its architecture that prevents you from using any editor that produces TSC EDITOR compatible files. Obviously this includes the TSC TEXT EDITOR, but also includes many cursor oriented screen editors such as Alford and Associates SCREDITOR III.

W A R N I N G

IF YOU USE AN EDITOR OTHER THAN THE EDITOR WITHIN PL/9 TO GENERATE YOUR PROGRAMS ENSURE THAT THE MAXIMUM LINE LENGTH DOES NOT EXCEED 127 CHARACTERS. EXCEEDING THIS LINE LENGTH WILL CAUSE THE LINE TO BE TRUNCATED AS IT IS LOADED.

If this truncation takes place the line will have four percent symbols (%%%%) appended to the end of the line as an aid in locating the line. In addition an error message 'LINE TOO LONG' will be posted as the file is loaded.

Many programmers prefer to use the editor they are most familiar with to do the main program entry. Once the vast majority of the program is entered and you enter the 'debugging' or 'fine tuning' stages of your program development work you will find the co-resident editor to be WORTH ITS WEIGHT IN GOLD if you value your time OR wish to keep your blood pressure down!

1.00.07 VARIATIONS BETWEEN VERSIONS

PL/9 VERSION 2.XX	January 1982. Initial Release
PL/9 VERSION 3.XX	June 1983. Tracer separated from compiler, REALS added to language pointer structures enhanced, upper/lower case supported, 'XREG' pseudo register added, SWI, SWI2, SWI3 and RESET procedures added, improved handling of unsigned bytes and integers, ability to call compiler from FLEX command line added. Improved REAL and INTEGER arithmetic. Manual improved.
PL/9 VERSION 4.XX	June 1984. Error messages separated from compiler, STACK manipulation enhanced, subscripted vector code generation improved, compile time options enhanced. More libraries and sample programs provided. Manual expanded.