A/BASIC COMPILER

REFERENCE MANUAL

MICROWARE SYSTEMS CORPORATION
P.O. BOX 954 • DES MOINES, IOWA 50304 • (515) 279-9856

A/BASIC COMPILER REFERENCE MANUAL

To our customers: Microware Systems Corp. is committed to providing high-performance, low cost software to you, now and in the future. This is not possible if unauthorized distribution and sale occurs. We ask your cooperation in controlling the distribution of this software to non-purchasers and notifying us of cases of unauthorized reproduction of this material so appropriate action may be taken.
Thank you.

REGISTRATION

#_1842_

Microware will provide patches or updates to A/BASIC during the warranty period only if the software is registered. Communication regarding A/BASIC must quote the registration number. To register your copy of A/BASIC sign, complete, and return this form.

Please print or type - Detach Here

- -

Company / Customer Name: ______________________________

Street Address: ______________________________

City / State / Zip: ______________________________

Phone (optional): ______________________________

Where Purchased: ______________________________

____

Signature: ______________________________

Mail To: Microware Systems Corp., P. O. Box 954, Des Moines IA 50304

Please do not mark below this line

Registration #_1842_

OVERVIEW OF A/BASIC

A/BASIC is a two-pass compiler for the M6800 family of microprocessors which converts programs written in a modified, extended BASIC language to pure M6800 machine-language programs.

A/BASIC is oriented towards applications currently programmed in assembly language, and is designed to generate optimized, compact and high-speed machine language. The output of the compiler can be run as stand-alone RAM, ROM or PROM programs, or multiprogrammed using RT/68 operating systems's multitasking capabilities.

The compiler does not require use of any "run-time package" as it contains an internal subroutine library "BASLIB" which contains copies of all required run time-subroutines. Integral to A/BASIC is a subroutine linkage editor which places one and only one copy of any required subroutine in the machine language program.

The compiler requires a minimum of 8K bytes of memory which is sufficient to typically compile 200 to 600 line programs. If additional memory is available, it may be used to expand tables so programs of virtually any size may be compiled.

A/BASIC AND its host operating system, RT68, may be used to effectively create fast, efficient programs for applications in areas such as:

* REAL-TIME CONTROL
* TEXT AND WORD PROCESSING
* DATA AQUISITION AND ANALYSIS
* SYSTEMS PROGRAMMING
* DATA COMMUNICATIONS
* SIMULATIONS AND GAMES
* REAL-TIME GRAPHICS
* BUSINESS PROGRAMS

The A/BASIC compiler and the RT/68 operating system can be customized for OEM applications that require special extensions or different operating environments.

INTRODUCTION AND DEFINITIONS

Because A/BASIC is both unique and quite powerful when used to full capability, it is highly reccommended that this manual be studied carefully before compilation is attempted.

Programmers with primary experience using interpreters will find several major differences between compilers such as A/BASIC and BASIC interpreters, mainly:

* There is no interactive operation during program debugging and testing. Often several compilations are required to produce the final version.

* While a machine language program is in execution, "run-time" errors are not detected unless checks were "written in" to the program, i.e., illegal input, array overflow, etc.

* Compilers, particularly A/BASIC, require the programmer to "lay out" memory assignments and utilization, and generally require a more sophisticated knowledge of actual machine characteristics and operation.

* Compiled machine langage programs run much faster and typically consume less memory than interpreted BASIC programs.

This manual is organized in such a way that it should be usable either for instuctional or reference purposes. In several areas references are made to other sections where a subject is more directly applicable.

Syntactic descriptions shown generally follow the form where a part of the syntax commonly used is enclosed such as <this>.

Commonly used definitions are:

<address> - a memory address represented as a decimal or hexa-decimal number. "Hex" numbers are preceded by a dollar sign such as $BF00.
<var> - A variable name. Usually the description that follows will qualify it as to what type it may be (string and/or numeric, etc.)
<expr> - An expression that results in a NUMERIC result, though it may include functions that have string-type arguments. An expres-sion may be complex and contain many terms or it may be a single constant number or a single variable.
<line #> - A legal line number that occurs somewhere in the BASIC program.
<str expr> - An expression that evaluates to a STRING result even though it may contain functions which have numeric expressions for argumants.
<str var> - A legal string variable name, or an indexed (subscripted) string variable.

Others such as <option>, <IO spec>, <array spec>, etc. are defined where they occur.

A/BASIC GLOSSARY

The terms defined below may be unfamiliar to some programmers or used in a special context in this manual.

ALLOCATION - The process of assignment of a specific memory address to variables or machine instructions.
CODE GENERATION - The process of creating machine language instructions.
COMPILE-TIME - Used to describe the time during which the BASIC program is being processed by the compiler.
LIBRARY - A collection of subroutines within the compiler (BASLIB) which are used to generate subroutines within the machine language program. It includes "images" of subroutines for mathematical functions, string processing, input/output, array operations, etc.
LINE REFERENCE - A reference from a statement to a line number which is that of ANOTHER line.
LINE REFERENCE TABLE - A table which is kept by the compiler which is used to store the correspondences between referenced line and the memory address assigned to the line.
LINKER/EDITOR - A portion of the A/BASIC compiler program which places subroutines which are required in the object program. The images are obtained from the library and the linker/editor inserts absolute addresses of the subroutines in the program. Only one copy of a subroutine will ever be generated in a program, and only those that are required by that specific program.
OBJECT CODE - the machine-language instructions generated by the compiler.
OBJECT FILE - The tape or other media file that the compiler has written the machine language program on.
OBJECT PROGRAM - The machine language program produced by the compiler from the BASIC source program.
PASS - A complete scan of the source program. A/BASIC is a two-pass compiler so it must completely read the source program twice.
REAL-TIME EXECUTIVE - The portion of the RT/68 Operating System that schedules and controls task execution.
RUN-TIME - Used to describe events or the time during which the machine language program is being executed.
SOURCE FILE _ The tape or other media containing the A/BASIC progam which is to be compiled.
SOURCE PROGRAM - The BASIC program text to be compiled.
SYMBOL TABLE - A table kept by the compiler during compilation that contains information about the correspondence between variable names and assigned memory addresses, and type of variable.
SYNTAX - Rules for proper constuction of parts of BASIC statements.
TWO'S COMPLIMENT - A method of binary representation of numbers where negative numbers are represented as the result of subtracting the absolute value of the number from zero.

A/BASIC PROGRAM STRUCTURE

An A/BASIC program consists of one or more source lines. Each source line must begin with a line number which is a positive number of up to 4 digits. Line numbers on succesive statements must be in ascending order without duplication.

A source line may contain one or more BASIC statements. If more than one statement is included on a line a colon : character is used to seperate them. A line may contain up to 72 characters. Blanks (spaces) may be used to improve readability but are not significant to the compiler except when included in character strings enclosed in quotes.

Examples of BASIC source lines:

```
 100 A=B+C : D=E*F
 200 GOTO 1020
 300 N=N+1 : GOSUB 520 : IF A<B THEN 200
```

Basic Statements

A/BASIC statements are grouped into five categories:

ASSIGNMENT STATEMENTS are used to assign a variable a value which is the result of an evaluation of an expresssion.

CONTROL STATEMENTS are used to alter the normal sequence of program execution, which may be dependent on some condition.

INPUT/OUTPUT STATEMENTS are used to transmit and accept data from peripheral input/output devices.

REAL-TIME AND SYSTEM CONTROL STATEMENTS allow the program to interact with the system environment and RT/68 operating system.

COMPILER DIRECTIVE STATEMENTS do not directly produce any corresponding machine language, but rather control the compilation process and compiler operation.

A section of this manual is devoted to each category listed above in which detailed definitions of each statement appear.

ARITHMETIC OPERATIONS

Numbers

A/BASIC's numeric data type is internally represented as 16-bit (2 byte) 2's compliment integers. Therefore the basic range in decimal of a number is -32768 to +32767.

A negative number is preceded by the minus sign. Numbers may also be used in hexadecimal form in which case the hex digits are preceded by a dollar sign (not to be confused with strings!) and the range in hex is therefore -$8000 (-32768 decimal) to $7FFF (32767 decimal).

Because A/BASIC supports logical operations and memory access that may require 16 bit UNSIGNED integers it will also consider decimal numbers up to 65535 and hex $FFFF to be legal - it is the context that is important.

Examples of legal numbers:

```
    200   -5000  $0100  $FE00 -$200  $C  -1
```

Some ILLEGAL numbers:

```
    99999 (too big)  -$FFFF(out of range)  +20 (plus sign illegal)
```

Because of the way 2's compliment numbers are represented, the numbers -1, $FFFF, and 65535 all have the same binary representation.

Variables

A numeric variable in A/BASIC has a name which consists of a single letter or a single letter and a digit. Two bytes of RAM are allocated for each variable.

Examples of legal variable names:

```
  X  N  R2  A9  Z3  F
```

Arrays

Numeric array may have one or two dimensions (subscripts). The maximum subscript size is 255. Array names are the same as variable names. Before an array variable is used, it must be declared by means of the DIM statement.

When array variables are used, the subscripts may be expressions but must not be: negative, zero, or greater than 255. The object program does NOT check subscripts when they are calculated so if they are out of legal range they may destroy other data or the program.

Wherever possible use one-dimensional arrays as they may be accessed fastest. Though finding an address of a two-dimensional array requires a multiplication, A/BASIC will use a special fast 8-bit multiplication subroutine.

When a variable is declared as an array but used in the program without subscripts, it will be implied that it is the first element of the array. for example, if an array X(20,20) is declared, the compiler will interpret X+X to mean X(1,1)+X(1,1).

Examples of legal usage of arrays:

```
   X(N*2)   V4(3,N)   A(B(N+1))   S5(2,6)   W(N*2,(A+B)*C)
```

Arithmetic Operators

There are five legal arithmetic operators which may be used in arithmetic expressions:

```
 + add
 - subtract
 * multiply
 / divide
 - (unary) negative
```

Arithmetic operations may produce errors due to overflow and underflow which may be tested (see EXTENDED ARITHMETIC OPERATIONS).

Logical Operators

The four logical operators perform bit-by-bit boolean functions on their operands and are useful for bit manipulation and testing, masking, etc. They may be mixed in numeric expressions with arithmetic operators as required.

```
 & - AND
 ! - OR
 % - EXCLUSIVE OR
 # - (UNARY) NOT
```

Order of Operations

Arithmetic expressions are evaluated on the basis of operator precedence where operators are grouped into priority classes. As the expression is scanned from left to right by the compiler, the expression will be evaluated by processing operators with the highest priority first, etc. Parentheses may be used to alter the order of evaluation.

PRIORITY	OPERATIONS
1	FUNCTIONS
2	UNARY - AND #
3	& ! %
4	* /
5	+ -

Arithmetic Expressions

Arithmetic expressions may range from a single constant or variable to a long formula with many operators and operands. The compiler will attempt to rearrange the expression for optimum code generation wherever possible. If temporary storage of intermediate results is required, the machine stack is used. The compiler will generate the shortest instruction sequence when array references and function calls follow the natural order of operations.

Examples of legal expressions:

```
  A    25    A*B    C+(D-F)   $200*N   C+(D&F*-$200/(RND/2)!N(X,Y))

  A%(B&$FF00)   2*M3    A(N+1)+A(N+2)* B(N,M)   #$40*-100
```

Arithmetic Functions

The following arithmetic functions are supported by A/BASIC:

ABS(<expr>) - absolute value of <expr>
POS - current position of output pointer in I/O buffer
CLK - RT/68 real time clock current time
PEEK(<address>) - contents of memory byte at <address>
RND - Random number from 0 to 32767. If a smaller range is desired, divide RND by 32768/(largest number desired) For example, to obtain a random number 0<=N<10 use (RND/3277). If an expression in parentheses is used with RND the value of the expression will be used as a new random number 'seed'.

Several functions that use string arguments return numeric values. See the section on STRING FUNCTIONS.

Extended Arithmetic Operations

A/BASIC has statements that allow error detection and recover for arithmetic error as well as permitting multiple-precision arithmetic. An overflow resulting from a multiply or addition will set the C (carry) bit in the MPU condition code register. An underflow from a subtraction (borrow) or attempt to divide with a zero divisor will produce the same effect. Two statement which are identical in operation may be used to detect these circumstances:

```
    ON ERROR GOTO <line #>      and
    ON OVR GOTO <line #>
```

These statements will jump to the line number specified if the carry bit is set AS A RESULT OF THE LAST ARITHMETIC OPERATION of an expression. If more than one operation must be tested, they may be written as successive expressions.

A statement which has the opposite effect, i.e., jump if NO overflow/ error occured is:

```
    ON NOVR GOTO <line #>
```

Multiple-Precision Arithmetic

If numbers greater than the 16-bit range of A/BASIC's numeric type must be processed, two or more 2-byte segments of larger numbers may be stored, operated upon, and tested by treating several A/BASIC variables as one larger number. In this way operations may be performed on numbers represented as multiples of 16 bits, i.e., 32, 48, 64 ... bits.

For addition and subtraction, the ON OVR GOTO and ON NOVR GOTO statement may be utilized. For example, assume two 32 bit numbers are stored in variable pairs A1, A2 and B1, B2 respectively. A1 and B1 shall be the variables containing the most significant halves of each value. The following subroutine may be used to add them:

```
 300 A2+A2+B2 : ON NOVR GOTO 310 : A1=A1+1
 310 A1=A1+B1 : RETURN
```

To subtract B from A the following subroutine is used:

```
 400 A2=A2-B2 : ON NOVR GOTO 410 : A1=A1-1
 410 A1=A1-B1 : RETURN
```

Similar routines may be written for multiplication and division. If an overflow occurs on a multiply, the multiply subroutine will return the least significant 16 bits of the result and place the high-order 16 bits of the product in the fast scratchpad area at addresses $002B and $002C.

The division routine does not preserve the remainder, if any, but because of the integer arithmetic the remainder of A/B can be found by evaluating A-A*(A/B).

STRING PROCESSING

Character Strings

The string data type is used for operations that involve processing text represented as variable length strings of characters. The data type allows use of all characters in the ASCII code except the NULL character ($00) which is used as an internal end-of-string delimiter.

String Constants

A string constant consists of any number of characters enclosed in double quote (") symbols. It may not include the carriage return symbol ($0D). If a double quote is to be part of the string, two adjacent to each other will be interpreted as one double quote and not end-of-string. Spaces within a string are considered characters. Examples:

```
   "A VERY SHORT STRING"
   "A DOUBLE QUOTE "" WITHIN"   (read as A DOUBLE QUOTE " WITHIN)
```

String Variables

String variable names consist of a single letter and a dollar sign so there are 26 possible string variable names. One-dimensional string arrays may be declared using the DIM statement. All subscripting rules for numeric-type arrays also are applicable to string arrays, including the characteristic that a string variable name declared as an array but used without a subscript is assumed to refer to the first element. For example, if an array A$(10) is declared and A$ is referenced without a subscript, the compiler will assume A$(1).

A string variable is allocated 32 bytes of RAM so it may contain up to 32 characters. The data is internally represented as left-justified (first character at lowest address) with a zero-byte terminator if less than 32 characters long.

String Expressions and Concatenation

When string variables are used in an A/BASIC program the compiler will allocate a minimum of 256 bytes of RAM for use as a string operations buffer. When string operations are performed the strings are moved to and operated on in this buffer. Because the string buffer is the last RAM allocated by the compiler if more memory is available beyond 256 bytes it can automatically be used for additional string working storage.

The basic operator in string expressions is the + symbol which represents concatenation. This essentially is the function of joining the left string (constant, variable or function) with the right string (constant, variable, or function) to form one string.

String Expressions and Concatenation - Continued

Several strings may be joined in an expression by using + operators in the expression. While processing a string expression, the string buffer typically fluctuates in size according to how long intermediate results of the evaluation are.

String expressions are evaluated from left to right, and no parentheses or operators other than + or functions are permitted.

The result of a string evaluation leaves the string result in the buffer which can be moved into memory in the case of a string assignment statement. If the result is to be stored as a string variable, up to the first (left) 32 characters will be stored. In the case of the special string variable BUF$, up to 128 characters will be stored.

String Functions

There are ten string functions grouped into two classes. The first class return a NUMERIC VALUE and have as arguments string expressions. The second class return STRING VALUES and have as arguments string and/or numeric expressions. The second class are identified by having a dollar sign as part of the function name.

Class 1 Functions -

ASC(<str expr>) - returns a number which is the numeric ASCII code for the first character of the string expression. An empty string returns 0.

LEN(<str expr>) - returns the length of the string expression up to 255 characters. An empty string returns 0.

SUBSTR(<str expr>,<str expr>) - searches for the first occurance of the first string within the second string, in other words, searches for the first string in the second string. If it is not found, a value of 0 is returned. If it is found, the number returned is the character position whre the beginning of the substring was found. For example:

```
SUBSTR("EXAM","AN EXAMPLE")  will return 4
if A$="CAT" then  SUBSTR(A$,"ANIMALS") will return zero
```

VAL(<str expr>) - returns the numeric representation of a number represented as a string of ASCII characters, in other words, a string to numeric conversion. The conversion starts at the beginning of the string, and leading spaces are skipped. The number may be positive or negative in the range -32768 or +65535. The conversion ends when 5 digits are read, or a blank, comma or end-of-string are reached. If the string is not ASCII symbols for digits or any error occurs, a value of zero is returned. Examples:

```
VAL("123,456") returns 123 numeric
VAL("-10000000)  returns zero
VAL("TEN") returns zero
```

Class 2 String Functions:

CHR$(<expr>) - returns the ASCII character corresponding to the number which is the result of the evaluation of the numeric expression.

LEFT$(<str expr>,<expr>) - returns the leftmost N characters of string expression. If the length of the string is less than N the entire string is returned. (N is the result of evaluation of <expr>).

MID$(<str expr>,<expr 1>,<expr 2>) - returns the middle part of the string <str expr> beginning from character position <expr 1> and including <expr 2> characters from that point. If <expr> is greater than the number of characters in <str expr> or <expr 2> is <= 0 a null string is returned. If there are not enough characters in <str exp> to satisfy the function, all characters from <expr 1> to the end are returned.
Examples:

```
MID$("DEMONSTRATION",3,5) returns "MONST"
MID$("LITTLE",10,5) returns ""
MID$("TOO SHORT",5,20) returns "SHORT"
```

RIGHT$(<str expr>,<expr>) - returns the right <expr> characters of the string <str expr>. If <expr> is greater than the number of characters in the string, the entire string is returned.
Example:

```
RIGHT$("DEMONSTRATION",4) returns "TION"
```

STR$(<expr>) - returns a string consisting of the ASCII characters that represent the numeric-to-string conversion of <expr>. If <expr> is negative a leading minus sign is included.

TRM$(<str expr>) - returns the string argument less any trailing blanks (spaces).
Example:

```
TRM$("SPACES      ")  returns "SPACES"
```

NOTE - IN ALL STRING FUNCTIONS THAT HAVE A NUMERIC ARGUMENT(S) EXCEPT STR$ THE VALUE OF <expr> MUST BE IN THE RANGE 0 TO 255. THE STRING ARGUMENT(S) MAY NOT EVALUATE TO A STRING WITH MORE THAN 255 CHARACTERS.

Null Strings

An important part of string processing involves the use of null or "empty" strings - string constants or variables that contain no characters. This is represented as "". To initialize a sting variable, an assignment A$="" is analogous to the numeric operation A=0. This is important because strings are variable length with end of strings terminated by a zero byte. A/BASIC does not initialize wariables to zero or null so initialization of certain ones may be required, depending on the program.

String Operations on the Input/Output Buffer

A very powerful feature of A/BASIC's string operations is the ability to perform string manipulations on the 128-byte input/output buffer. The special string variable BUF$ may be used anywhere a string variable is legally used. Additionally, special forms of the four input/output statements are available to read or write the buffer:

```
 PRINT BUF$
 INPUT BUF$
 TREAD BUF$
 TWRITE BUF$
```

Any of the above statements will read/write up to 128 characters to/from the terminal or cassette system and deposit them in the input/output buffer BUF$ without any change. Also, user-written subroutines in A/BASIC or machine language may read or write data to/from BUF$ to devices not supported by A/BASIC directly, such as disks, tape devices, memory-mapped video displays, etc. BUF$ is also useful for formatting data or reading unformatted data when standard BASIC I/O statements will not operate properly, such as reading strings that have imbedded commas, etc.

Note that BUF$ is not the same buffer as the string buffer so references to BUF$ in expressions do not disturb data in it. Though BUF$ is used as a string variable, it is not limited to 32 character string lengths - the entire 128 byte buffer is used.

Data in BUF$ is modified only when:
1) another input/output statement is executed
2) BUF$ is given as the result of a string assignment statement, for example: BUF$=A$+RIGHT$(B$,N)

Examples of BUF$ Usage:

Adding a string to the end of the buffer-
```
 BUF$=BUF$+A$
```

Removing the last character from BUF$-
```
 BUF$=LEFT$(BUF$,LEN(BUF$)-1)
```

Read a line from the terminal and print it backwards:

```
100 INPUT BUF$
200 N=LEN(BUF$)
300 FOR K=1 TO N
500 BUF$=RIGHT$(BUF$,1)+MID$(BUF$,2,N-1)
600 NEXT K
700 PRINT BUF$
```

String Compare Statement

Syntax: IF <str expr> <str relation> <str expr> THEN <line #>

Description: The first string expression is tested for equivalence against the second string expresssion. For the two string expresssions to match, they must be of the same length and include the same characters exactly including spaces.

Two string relationals are permitted: = equal to and <> or >< not equal to. Use of any other relationals will result in an error. If the relation is true, the line number specified will be executed.
Examples:

```
  IF A$=B$ THEN 200
  IF A$(N+1)+"END"<>C$+LEFT$(B$,S) THEN 1040
  IF A$ = "MATCH" THEN 2305
```

Code Generation for String Operations

If string operations are used by a program, the linker/editor will select string subroutines in BASLIB for inclusion in the program. For simple string assignments such as A$=B$ the variables are transfered directly from memory to memory and never enter the string buffer. When string operations that involve one or more levels of pending operation (such as LEFT$, MID$, etc.) the string values are moved to the string buffer as they are encountered in the source line. Pointers to the beginning position of each unprocessed string value are pushed onto the system stack when entered and pulled when ready to process, so the string buffer acts as a "string stack". The string buffer may grow to a maximum size when many string operations are pending such as:

```
 LEFT$(A$+RIGHT$(B$+MID$(C$+D$,N,M),X),Y)
```

In this example, A$,B$,C$ and D$ will be moved to the string buffer at the point the MID$ operation is finally processed. This would require 32*4 or 128 bytes of buffer to store all strings, worst case. Because the compiler allocates a 256 byte string buffer, almost any expression can be accomodated EXCEPT when several operations are pending which use BUF$ for an operand, which may contain up to 128 bytes. In such a case it is desireable to use multiple statements for BUF$ operations so never move than 1 is pending OR allow extra room at the end of variable storage for the string buffer to expand.

Calculation of addresses of string variables which are arrays requires a multiplication. To generate code to address string array elements, the linker will edit in subroutine calls to the BASLIB fast 8 by 8 register multiply subroutine which can calculate the absolute address at run-time in less than 200 MPU cycles. (The same subroutine is used for 2-dimension numeric arrays).

ASSIGNMENT STATEMENTS

Arithmetic Assignment Statements

```
Syntax:  LET <var> = <expr>
         <var> = <expr>
         POKE <address> = <expr>
```

Description: The arithmetic assignment statement evaluates the expression <expr> and places the result in <var> which may be a simple numeric variable or an indexed numeric variable, in which case the subscript(s) may also be expressions. Use of the LET keyword is optional and in no way affects the operation.

If the assignment statement is specified as a POKE, the low-order byte of the result is stored at <address>. The high-order byte is ignored. Note - the two MPU accumulators ACCA and ACCB always contain the result of the expression (ACCA is MS byte) after execution of the statement. If the result destination was indexed, the address of the array element will be in the MPU index register. This information may be useful when calling machine language routine where numeric data is to be passed.

String Assignment Statement

```
Syntax: <str var> = <str expr>
```

Description: The string expression is evaluated, the result of which is the final contents of the string buffer. The leftmost 32 characters (or less, if the result is less than 32 characters) are moved to <str var> which may be simple or indexed.

A special form of the string assignment, BUF$=<str expr> is described in the STRING OPERATIONS section.

CONTROL STATEMENTS

Call Statement

Syntax: CALL <address>

Description: The CALL statement is used to directly call a machine-language subroutine at the address specified. The subroutine will return to the BASIC program if it terminates with an RTS instruction and does not disturb the return address on the stack.

```
Examples:  CALL $E0CC        Call subroutine at address $E0CC
           CALL 1024         Call subroutine at decimal addr. 1024
```

For/Next Statement

```
Syntax:  FOR <var> = <expr> TO <expr> [STEP <expr>]
         NEXT <var>
```

Description: The FOR/NEXT uses a variable <var> as a counter while performing the loop delimited by the NEXT statement. If no step is specified, the increment value will be 1. The FOR/NEXT implementation in A/BASIC differs slightly from other BASIC due to a looping method that results in extremely fast execution and minimum length. Note the following characteristics of FOR/NEXT operation:

1. <var> must be a non-subscripted numeric variable.
2. The loop will be executed at least once regardless of the terminating value.
3. After termination of the loop, the counter value will be GREATER than the terminating value because the test and increment is at the bottom (NEXT) part of the loop.
4. FOR/NEXT loops may be exited and entered at will.
5. At compile time, up to 16 loops may be active, and all must be properly nested.
6. The initial, step, and terminating values may be positive or negative. The loop will terminate when the counter variable is greater than the terminating value.

```
Examples:  FOR N = J+1 TO Z/4 STEP X*2
           FOR A= -100 TO -10 STEP -2
```

Gosub/Return Statements

```
Syntax:  GOSUB <line #>
         RETURN
```

Description: The GOSUB statement call a subroutine starting at the line number specified. If no such line exists, an error message will be generated on the second pass. The machine stack is used for return address linkage. The RETURN statement terminates the subroutine and returns to the line following the calling GOSUB. Subroutines may have multiple entry and return points. THe GOSUB and RETURN statements compile directly to JSR and RTS machine instructions, respectively.

If/Then Statement

Syntax: IF <expr> <relation> <expr> THEN <line #>
        IF <expr> <relation> <expr> GOSUB <line #>

Description: The IF/THEN or IF/GOSUB is used to conditionally branch to another statement or conditionally call a subroutine based on a comparison of two expressions. For operation with strings, see the STRING PROCESSING section. Legal relations are:

```
 < - less than
 > - greater than
 = - equal to
 <> or >< - not equal to
 <= or =< less than or equal to
 >= or => greater than or equal to
```

IF the statement is an IF/GOSUB the subroutine specified will be called if the relation is true and will return to the statement following. Because of A/BASIC's multiple statement line capability, the IF statement can be used as an IF..THEN..ELSE function if another statement follows on the same line.

```
Examples:  IF N<= 100 THEN 1210
           IF A+B=C*D GOSUB 5500
           IF X < 200 THEN 240 : GOTO 1100
```

On Condition Statements

See ARITHMETIC OPERATIONS section.

On-Goto/On-Gosub Statements

Syntax: ON <expr> GOTO <line #>,<line #>, ... ,<line #>
         ON <expr> GOSUB <line #>,<line #>, ... ,<line #>

Description: The expression is evaluated and one line number in the list corresponding to the value is selected for a branch or subroutine call, i.e., if the expression evaluates to 5, the fifth line number is used. If the result of the expression is less than or equal to zero, or is higher than the number of line numbers specified, the next statement is executed.

```
Examples:  ON A*(B+C) GOTO 200,350,110,250,350
           ON N GOSUB 500,510,520,500,100
```

Stop Statement

Syntax: STOP

Description: The STOP statement is used to terminate execution of a program by causing a jump to the entry point of the RT/68 console monitor. The END statement should not be confused with STOP as STOP will not terminate compilation of the program.

INPUT/OUTPUT STATEMENTS

All input and output statements use a 129-byte buffer for intermediate storage of data. This buffer may contain up to 128 characters. The buffer is automatically allocated by the compiler after allocation of all other memory space except the string buffer (if used). This buffer is only allocated for programs that use input/output.

Character level input/output subroutines used are those provided by the RT/68 I/O system. Vectored I/O through an ACIA-type interface at any address may be accomplished by storing the address of the ACIA at RT/68's vector location. Refer to the RT/68 systems manual for details.

NOTE: A special form of all input/output statements designed for buffer direct input/output using the special string variable BUF$ is described in the STRING PROCESSING section.

Input Statement

Syntax: INPUT <var>, ... ,<var>

Description: This statement causes code to be generated which prints a $ prompt and space on the terminal device, then reads characters into the input buffer until 128 characters have been read or a carriage return symbol is read. A carriage return/line feed is printed when the last character in input.

At run-time, entry of a CONTROL X will print *DEL* and CR/LF and reset the buffer. A CONTROL O will backspace in the buffer and echo the deleted characters.

The variables specified <var> may be numeric or string, subscripted or simple type. When the program is "looking for" a number from the current position in the input buffer, it will skip leading spaces, if any and read a minus sign (if any) and up to five number characters. The numeric field is terminated by a space, comma, or end of line. If a non-digit character is read, or any other illegal condition a value of zero will be returned for the number.

If a string-type field is being processed, up to 32 characters from the current position will be accepted including blanks, if any. The string field is terminated by a comma or end of line, or when 32 characters are read. If no characters are available, a null string will be returned.

Examples:

```
   INPUT A,B,S$,B$
   INPUT A(N+1,M-1),B,A(4,N)
   INPUT A$(N),B$(N+1),D$
   INPUT B
```

Print Statement

Syntax: PRINT <out spec><delimiter> ... <delimiter><out spec>

Description: This statement processes the list of <out spec>'s and puts the appropriate characters in the buffer. The buffer is then output to the terminal device.

An <out spec> may be a string expresssion or a numeric expression, or the output function TAB<expr> which inserts spaces in the buffer until the position <expr> is reached. Each item in the list is seperated by a delimiter which is a comma or semicolon. The buffer is divided into sixteen 8-character zones, which are effectively tab stops every eighth position. If a comma is used as a delimiter, the next item will begin at the first position of the next zone. If a semicolon is used, NO spacing will occur. A semicolon at the end of a Print statement will inhibit printing of a carriage return/ line feed at the end of the line.

Examples:
```
   PRINT A,B;C
   PRINT A$(N);A$(N+1)
   PRINT A,A$,B,B$
   PRINT TAB(N=1),Z4
   PRINT A;B;C;
   PRINT A$;TAB(N+M);B$
```

Tape Input/output

Syntax: TREAD <IO list>
        TWRITE <out spec><delimiter> ... <delimiter><out spec>

Description: The tape I/O statements are used to write data records on audio cassettes. The format, syntax, and functions of TREAD and TWRITE are identical to INPUT and PRINT respectively in regards to the way data is placed in/extracted from the I/O buffer. The major difference is in the way the record is written to or read from tape.

The tape record format is:
<90 NULLS><$02><UP TO 128 DATA BYTES><$03>

During execution of TREAD, a READER ON control character is transmitted and a loop is entered to ignore characters until a $02 start-of-record character is read. All characters following are read and placed in the input buffer until either 128 bytes have been read or a $03 end-of-record character is read. A READER OFF control character is transmitted and the input buffer then processed in a manner identical to the INPUT statement.

The TWRITE statement fills the I/O buffer in a manner identical to the PRINT statement. A PUNCH ON control character is transmitted, then a leader of 90 nulls to allow for motor on/off time, a $02 start-of-record character, the up to 128 character contents of the I/O buffer, and the $03 end-of-record character followed by a PUNCH OFF control character.

COMPILER DIRECTIVE STATEMENTS

Base Statement

Syntax: BASE = <address>

Description: The A/BASIC has two internal "pointers" used during compilation to control memory allocation: a "data" pointer used to keep track of the next available RAM location for assignment of storage for variables and arrays; and a "program counter" used to locate the address of the current machine instruction being generated. The BASE statement is used to set or change the data pointer. The ORG statement controls the program counter and is described below.

These two statements are critical for correct execution of the object program because they define where data (variables and arrays) are stored and where the program is stored. These two areas MUST NOT ever overlap.

During compilation, RAM storage is assigned for variables at the time they are first encountered in the program. The data pointer is used for the address for the variable, then the size of the storage required is added to it. Two bytes are required for each numeric-type variable or each element of a numeric array, and thirty-two bytes for each string variable or element of a string array.

After the last statement of the program has been compiled, the data pointer is used to allocate storage for a 129-byte input/output buffer (if the program uses input/output statements) and/or a 256 byte string buffer (if the program uses string operations). Though a 256-byte minimum string buffer is specified, it may actually be larger (see STRING OPERATIONS).

At the beginning of compilation, the data pointer is assigned an initial default value of $0030. Memory below this address is reserved for operating systems and run-time fast scratch storage.

Because more than one BASE statements may be used in an A/BASIC program, it may be used to specify RAM variable storage blocks, assign specific variable names to specific addresses (which may even be addresses of peripheral control and data registers.

IMPORTANT - Because the compiler will select the 6800 direct addresssing mode (2 byte instructions) wherever possible, numeric variables should be allocated storage at addresses $0030 through $00FF whenever possible. String variables and arrays may be located elsewhere. This technique may reduce program size by as much as 40 percent!

This is most easily accomplished by using a BASE statement at the beginning of a program to set the data pointer to a "high" memory address, then using DIM statements to declare all arrays and string variables or other variables to be assigned specific addresses, and then using the BASE statement again to set the data pointer to "low" memory.

Base Statement - Continued

Note that because A/BASIC considers variable X to be the same as array element X(1) the DIM statement may be used to declare "simple" variables by defining them as one-element arrays.

Where programs are long and many variables are used, "trial" BASE values may be used and the program compiled with the OPT S option selected so the symbol table is printed with the exact variable storage assignments. The exact values for the BASE statements may used in the next compilation.

In the following example, several uses of the BASE statement are shown. The example program is set up to:

1) Assign storage for variables reference in the program from addresses $0030 to $00FF.
2) Assign array and string storage above $0100
3) Assign the variable name P1 to a peripheral register at $8010
4) Assign the variable name A1 to a peripheral register at $8020

```
050 OPT S : REM ENABLE SYMBOL TABLE

100 BASE = $0100
200 DIM X(10),Y(20,4),Z3(4,100) : REM DECLARE NUMERIC ARRAYS
300 DIM S$(20),T$(10),U$(4) : REM DECLARE STRING ARRAYS
400 DIM A$(1),B$(1),C$(1),D$(1) : REM "SIMPLE" STRING VARIABLES
500 BASE = $8010 : DIM P1(1) : REM DECLARE PERIPHERAL
600 BASE = $8020 : DIM A1(1) : REM DECLARE PERIPHERAL
700 BASE = $0030 : REM NORMAL ALLOCATION STARTS HERE
800 REM REMAINDER OF PROGRAM FOLLOWS .....
```

Dimension Statement

Syntax: DIM <array spec> ... ,<array spec>

Description: This statement is used to declare arrays. All arrays must be declared before they are referenced in the program. Arrays may not be redefined. The maximum size for a subscript is 255. One or two-dimensional numeric arrays are permitted, and one dimensional string arrays may be declared. Each element of a numeric array requires two bytes of RAM storage, and each element of a string array requires thirty-two bytes of storage.

The DIM statement may be used in conjunction with the BASE statement to declare "simple" variables, or allow the same memory to be "shared" by different sized or named arrays.

```
Examples:  DIM A(10),B(3,8),N(255),B3(40)
           DIM B$(5),T$(100)
           DIM A(4),A$(4)
           DIM X($20)
```

End Statement

Syntax: END

The END statement is used to indicate the last statement of the source program. Compilation ceases for the pass when the END is encountered. Only one END statement is permitted, and it must be the last statement in the program. If a jump to the RT/68 monitor is desired at the end of the program, a STOP statement should precede the END. If end-of-file is read from the source input device before an END is encountered, the compiler will assume an END.

Options Statement

Syntax: OPT <option>, ... ,<option>

Description: The OPT statement is used to activate options which affect the compilation process. The option symbols and functions are:

I - Inhibit generation of object file
H - Print hexadecimal instruction field in listing
N - No listing
S - Print symbol table

All options are not selected by default. If OPT H is selected, the compiler will print one or more lines of three columns of hex data after each source line. Each of these lines correspond to a machine instruction generated by the compiler. The leftmost column is the absolute address, the next column is the opcode, and the rightmost column is the operand (if any).

If the S option is selected, a dump of the compiler's symbol table is printed after the end of the second pass. The leftmost column is the symbol, the next columns are the absolute memory address, array row length (if array) and array column length (if array), respectively. If applicable, several "internal" symbols will be included to represent compiler-generated storage assignments. They are:
IO - Input/output buffer (129 bytes)
RR - Random number temp. (2 bytes)
ST - String buffer (256+ bytes)
ZZ - Index calculation scratch (2 bytes)
Any symbol with an asterisk - for/next loop terminate or increment temporary storage (2 bytes)

Examples: OPT S
OPT I,N,S

Program Origin Statement

Syntax: ORG = <address>

Description: The ORG statement is used to assign a value to the compilers internal 'program counter' which is the address at which the compiler will generate the next instruction. This statement may be used more than once if it is desired to put different portions of the program at different memory areas. If used at the start of a program, it establishes the first address of the program. If no ORG statement is included, a default value of $1000 is supplied by the compiler which is the normal starting address.

It is extremely important that the program counter does not ever cause instructions to be located at areas assigned for variable storage. Note that if default values for BASE and ORG are used, a little under 4K bytes of storage for variables is available before any overlap occurs. See the BASE statement description for more information.

Note that if the ORG is used in the middle of a program to change the program counter (as might be required if a PROM boundary is reached, for example) a GOTO may be required before the ORG to jump to the succeeding program segment.

```
Examples: ORG=$7000
          ORG = 8192
```

Remark Statement

Syntax: REM <text>

Description: This statement is used to include comments in the program. On multiple statement lines any REM must be the last statement on the line, or following statements will be ignored. Use of REM statements will not increase the length of the object program generated.

```
Examples:  REM A COMMENT GOES HERE
           REMARKABLE PROGRAMMING!
```

REAL-TIME AND SYSTEM CONTROL STATEMENTS

Gen Statement

Syntax: GEN <number>,<number>, ... ,<number>

Description: The GEN statement allows data or machine language instructions to be directly inserted in the program. The list of values supplied are inserted directly into the object program. If a value given in the list is less than 255 one byte will be generated for that value regardless of leading zeros.

```
Example:  GEN $BD,$E141,$CE,1024  (PRODUCES 6 BYTES)
          GEN 0040,$00,32767  (PRODUCES 4 BYTES)
```

IRQ On/Off Statements

```
Syntax:  IRQ ON
         IRQ OFF
```

Description: These statements are used to control the state of the MPU interrupt mask flag in the condition code register and are used to enable/disable interrupt recognition. These statements correspond directly to the 6800 CLI and SEI instructions. Refer to the RT/68 Systems Manual and the M6800 Programming guide for specifics on interrupt processing.

On Interrupt Statements

```
Syntax:  ON IRQ GOTO <line #>
         ON NMI GOTO <line #>
```

Description: These statements are used for generating programs to be used in the RT/68 SINGLE TASK MODE (non-multiprogramming) where interrupts are processed as vectors to specific service routines. When encountered in a program these statements cause the absolute address of the program corresponding to the line number specified to be stored at the interrupt vector addresses in the operating system scratchpad ($A000 for IRQ and $A006 for NMI).

The line number specified should be the beginning of the interrupt service routine which would typically service the device causing the interrupt. This routine is similar to a BASIC subroutine except it is terminated by an RETI (return from interrupt) statement instead of a RETURN statement.

```
Examples:  ON IRQ GOTO 2010
           ON NMI GOTO 400
```

Interrupt Return Statement

Syntax: RETI

Description: This statement terminates an interrupt-caused routine by loading the MPU register contents prior to the interrupt from the machine stack and resuming program execution from the point where the interrupt was acknowledged. This statement corresponds directly to the machine language RTI instruction.

Stack Assignment Statement

Syntax: STACK = <address>

Description: This statement is used to initialize or change the MPU stack pointer register. This is typically one of the first statements in an A/BASIC program if used. If a STACK statement if not included in the program, the operating system scratchpad stack ($A049 to $A016) will be used by the program for the machine stack. This is adequate for most programs that do not: nest subroutines extensively; use interrupts; run in the multiprogramming mode; or process elaborate arithmetic expressions. Otherwise, a specific memory area should be dedicated for the stack and the STACK instruction used to load the stack pointer with the TOP (highest address) of the stack.
Example: STACK = $A042

Switch Statement

Syntax: SWITCH

Description: This statement is used in the RT/68 MULTITASK MODE to call the executive, thereby causing the task to be suspended while other tasks may run. The task selection/switching process is described in section 4 of the RT/68 Systems Manual. The machine instructions generated by this statement include operations that set the MPU interrupt mask and the RT/68 task switch flag byte (RELFLG) and a software interrupt (SWI) which activates the executive. When the task is run again, execution resumes at the statement following the SWITCH statement.

Task On/Off Statements

```
Syntax:  TASK <task #> ON
         TASK <task #> OFF
```

Description: When used in the RT/68 MULTITASK MODE this statement will find the task status byte of the specified task in the status table and either set or clear the task state bit. This causes the task to be either activated (able to run) or disactivated (may not run) when the executive is selecting a task to execute. Note that this statement does not directly cause the task to start/stop- it merely enables/disables it from consideration when the executive is searching for a task to run. A task may use the TASK N OFF to turn itself off and become dormant. This is often used when the task was activated by another to perform a specific function and has completed its operation.

COMPILER OPERATION

PASS 1: During the first pass the entire source program is compiled in a "dummy compile" that performs a complete syntax analysis and pseudo-code generation to build the symbol table and line reference table. At the end of pass one the addresses will have been fixed in the line number table and symbol table.

As the first pass is performed references to subroutine calls to BASLIB routines are noted by setting individual flags for each routine required. At the end of pass 1 the linker/editor will allocate storage for all required subroutines at the end of the main object program, and allocate variable storage for any temporary variables or buffers required. These addresses are therefore "fixed" at the end of the first pass.

PASS 2: During the second pass the code generation system will produce object code. All addresses of forward and backward references are obtained from the line reference table, and subroutine (BASLIB) addresses and buffer addresses are obtained from the linker system. After the main object program is complete, the linker is called again to copy images of subroutines selected from BASLIB and append them to the end of the object program. Note that all routines contributed by BASLIB are "packed" at the end of the object program and one and only one copy of each is included.

Compilation Techniques: A/BASIC is extremely short primarily due to use of a sophisticated integrated scanner/parser/code generator that produces object code directly from source lines. Intermediate code is generated only for string and numeric expressions for optimization purposes. The basic parsing technique is a top-down, predictive system.

Code Generation: All code generation is designed to generate the fastest, shortest code possible. Instruction sequences and conventions take maximum advantage of the 6800 architecture and addressing modes. The machine stack is used extensively for fast intermediate storage. Arithmetic processing uses the MPU accumulators and hardware stack as a single virtual stack for optimum code generation. The arithmetic parser will attempt to rearrange expressions for shortest instruction sequences wherever possible.

The compiler will automatically select shorter direct addressing mode instructions wherever possible, and makes extensive use of short immediate mode instructions. The index register is used as an operand pointer, secondary stack pointer, string pointer, and for indirect addressing for arrays.

COMPILER OPERATIONAL PROCEDURES

The first time the compiler is to be used, it should be loaded and any modifications (SEE CUSTOMIZING A/BASIC) required for the specific system should be made. One or more backup copies of the compiler should be made by using the RT/68 DUMP command:

```
     D,0100,1BBF
```

Preparing an A/BASIC Program

The RTEDIT program on side 2 of the cassette is used to create/modify an A/BASIC program. It is loaded using the RT/68 LOAD function and started by entering:

```
    E,0100
```

It will print a question mark prompt and the program may be entered/edited. When the program is complete, it should be written on a tape using the "SAVE" or "END" functions. The editor is then exited.

The next step is to load the compiler program (side 1 of the cassette) using the RT/68 LOAD function. After the compiler is loaded, the tape (source file) is placed in the READ cassette machine and the cassette interface prepared for automatic read operation.

The compiler may be started by the RT/68 command:

```
    E,0100
```

and should immediately start reading the source cassette. Nothing will be printed during this first pass except error messages. At the end of the first pass, the compiler will stop reading the tape and print the message "PASS 2". If any error occured during pass 1, no further compilation should be attempted because the object program will be incorrect. The source file should be re-edited to change lines in error.

The source tape should be rewound and prepared to be read again. If an object tape is to be made, it should be placed in the RECORD cassette machine and advanced past the leader. The cassette interface should be prepared for automatic record and read. When both tapes are set up properly, enter any character on the terminal and the compiler will print a page heading and start to read the source tape. Each line will be printed and object records written on the object tape as required.

After the last source line is processed, the A/BASIC linker/editor program will append any subroutines required by the program to the end of the object program (without interrecord gaps).

Program statistics are then printed:

ERRORS - are the (decimal) number of errors occuring on the second pass
VAR LEN - is the (hex) number of bytes of variable storage allocated by the compiler
PGM LEN - is the (hex) length of the object program in bytes
PGM END - is the (hex) address of the last instruction of the object program.

If the OPT S option was selected, the symbol table will be printed in the order the variables were allocated storage by the compiler.

Error Messages: When an error occurs, the source line containing the error is re-printed. On the next line the compiler will print an up arrow symbol pointing to the position in the source line where either the error was detected OR the last recognized element of the source line before the error occured. This facility is not 100% accurate so the source line should be examined carefully. When an error occurs the processing of the rest of the line is terminated so following statements (if a multiple statement line) are NOT processed.

Listing Format: The first data on a source line is a four-character hex number which is the actual address where the line listed will have its corresponding machine instructions located. The source line is then printed exactly as read.

Loading and Running the Object Program

If no error occured during the compilation process, the object tape may be rewound and loaded using the RT/68 LOAD function just as any other machine language program. The starting address of the program will be $1000 unless an ORG statement was included at the beginning of the program that started it elsewhere.

Debugging the Object Program

Even if the program was compiled correctly it may not execute properly for one of three typical reasons:

1. The program is incorrectly written.
2. Memory was assigned incorrectly - data and program areas may be overlapped (see BASE statement description) or not enough memory is available.
3. Hardware/software relationships (initialization, addresses, operation, etc.) are incorrect, particularly when non-terminal input/output is used.

Very valuable information is contained on the listing and symbol table dump. The RT/68 BREAKPOINT function may be used to place breakpoints at the addresses of each BASIC source line (obtained from the listing) and the breakpoint will occur when the machine language corresponding to that statement is executed. Addresses of variable can be obtained from the symbol table and examined (in hex) to check their value at that point in program execution by using the RT/68 MEMORY EXAMINE/CHANGE or DUMP functions.

Advanced programmers may wish to select the OPT H option to obtain a full instruction listing and place breakpoints at the single instruction level if required.

A/BASIC RUN-TIME ENVIRONMENT

Memory Assignment

A/BASIC permits total control of memory allocation but two important areas must be preserved:

1) Operating System Area - memory at addresses $0000-$001F are reserved for operating systems and normally should not be used.

2) Run-Time Fast Scratch Storage - 16 bytes of memory at addresses $0020-$002F are used by subroutines in BASLIB for fast working storage. Any programs that use: strings, input/output, or multiplication must not overwrite this area. Memory assignments are:

$20	XR temp. for strings, I/O
$22	I/O buffer position pointer
$24	I/O buffer beg. address
$26	I/O buffer character count
$27	string buffer pointer
$29	string operation XR temp.
$2B	multiplication overflow word, string scratch storage
$2B-$2F	string misc. scratch

Operating System Interfaces

The RT/68 operating system must be present to compile in all cases, and present for the object program if it uses either input/output to tape or the terminal; or any of the real-time or interrupt capablilities.

If the program is to be run on another system, addresses of equivalent I/O subroutines must be patched into BASLIB before compilation, and no interrupt or real-time statements used in the program.

CUSTOMIZING A/BASIC

A number of operating characteristics of the compiler may be changed by "patching" memory locations with different values after the compiler is loaded into memory. A copy of the compiler may then be saved for later use.

Source listings of important I/O regions of the compiler are included for users who have special I/O requirements. These may be modified as desired as long as the subroutine length and addresses are not different. The source listing titled "BAS1.0" is the compile-time I/O system and the listing titled "BASLIB.1" is an "image" of subroutines which may be included in the program generated.

INTERRECORD GAP: The interrecord gap is a delay placed between tape data (or source) records to allow for tape motor start/stop. It is generated by transmitting a series of ASCII NULL (0) characters. This time is equal to the sum of the start and stop times Unless modified, 90 nulls are transmitted which is a 3 second gap at 30 CPS. The addresses of the counts are $17AC for run-time (BASLIB) and $0151 for compile-time. The count may be 0 to 255.

TABLE SIZES: If more than 8K of RAM is available the symbol table and line reference tables may be expanded to permit compilation of longer programs. The table below refers to the constants "X" at addresses $0114-$0115 and "Y" at addresses $011B-$011C which establish the table sizes.

MEMORY SIZE	NUMBER OF SYMBOLS	NUMBER OF LINE REFS.	TYP. PROGRAM LENGTH (LINES)	X	Y
8K	50	100	200-500	1ED2	1FFE
12K	250	824	2000-4000	2A22	2FFE
16K	400	1620	3000-9999	3692	3FF2

CASSETTE PORT CONTROL: A/BASIC as supplied will bring the control PIA (addr $8004-$8007) output CB2 "reader control" to a high state for either tape READ or WRITE to allow RT/68 to automatically switch ports as described in section 8 of the RT/68 Systems Manual. If the cassette interface used does NOT operate properly on RECORD with this signal change addresses $16F6 and $17A7 to $34.

IMPORTANT ADDRESSES:

```
TAPE RECORD BUFFER: 1C78 TO 1CF8
SOURCE LINE BUFFER: 1CF9 TO 1D41
OBJECT CODE BUFFER: 0090 (BYTE COUNT)
                    0091-0092 (LOAD ADDRESS)
                    0093-00AA (BINARY DATA BUFFER - NOT HEX!!)
I/O SUBROUTINE CALLS:
READ A SOURCE LINE (RETURNS W/CARRY CLEAR FOR E.O.F.): 0167
REWIND SOURCE FILE: 0102
DUMP OBJECT CODE BUFFER: 0625
JUMP TO RT68: 0826
```

RTEDIT

RTEDIT allows the user to enter or modify an A/BASIC source program. The program may use line numbers up to 9999. Each line can contain up to 64 characters. In an 8K system 50 program lines may be entered or edited at a time. For programs longer than 50 lines, repeating the LOAD/SAVE command sequence will read blocks of up to 50 program lines IN/OUT of memory at a time.

COMMANDS

NEW (cr)	Clears memory of all program lines.
RT68 (cr)	Returns user to RT68MX console monitor.
LIST (cr)	Prints all A/BASIC program lines in numeric ascending sequence.
L,<NNNN> (cr)	Lists A/BASIC program line, where <NNNN> is the one to four digit line number of the program line.
L,<BBBB>,<EEEE> (cr)	Lists A/BASIC program lines beginning at line number <BBBB> and ending at line number <EEEE>. <BBBB> and <EEEE> may be one to four digit line numbers and must be seperated by a comma.
MEM (cr)	Directs RTEDIT to print the number of program lines that can be added.
F/<SSS....S> (cr)	The "FIND STRING" function will cause RTEDIT to search the A/BASIC program lines for the first occurance of the character string <S>. String variable <S> may contain numeric, alphabetic, or special characters and can be a maximum of 32 characters. If a match is found the program line containing the match will be printed. If no match is made "NOT FOUND" will be printed.
LOAD (cr)	Causes RTEDIT to read a cassette tape of previously "SAVED" A/BASIC program lines. If the end of file character is read then "*EOF*" is printed.
<NNNN> < A/BASIC > statement	Entering a one to four digit line number followed by a space and an A/BASIC program line will cause the RTEDIT program to search the program lines in memory. If no line number equaling <NNNN> is found the program line will be added. If <NNNN> is found the current program line will replace the previous program line.
<NNNN> (cr)	Deletes program line number <NNNN>.

SAVE (cr)	This function writes all the program lines in memory on cassette tape. A/BASIC program lines will be written to tape in numeric ascending sequence. The tape will be written 128 bytes or 4 lines per record.
S,<BBBB>,<EEEE>	Program lines beginning at line number <BBBB> and ending at line number <EEEE> will be written to cassette tape. Program lines will be written on tape in the same format as "SAVE".
END (cr)	Performs function similar to "SAVE" except that the character "1A" is written as the last character on the cassette tape. Character "1A" signifies end of file (EOF).

NOTES

1. No line longer than 64 characters is allowed.

2. A line number of zero is not permitted.

3. For commands not recognized by RTEDIT the word "WHAT" is printed.

4. Cold start= E,0100 : Warm start= E,013F

5. CONTROL "X" entered before a carriage return will delete the current input line.
   CONTROL "O" entered before a carriage return will delete the last input line, character by character.

6. For users with more than 8K RAM memory the RTEDIT program can be easily modified and recompiled to allow the user to work with up to 127 program lines in memory at a time.

   For 12K system change :
   ```
   010 DIM N(100),B$(200)
   015 V=100 : FOR K=1 TO V : N(K)=0 : NEXT K
   ```

   For 16K system change :
   ```
   010 DIM N(127),B$(254)
   015 V=127 : FOR K=1 TO V : N(K)=0 : NEXT K
   ```

```
1000 01 REM PROGRAM : RTEDIT/V1/L1/12-77/TC ------------------------
1000 02 REM COMMANDS : RT,LI,ME,F/,LO,S,,SA,EN,NE,L,,(ADD,REP,DEL)
1000 03 REM COLD START = E,0100 : WARM START = E,013F
1000 04 REM
1000 05 REM INIT/NEW ----------------------------------------------
1000 06 OPT S
1000 07 ORG=$0100
0100 08 BASE=$1300
0100 10 DIM N(50),B$(100)
0100 12 BASE=$30
0100 15 V=50 : FOR K=1 TO V : N(K)=0 : NEXT K
0147 98 REM
0147 99 REM PICK COMMAND -----------------------------------------
0147 100 PRINT : INPUT BUF$ : Z$=LEFT$(BUF$,2)
017A 101 IF Z$<>"RT" THEN 103 : STOP
01A8 103 IF Z$="ME" THEN 900
01D3 105 IF Z$="LI" THEN 200
01FE 107 IF Z$="F/" THEN 800
0229 110 IF Z$="LO" THEN 600
0254 115 IF Z$="S," THEN 310
027F 117 IF Z$="SA" THEN 305
02AA 120 IF Z$="EN" THEN 400
02D5 125 IF Z$="NE" THEN 015
0300 130 IF Z$="L," THEN 500
032B 135 Z=VAL(BUF$) : IF Z>0 THEN 145
0355 140 PRINT "WHAT" : GOTO 100
036E 145 GOSUB 7000 : IF X<=64 THEN 150 : PRINT "LINE TOO LONG"
03A2 146 GOTO 100
03A5 148 REM
03A5 149 REM DELETE/REPLACE/ADD A PROGRAM LINE --------------------
03A5 150 GOSUB 4000 : IF K>V THEN 160
03BA 155 N(K)=0 : IF X>4 THEN 160 : GOTO 100
03E6 160 GOSUB 5000 : IF K<V+1 THEN 175
0407 170 PRINT "MEMORY FULL" : GOTO 100
0427 175 N(K)=Z
0440 180 B$(K)=LEFT$(BUF$,32) : B$(K+V)=MID$(BUF$,33,32)
04B3 185 GOTO 100
04B6 199 REM
04B6 200 REM LIST ALL LINES --------------------------------------
04B6 210 L=1
04BD 220 GOSUB 2000 : IF T=10000 THEN 100
04D0 230 GOSUB 3000 : GOTO 220
04D6 299 REM
04D6 300 REM SAVE ALL LINES ON TAPE ------------------------------
04D6 305 L=1 : H=9999 : GOTO 320
04E8 308 REM
04E8 309 REM   S,<BBBB>,<EEEE> ROUTINE --------------------------
04E8 310 GOSUB 6000
04EB 320 A1=0 : GOTO 420
04F4 399 REM
04F4 400 REM END/SAVE ROUTINE -----------------------------------
04F4 410 A1=1 : L=1 : H=9999
050A 420 GOSUB 496
050D 430 GOSUB 2000 : IF L>H+1 THEN 440 : IF T<10000 THEN 470
0540 440 IF A1=0 THEN 450 : BUF$=BUF$+CHR$($1A) : X=X+1
057B 450 IF X=0 THEN 100
058B 460 TWRITE BUF$ : GOTO 100
```

```
0597 470 S1=LEN(B$(S)+B$(S+V))
05D3 480 IF P1+S1+1>128 GOSUB 494
05ED 490 BUF$=BUF$+B$(S)+B$(S+V)+CHR$($D)
0636 492 P1=P1+S1+1 : X=X+1 : IF X>3 GOSUB 494 : GOTO 430
0667 494 TWRITE BUF$
0670 496 P1=0 : X=0 : BUF$="" : RETURN
068B 499 REM
068B 500 REM  L,<BBBB>,<EEEE> ROUTINE -----------------------------------
068B 510 GOSUB 6000
068E 530 GOSUB 2000 : IF T=10000 THEN 100 : IF L>H+1 THEN 100
06C4 540 GOSUB 3000 : GOTO 530
06CA 598 REM
06CA 599 REM  ROUTINE LOADS A PROGRAM TAPE ----------------------------
06CA 600 GOSUB 6500
06CD 630 IF X<4 THEN 100
06DA 640 TREAD BUF$
06E0 650 X=SUBSTR(CHR$($D),BUF$)
0705 655 IF X<>0 THEN 670
0715 660 X=SUBSTR(CHR$($1A),BUF$)
073A 665 IF X=0 THEN 600 : PRINT "*EOF*" : GOTO 100
0764 670 GOSUB 5000 : B$(K+V)="" : N(K)=VAL(BUF$)
07B3 675 IF X<=33 THEN 695
07C5 690 B$(K+V)=MID$(BUF$,33,X-33)
0808 695 B$(K)=LEFT$(BUF$,X-1)
0842 700 X1=LEN(BUF$)-X
085E 710 BUF$=MID$(BUF$,X+1,X1) : GOTO 650
0890 798 REM
0890 799 REM  FIND/<STRING>  ------------------------------------------
0890 800 Z$=MID$(BUF$,3,32)
08B9 810 FOR S=1 TO V : IF N(S)=0 THEN 850
08E1 830 IF SUBSTR(Z$,B$(S)+B$(S+V))<>0 THEN 870
0932 850 NEXT S
094C 860 PRINT Z$+"   NOT FOUND" : GOTO 100
0981 870 GOSUB 3000 : GOTO 100
0987 898 REM
0987 899 REM MEMORY AVAILABLE ROUTINE ---------------------------------
0987 900 GOSUB 6500
098A 910 PRINT X ; "  LINES AVAILABLE"
09B4 920 GOTO 100
09B7 998 REM
09B7 999 REM
09B7 1000 REM SUBROUTINES FOR DECODED COMMANDS ----------------------
09B7 1001 REM
09B7 1998 REM
09B7 1999 REM GET NEXT LINE ----------------------------------------
09B7 2000 J=1 : T=10000
09C6 2020 Q=N(J)
09D7 2050 IF Q=0 THEN 2300
09E7 2100 IF L>Q THEN 2300
09F9 2150 IF Q=L THEN 2400
0A09 2200 IF Q>T THEN 2300
0A1B 2250 T=Q : S=J
0A2B 2300 J=J+1 : IF J<>V+1 THEN 2020
0A58 2350 IF T<>10000 THEN 2450 : RETURN
0A69 2400 T=L : S=J
0A79 2450 L=T+1 : RETURN
0A86 2998 REM
```

```
0A86 2999 REM PRINT LINE -----------------------------------------------
0A86 3000 PRINT B$(S)+B$(S+V) : RETURN
0AC8 3998 REM
0AC8 3999 REM IS LINE # ALREADY IN LINE TABLE ? ---------------------
0AC8 4000 FOR K=1 TO V : IF Z=N(K) THEN 4010 : NEXT K
0B17 4010 RETURN
0B18 4998 REM
0B18 4999 REM IS THERE ROOM IN LINE # TABLE ---------------------------
0B18 5000 FOR K=1 TO V : IF N(K)=0 THEN 5010 : NEXT K
0B5A 5010 RETURN
0B5B 5998 REM
0B5B 5999 REM PICKS LOW + HIGH LINE NUMBER FOR L,/S, -----------------
0B5B 6000 C1=SUBSTR(",",MID$(BUF$,3,5))
0B94 6010 L=VAL(MID$(BUF$,3,4))
0BBC 6020 H=VAL(MID$(BUF$,C1+3,4)) : RETURN
0BEA 6498 REM
0BEA 6499 REM HOW MUCH ROOM IN THE LINE # TABLE ?  -------------------
0BEA 6500 X=0 : FOR K=1 TO V : IF N(K)=0 GOSUB 6505 : NEXT K :RETURN
0C33 6505 X=X+1 : RETURN
0C40 6998 REM
0C40 6999 REM LENGTH OF BUF$ ------------------------------------------
0C40 7000 X=LEN(BUF$) : RETURN
0C59 9998 BASE=$1000
0C59 9999 END

ERRORS  000
VAR LEN $0EA8
PGM LEN $0EDC
PGM END $0FDB

ZZ    1300 00 00
N     1302 32 00
B$    1366 64 00
V     0030 00 00
K     0032 00 00
*     0034 00 00
Z$    0036 00 00
Z     0056 00 00
X     0058 00 00
L     005A 00 00
T     005C 00 00
H     005E 00 00
A1    0060 00 00
S1    0062 00 00
S     0064 00 00
P1    0066 00 00
X1    0068 00 00
*     006A 00 00
J     006C 00 00
Q     006E 00 00
C1    0070 00 00
IO    1000 00 00
ST    1081 00 00
```

A/BASIC V1.0C ERROR CODES

01	Line number missing
02	Line number duplicated or out of sequence
03	Unrecognized statement
04	Syntax error - specific error not recognizable
05	Variable name missing or in error
06	Equal sign missing
07	Undefined line reference - no such line number
08	Right parenthesis missing/misnested parentheses
09	Operand missing in expression
10	Destination line number missing or in error
11	Number missing
12	Misnested for/next loop(s)
13	Symbol Table overflow - too many variable names*
14	Illegal task number
15	Missing, incorrect or wrong type relational operator
16	Delimiter (, or ;) missing
17	Quote missing at end of string
18	Illegal type or missing variable for FOR/NEXT counter
19	Redefined array name
20	Error in array specifcation
21	Size specification zero or greater than 255 in DIM
22	Variable storage overflow - tried to allocate past $FFFF
23	Reference to undeclared array
24	Subscript error in array reference
25	Wrong type or error in function argument
26	Illegal option
27	Unrecognized operator in string expression
28	Unrecognized operator in string expression (+ missing)
29	Arguments in string function in error or wrong type
30	Too many for/next loops active (max is 16)
31	Line reference table overflow*
32	Memory overflow - tried to allocate past $FFFF
33	ON or OFF missing in IRQ statement
34	GOTO or GOSUB missing
35 - up	syntax error, nature not identified

* These table overflow errors are not program errors. If more memory is available, either table may be expanded to include a larger number of entries.

SUMMARY OF A/BASIC LANGUAGE

ASSIGNMENT STATEMENTS:

LET	POKE

CONTROL STATEMENTS:

CALL	IF .. THEN	ON NOVER GOTO
FOR .. TO .. STEP	IF .. GOSUB	ON .. GOTO
NEXT	ON ERROR GOTO	ON .. GOSUB
GOSUB	ON OVR GOTO	STOP
GOTO	RETURN	

INPUT/OUTPUT:

INPUT	TREAD	TWRITE
PRINT		

SYSTEM CONTROL AND REAL-TIME:

GEN	ON NMI GOTO	SWITCH
IRQ ON	RETI	TASK .. OFF
IRQ OFF	STACK	TASK .. ON
ON IRQ GOTO		

COMPILER DIRECTIVES:

BASE	END	ORG
DIM	OPT	REM

NUMERIC FUNCTIONS:

ABS	POS	CLK	RND	PEEK	TAB

STRING FUNCTIONS:

ASC	LEN	SUBSTR	VAL
CHR$	LEFT$	MID$	RIGHT$
STR$	TRM$	BUF$	

ARITHMETIC/LOGICAL OPERATORS

+	ADD
-	SUBTRACT
/	DIVIDE
*	MULTIPLY
&	AND
!	OR
%	EXCLUSIVE OR
#	(UNARY) NOT
-	(UNARY) NEGATIVE
+	STRING CONCATENATE

RELATIONAL OPERATORS

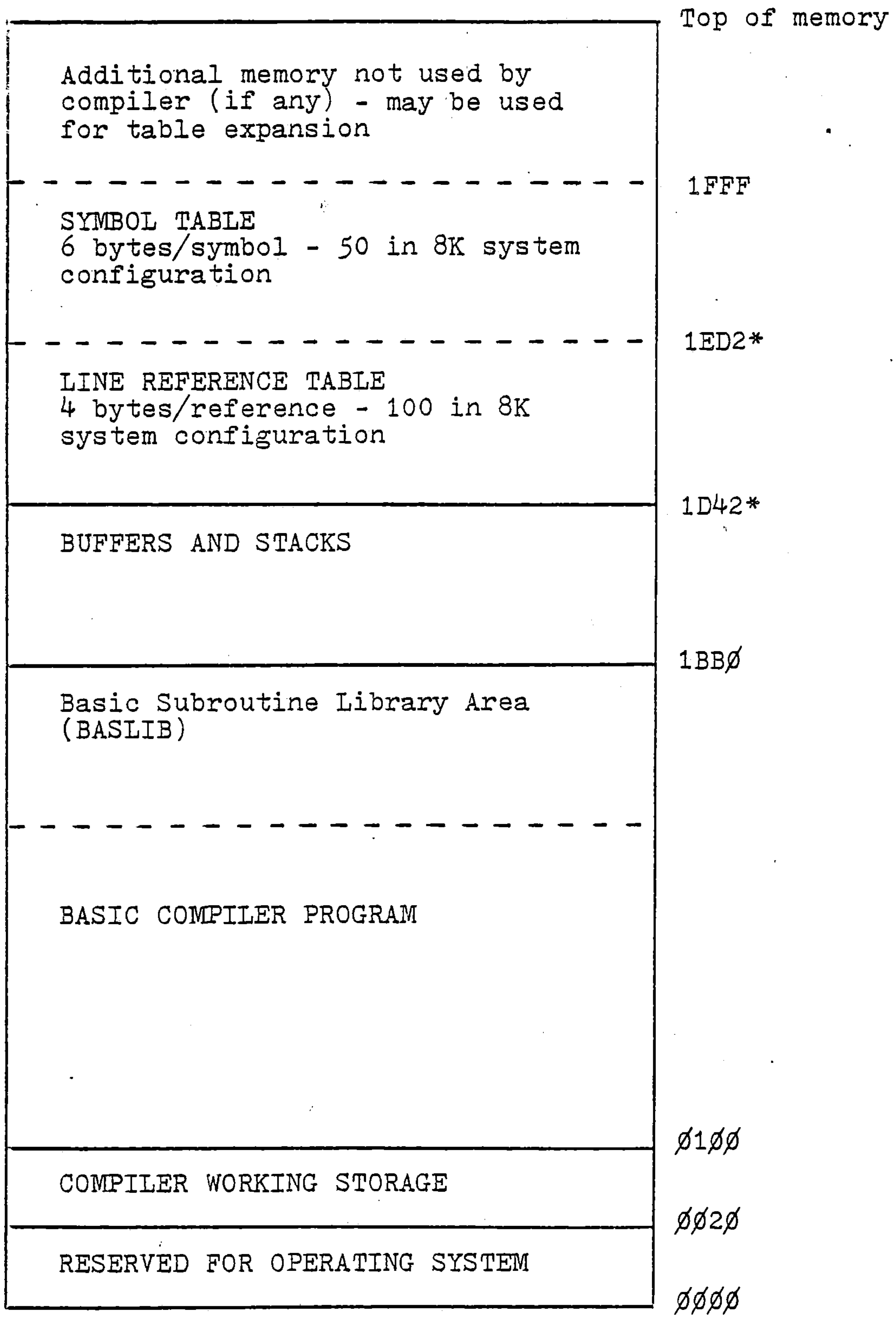
<	LESS THAN
>	GREATER THAN
=	EQUAL
<>	NOT EQUAL
<=	LESS OR EQUAL
=>	GREATER OR EQUAL
<>	STRING NOT EQUAL
=	STRING EQUAL

SPECIAL SYMBOLS:

( )	PARENTHESES
$	HEXADECIMAL PREFIX
:	STATEMENT SEPERATOR
"	STRING DELIMITER

LINE NUMBERS: 1 TO 9999

A/BASIC V 1.0C SYSTEM MEMORY MAP
<u>COMPILE TIME</u>



Top of memory
Additional memory not used by compiler (if any) - may be used for table expansion
1FFF
SYMBOL TABLE
6 bytes/symbol - 50 in 8K system configuration
1ED2*
LINE REFERENCE TABLE
4 bytes/reference - 100 in 8K system configuration
1D42*
BUFFERS AND STACKS
1BBØ
Basic Subroutine Library Area (BASLIB)
BASIC COMPILER PROGRAM
Ø1ØØ
COMPILER WORKING STORAGE
ØØ2Ø
RESERVED FOR OPERATING SYSTEM
ØØØØ

* These limit addresses are changable by user to expand tables above 8K if desired.

Application Note B-2

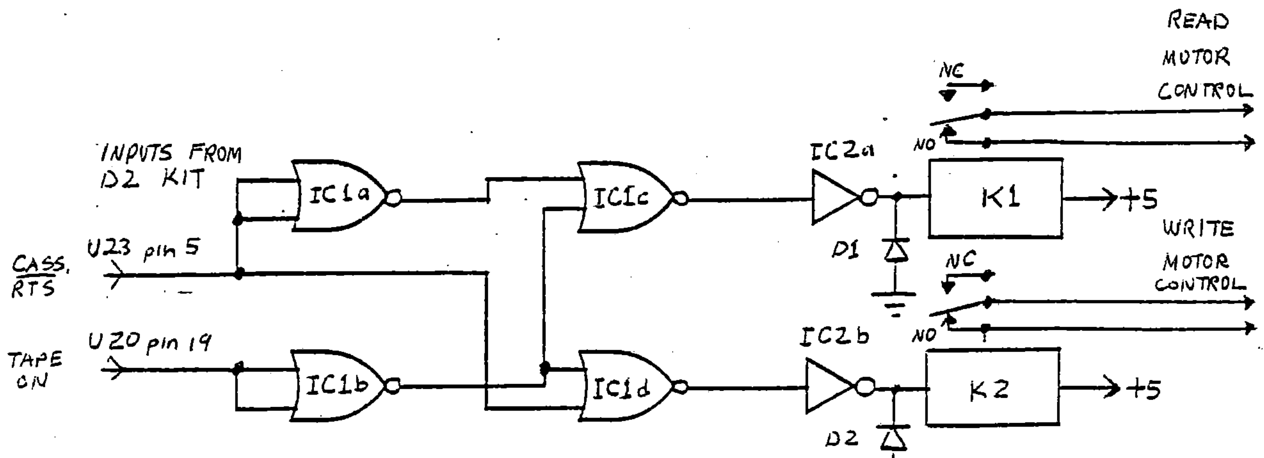
IMPLEMENTATION OF A/BASIC V1.0C ON THE MOTOROLA EVALUATION KIT 2 WITH THE DA-1 UPGRADE KIT

A/BASIC V1.0C has hooks built in to support the MEK6800D2 evaluation kit equipped with the MICROWARE DA-1 upgrade kit.

To support A/BASIC, the D2 kit must be equipped with:

1 - At least 8K bytes of RAM memory starting at address Ø.
2 - two cassette tape units and appropriate motor control for each unit.

The compiler must be able to control two tape units, one for reading the source program and one for writing the object tape. Automatic motor control is necessary because the compiler must alternately activate the READ and WRITE cassette units. The schematic below can be used to decode READ ON and WRITE ON from control signals available on the D2 MPU card. These motor control outputs can drive the motor control relay circuit also illustrated. The relay's contacts are connected to a cable and suitable plug (typically a micro phono plug) which is plugged into the cassette recorder's REMOTE jack.



IC1 - 4001B CMOS QUAD NOR
IC2 - 7405 TTL HEX O.C. INVERTER
D1,2 - 1N914 DIODE
K1,2 - RELAY, RADIO SHACK 275-004 or equiv.

2.99/PK. 10.

USING A/BASIC WITH THE MOTOROLA EVALUATION KIT 2 ("D2 KIT")
SOFTWARE MODIFICATIONS:

A/BASIC AND RT/EDIT ARE DESIGNED TO BE USED WITH THE D2 KIT BUT PATCHES MUST BE MADE TO TELL THE COMPILER AND EDITOR WHERE THE D2'S AUDIO CASSETTE INTERFACE IS LOCATED IN MEMORY, AS WELL AS SUPPLYING THE CORRECT CONTROL REGISTER VALUES FOR THE AUDIO CASSETTE INTERFACE ACIA.

TO MAKE THE PATCHES, LOAD THE COMPILER OR EDITOR AND USE THE RT/68 MEMORY CHANGE FUNCTION TO MODIFY THE LOCATIONS LISTED FOR EACH. THEN USE THE PUNCH FUNCTION TO CREATE A NEW TAPE WITH THE MODIFIED VERSION FOR SUBSEQUENT USE.

PATCHES FOR RT/EDIT:

```
0DAC - 51     0CEB - 10
0DC4 - 10     0D0A - 10
0DCA - B7     0D0D - B7
0DCB - 80     0D0E - 80
0DCC - 08     0D0F - 08
```

SAVE RT/EDIT BY USING THE RT/68 COMMAND: P,0100,0FDB

PATCHES FOR A/BASIC:

```
16F4 - 51     17A6 - 51     18DF - 10
1705 - 10     17BE - 10     18FE - 10
1708 - F7     17C4 - B7     1901 - B7
1709 - 80     17C5 - 80     1902 - 80
170A - 08     17C6 - 08     1903 - 08
170B - B7
170C - 80
170D - 08
```

SAVE A/BASIC BY USING THE RT/68 COMMAND: P,0100,1BBF

```
                          NAM    BASLIB.1
                          OPT    O
                   * BASLIB V1.1 BASIC SUBROUTINE LIBRARY FOR
                   * A/BASIC V1.0C - CASSETTE ORIENTED
                   * 11/2/77 K. KAPLAN


                   * RUN TIME STORAGE
0020                      ORG    $20
0020 0002          XSAVE  RMB    2
0022 0002          BUFPTR RMB    2
0024 0002          BUFBEG RMB    2
0026 0001          ZONE   RMB    1
0027 0002          STRPTR RMB    2
0029 0002          STRSAV RMB    2


                   * RT/68 REFERENCES
     E141          CRLF   EQU    $E141
     E0CC          OUTSP  EQU    $E0CC
     E3A6          OUTCH  EQU    $E3A6

     E350          INCH   EQU    $E350
     E07E          PDATA  EQU    $E07E

     8007          PIACB  EQU    $8007


1760                      ORG    $1760
                   *PUT SPACE IN I/O BUFFER
1760 86 20         ZOUTS  LDA A  #$20

                   *PUT CHAR IN I/O BUFFER
1762 DF 20         ZOUT   STX    XSAVE
1764 DE 22                LDX    BUFPTR
1766 A7 00                STA A  0,X
1768 96 26                LDA A  ZONE
176A 81 80                CMP A  #128      BUFFER LIMIT
176C 22 02                BHI    ZOUT2
176E 4C                   INC A
176F 08                   INX
1770 6F 00         ZOUT2  CLR    0,X       MARK END OF BUFFER
1772 97 26                STA A  ZONE
1774 DF 22                STX    BUFPTR
1776 DE 20                LDX    XSAVE
1778 39                   RTS


                   *INITIALIZE I/O
1779 DF 24         RESET  STX    BUFBEG
177B 20 04                BRA    ZINZ2
177D 86 01         ZINZIO LDA A  #1
177F 97 26                STA A  ZONE
1781 DE 24         ZINZ2  LDX    BUFBEG
```

```
1783 DF 22               STX     BUFFTR
1785 39                  RTS

                 *SKIP ZONE
1786 8D D8       ZSKPZ2  BSR     ZOUTS
1788 96 26       ZSKPZ   LDA A   ZONE
178A 84 07               AND A   #7
178C 81 01               CMP A   #1
178E 26 F6               BNE     ZSKPZ2
1790 39                  RTS

                 * PRINT BUFFER - NO CR/LF
1791 8D EE       ZENDBN  BSR     ZINZ2
1793 A6 00       ZEBN2   LDA A   0,X
1795 27 0B               BEQ     ZTAB3
1797 8D 2F               BSR     PRCHR
1799 08                  INX
179A 20 F7               BRA     ZEBN2

179C 8D C2       ZTAB2   BSR     ZOUTS
179E D1 26       ZTAB    CMP B   ZONE
17A0 22 FA               BHI     ZTAB2
                 * TAB FUNCTION
17A2 39          ZTAB3   RTS

17A3 20 BD       ZGOUT   BRA     ZOUT

                 * WRITE BUFFER TO TAPE
17A5 86 12       ZENDBT  LDA A   #$12        TAPE ON CODE
17A7 C6 3C               LDA B   #$3C
17A9 8D 16               BSR     TAPCON
17AB C6 5A               LDA B   #90
17AD 4F          ZENDT2  CLR A
17AE 8D 18               BSR     PRCHR
17B0 5A                  DEC B
17B1 26 FA               BNE     ZENDT2
17B3 86 02               LDA A   #2
17B5 8D 11               BSR     PRCHR
17B7 8D D8               BSR     ZENDBN
17B9 86 03               LDA A   #3
17BB 8D 0B               BSR     PRCHR
17BD 86 14               LDA A   #$14
17BF C6 34               LDA B   #$34
17C1 F7 8007     TAPCON  STA B   PIACB
17C4 8D 02               BSR     PRCHR
17C6 01                  NOP                 PATCH AREA FOR D2 KIT
17C7 39                  RTS
17C8 7E E3A6     PRCHR   JMP     OUTCH
```

```
PAGE  003   BASLIB.1


                        * PRINT BUFFER + CR/LF
17CB 8D C4    ZENDB   BSR    ZENDBN
17CD 8D AE    ZCRLF   BSR    ZINZIO
17CF 7E E141          JMP    CRLF
```

```
PAGE  009   BASLIB.1


                        * READ TAPE DATA RECORD
18DE 86 11    ZRDREC LDA A  #$11
18E0 C6 3C           LDA B  #$3C     CONTROL CODES
18E2 8D 1D           BSR    XCNTRL   TURN ON LOAD
18E4 BD E350  ZREC2  JSR    INCH     SEARCH FOR SOR
18E7 81 02           CMP A  #2       START OF RECORD CHAR?
18E9 26 F9           BNE    ZREC2
18EB 5F              CLR B           SET UP CHAR. COUNT
18EC BD E350  ZREC3  JSR    INCH
18EF 81 03           CMP A  #3
18F1 27 08           BEQ    ZREC4    EOR DETECTED
18F3 A7 00           STA A  0,X
18F5 08              INX
18F6 5C              INC B           BUMP CHAR COUNT
18F7 C1 80           CMP B  #128     BUFFER FULL?
18F9 25 F1           BCS    ZREC3    GET ANOTHER IF NOT
18FB 6F 00    ZREC4  CLR    0,X      MARK END OF BUFFER
18FD 86 13           LDA A  #$13     READ OFF CODE
18FF C6 34           LDA B  #$34     PIA CNTRL BYTE
1901 BD E3A6  XCNTRL JSR    OUTCH    XMIT CONTROL CODE
1904 F7 8007         STA B  PIACB
1907 39              RTS
```

```
                    * READ TERMINAL INPUT LINE
1908 86 3F          ZRDLIN LDA A  #'?
190A BD E3A6               JSR    OUTCH
190D BD E0CC               JSR    OUTSP
                    * READ INPUT BUFFER
1910 DF 22          ZRDBUF STX    BUFPTR
1912 DF 24                 STX    BUFBEG
1914 C6 80                 LDA B  #128
1916 BD E350        ZRBUF2 JSR    INCH
1919 81 0F                 CMP A  #$F        BACKSPACE?
191B 26 0C                 BNE    CHKDEL
191D 9C 24                 CPX    BUFBEG
191F 27 0C                 BEQ    LINKIL
1921 09                    DEX
1922 A6 00                 LDA A  0,X
1924 BD E3A6               JSR    OUTCH
1927 20 ED                 BRA    ZRBUF2
1929 81 18          CHKDEL CMP A  #$18
192B 26 17                 BNE    ZRBUF4
                    * DELETE LINE
192D 8D 07          LINKIL BSR    ZRBUF3
192F 20                    FCC    / *DEL*/
1930 2A
1931 44
1932 45
1933 4C
1934 2A
1935 04                    FCB    4
1936 30             ZRBUF3 TSX
1937 EE 00                 LDX    0,X
1939 31                    INS
193A 31                    INS
193B BD E07E               JSR    PDATA
193E 8D 0C                 BSR    ZRLIN3
1940 DE 24                 LDX    BUFBEG
1942 20 CC                 BRA    ZRDBUF
1944 A7 00          ZRBUF4 STA A  0,X
1946 81 0D                 CMP A  #$D
1948 26 09                 BNE    ZRBUF5
194A 6F 00                 CLR    0,X
194C 86 01          ZRLIN3 LDA A  #1
194E 97 26                 STA A  ZONE
1950 7E E141               JMP    CRLF
1953 5D             ZRBUF5 TST B
1954 27 C0                 BEQ    ZRBUF2
1956 08                    INX
1957 5A                    DEC B
1958 20 BC                 BRA    ZRBUF2
```

```
                     * A/BASIC V1.0C CASSETTE-ORIENTED INPUT/OUTPUT
                     * SUBROUTINE PACKAGE

                     * JMP TABLE TO RT/68 ROUTINES
16DE 7E E0EA TAPOUT JMP    $E0EA
16E1 7E E3A6 PRNTCH JMP    $E3A6     CONSOLE OUTPUT CHAR
16E4 7E E350 INCH   JMP    $E350     CONSOLE INPUT CHAR
        E16A CLNUP  EQU    $E16A     RT/68 ENTRY POINT
16E7 7E E0C8 OUT4HS JMP    $E0C8     PRINT 4 HEX + SP
             *
16EA 7E E0CA OUT2HS JMP    $E0CA     PRINT 2 HEX + SP
16ED 7E E141 OUTEOL JMP    $E141     PRINT CR+LF
16F0 7E E07E RTPDAT JMP    $E07E     PRINT CHAR. STRING

                     * TAPE OUTPUT DEVICE ON
16F3 86 12   OBJFWD LDA A  #$12      TAPE ON CHAR
16F5 C6 3C          LDA B  #$3C      PIA CONTRL BYTE
16F7 8D 0F          BSR    TAPCON
16F9 D6 42          LDA B  NULCNT
16FB 27 06          BEQ    DELAY2
16FD 4F      DELAY  CLR A
16FE 8D E1          BSR    PRNTCH
1700 5A             DEC B
1701 26 FA          BNE    DELAY
1703 39      DELAY2 RTS

                     * TAPE OUTPUT DEVICE OFF
1704 86 14   OBJSTP LDA A  #$14      TAPE OFF CHAR
1706 C6 34          LDA B  #$34      PIA CNTRL BYTE

1708 8D D7   TAPCON BSR    PRNTCH
170A F7 8007        STA B  $8007
170D 01             NOP
170E 39             RTS

                     * SUBR TO REWIND SOURCE FILE
170F 8D DC   SRCREW BSR    OUTEOL
1711 CE 175A        LDX    #REWMSG
1714 8D DA          BSR    RTPDAT    PRINT MESSAGE
1716 20 CC          BRA    INCH      WAIT FOR CHR

                     * SUBR TO GET SOURCE LINE
1718 C6 48   NEWBUF LDA B  #72
171A 8D 27          BSR    NEWBF9
171C 37      NEWBF2 PSH B            SET MAX CHAR COUNT
171D DE F3   NEWBF3 LDX    SRCPTR    GET TAPE BUF PTR
171F A6 00          LDA A  0,X       GET NEXT CHR
1721 26 04          BNE    NEWBF4    BRA IF NOT END OF RECORD
1723 8D 25          BSR    SRCREC    READ A NEW TAPE RECORD
1725 20 F6          BRA    NEWBF3
```

```
03358 1727 08         NEWBF4 INX                  ADV TAPE BUF PTR
03359 1728 DF F3             STX     SRCPTR       SAVE IT
03360 172A 33                PUL B
03361 172B 81 1A             CMP A   #$1A
03362 172D 26 02             BNE     NEWBF5
03363 172F 0C                CLC
03364 1730 39                RTS
03365 1731 DE 31      NEWBF5 LDX     BUFPTR       GET LINE BUF PTR
03366 1733 A7 00             STA A   0,X          DROP IN THE CHR
03367 1735 08                INX
03368 1736 DF 31             STX     BUFPTR
03369 1738 81 0D             CMP A   #$D
03370 173A 27 07             BEQ     NEWBF9
03371 173C 5A                DEC B
03372 173D 26 DD             BNE     NEWBF2
03373 173F 86 0D             LDA A   #$D
03374 1741 A7 00             STA A   0,X
03375 1743 CE 1CF9    NEWBF9 LDX     #LINBUF
03376 1746 DF 31             STX     BUFPTR
03377 1748 0D                SEC
03378 1749 39                RTS

03380                 * SUBR TO READ A TAPE RECORD, USES SUBR
03381                 * 'ZRDBUF' FROM BASLIB
03382 174A 8D 03      SRCREC BSR     SREC2        INIT PTRS
03383 174C BD 18DE           JSR     $18DE        CALL SUBR
03384 174F CE 1C78    SREC2  LDX     #TAPBUF
03385 1752 DF F3             STX     SRCPTR
03386 1754 39                RTS

03388                 * SUBR TO INIT I/O
03389 1755 8D F8      INZIO  BSR     SREC2
03390 1757 6F 00             CLR     0,X
03391 1759 39                RTS
03392 175A 50         REWMSG FCC     /PASS2/
03393 175F 04                FCB     $4
```

A/BA.   V1.0C Tape Dump   Page 1

```
S1 13 0100 20 0E BD 17 0F 96 3D 8A 80 97 2E BD 03 6F 20 38 B1
S1 13 0110 8E 1C 77 CE 1E D2 DF 4F DF 3E CE 1F FE DF 40 CE D9
S1 13 0120 1D 42 DF 2F 6F 00 08 9C 40 26 F9 5F D7 39 D7 38 6E
S1 13 0130 D7 3D D7 69 D7 6A D7 2E CE 00 30 DF 22 CE 00 6B E9
S1 13 0140 6F 00 08 8C 00 89 26 F8 CE 10 00 DF 24 BD 17 55 F7
S1 13 0150 86 5A 97 42 CE 00 00 DF 90 DF 5D DF 3A DF 67 BD 4D
S1 13 0160 06 21 CE 00 EA DF 59 BD 17 18 25 03 7E 07 BC D6 49
S1 13 0170 2E 27 0C 56 25 09 CE 00 24 BD 16 E7 BD 03 58 BD 15
S1 13 0180 03 12 81 0D 27 E1 BD 05 72 25 04 86 01 20 59 91 D2
S1 13 0190 3A 25 06 26 08 D1 3B 22 04 86 02 20 4B 97 3A D7 FB
S1 13 01A0 3B BD 03 D7 DF 31 BD 0E F9 BD 03 0D BD 03 1A 25 D9
S1 13 01B0 02 20 33 A6 01 BD 03 1A 25 05 BD 02 44 20 15 BD 46
S1 13 01C0 03 EF 24 22 5F CE 06 35 11 27 05 08 08 5C 20 F8 CA
S1 13 01D0 EE 00 AD 00 24 12 BD 03 12 81 0D 27 0D 08 81 3A F3
S1 13 01E0 27 C2 86 04 20 02 86 03 8D 07 7E 01 67 31 31 20 F1
S1 13 01F0 F9 36 BD 03 58 9C 31 27 0B 8C 1D 41 27 0A 08 BD D5
S1 13 0200 03 A3 20 F1 86 5E 8D 2D 8D 0C CE 02 38 BD 03 42 F2
S1 13 0210 32 7C 00 5D 8D 03 7E 03 62 16 4F C1 63 23 05 C0 EB
S1 13 0220 64 4C 20 F7 8D 0D 4F C1 09 23 05 C0 0A 4C 20 F7 FB
S1 13 0230 8D 01 17 8A 30 7E 03 4E 2A 2A 20 45 52 52 4F 52 8E
S1 13 0240 20 0D DE 31 8D 65 24 16 BD 0E F9 BD 0B 10 24 0E 74
S1 13 0250 96 28 27 05 BD 0D 95 20 03 BD 0F 23 25 01 39 96 4A
S1 13 0260 28 26 16 96 26 27 07 8D 39 BD 13 F7 20 07 96 55 9D
S1 13 0270 D6 56 BD 14 F3 DF 31 0D 39 4F C6 27 BD 15 52 8D 47
S1 13 0280 07 96 37 BD 0E B6 20 ED 96 26 27 02 20 14 96 37 22
S1 13 0290 27 06 96 87 D6 88 20 04 96 55 D6 56 BD 13 D4 7E 55
S1 13 02A0 13 E7 BD 14 0A BD 13 CF 7E 13 D9 5F D7 37 D7 26 02
S1 13 02B0 D7 28 8D 5E 8D 64 25 04 86 05 0C 39 E6 01 C1 24 9A
S1 13 02C0 26 06 08 08 D7 28 20 03 BD 10 DB 8D 39 27 13 7D A7
S1 13 02D0 00 28 27 05 BD 13 0D 20 03 BD 13 12 97 55 D7 56 CB
S1 13 02E0 0D 39 7C 00 26 7D 00 28 26 05 BD 11 1F 20 03 BD 85
S1 13 02F0 11 1A 24 C6 7E 14 0E DE 87 DF 55 86 11 97 28 97 BF
S1 13 0300 37 8D 0A 7E 02 48 37 E6 00 C1 28 33 39 DE 31 20 B2
S1 13 0310 01 08 A6 00 81 20 27 F9 0D 39 81 41 25 1B 81 5B 45
S1 13 0320 22 17 0D 39 81 30 25 11 81 39 22 0D 0D 39 BD 03 74
S1 13 0330 12 81 0D 27 F7 81 3A 27 F3 0C 39 81 2C 26 FA 08 0C
S1 13 0340 0D 39 A6 00 81 0D 26 01 39 08 8D 02 20 F4 7E 16 90
S1 13 0350 E1 DF 31 8D 0D DE 31 39 8D 04 8D E6 8D 04 CE 1C 47
S1 13 0360 F9 39 BD 16 ED 96 39 4C 97 39 81 39 24 01 39 C6 CE
S1 13 0370 04 8D 29 D7 39 C6 04 86 2D 8D D3 5A 26 F9 BD 16 86
S1 13 0380 ED BD 16 ED CE 03 AD 8D B9 96 38 4C 97 38 BD 02 50
S1 13 0390 19 C6 0A 8D 10 CE 03 B1 8D A8 C6 03 BD 16 ED 5A 39
S1 13 03A0 26 FA 39 C6 01 86 20 8D A5 5A 26 F9 39 50 47 20 E8
S1 13 03B0 0D 41 2F 42 41 53 49 43 20 31 2E 30 43 0D DF 31 4B
S1 13 03C0 CE 1D 3E 08 08 08 08 9C 2F 26 02 0C 39 A1 00 26 E1
S1 13 03D0 F2 E1 01 26 EE 0D 39 8D E5 24 11 9C 3E 26 05 86 B9
S1 13 03E0 1F 7E 10 57 96 24 D6 25 A7 02 E7 03 DE 31 39 DF 96
S1 13 03F0 31 CE 04 77 DF 49 CE 05 00 20 25 DF 31 CE 05 29 33
S1 13 0400 DF 49 CE 05 50 20 19 DF 31 CE 05 51 DF 49 CE 05 35
S1 13 0410 71 20 0D BD 03 0D DF 31 CE 04 EF DF 49 CE 05 28 79
S1 13 0420 DF 4B 86 07 97 3C DE 31 DF 5F DF 45 7A 00 3C 26 F1
S1 13 0430 06 DE 5F DF 31 0C 39 DE 31 A6 00 BD 03 1A 24 F1 7C
S1 13 0440 08 DF 31 DE 49 DF 47 DE 45 DF 20 DE 47 A6 00 2B 2B
S1 13 0450 1E 08 DF 47 DE 20 A1 00 26 05 08 DF 20 20 EC DE 91
S1 13 0460 47 A6 00 2B 03 08 20 F9 9C 4B 27 C0 08 20 D6 84 FC
S1 13 0470 7F DE 20 DF 31 0D 39 45 4E 44 80 46 4F 52 81 4E 98
S1 13 0480 45 58 54 82 52 45 4D 84 47 4F 54 4F 85 49 46 86 BA
```

Page 2   A/BASIC V1.0C

```
S1 13 0490 47 4E 50 55 54 87 50 52 49 4E 54 88 52 45 54 55 EC
S1 13 04A0 52 4E 89 4F 52 47 8A 42 41 53 45 8B 4C 45 54 8C 96
S1 13 04B0 53 54 41 43 4B 8E 53 54 4F 50 8F 43 41 4C 4C 90 B3
S1 13 04C0 44 49 4D 91 50 4F 4B 45 92 4F 50 54 93 47 45 4E 9C
S1 13 04D0 94 54 52 45 41 44 96 54 57 52 49 54 45 97 52 45 71
S1 13 04E0 54 49 98 54 41 53 4B 99 53 57 49 54 43 48 9A 4F 4C
S1 13 04F0 4E 8D 47 4F 53 55 42 83 42 55 46 24 95 49 52 51 98
S1 13 0500 9B 54 48 45 4E C0 54 4F C1 53 54 45 50 C2 4F 46 66
S1 13 0510 46 C3 45 52 52 4F 52 C4 4F 56 52 C4 4E 4D 49 C5 1C
S1 13 0520 4E 4F 56 52 C6 54 41 42 C7 41 42 53 83 52 4E 44 E1
S1 13 0530 82 50 4F 53 80 50 45 45 4B 84 43 4C 4B 81 4C 45 2E
S1 13 0540 4E 85 56 41 4C 86 41 53 43 87 53 55 42 53 54 52 2A
S1 13 0550 88 53 54 52 24 85 52 49 47 48 54 24 82 4C 45 46 72
S1 13 0560 54 24 81 4D 49 44 24 83 43 48 52 24 84 54 52 4D 95
S1 13 0570 24 80 BD 03 24 24 3A C6 05 D7 3C 5F D7 4D D7 4E 0B
S1 13 0580 BD 03 24 24 29 80 30 36 DF 31 CE 00 4D 8D 23 E6 8F
S1 13 0590 01 A6 00 8D 1D 8D 1B EB 01 A9 00 30 EB 00 89 00 25
S1 13 05A0 31 BD 13 D4 DE 31 08 A6 00 7A 00 3C 26 D2 96 4D 24
S1 13 05B0 0D 39 68 01 69 00 39 DF 5F 36 37 96 2E 27 2A 84 A2
S1 13 05C0 02 27 26 C6 02 BD 03 A5 CE 00 24 BD 16 E7 CE 00 31
S1 13 05D0 89 BD 16 EA 33 37 C1 01 27 0C C1 02 26 05 BD 16 B1
S1 13 05E0 EA 20 03 BD 16 E7 BD 03 62 33 CE 00 69 8D 05 32 D0
S1 13 05F0 DE 5F 0D 39 A6 00 08 DF 65 DE 24 08 DF 24 37 D6 68
S1 13 0600 2E 27 17 DE 57 A7 00 08 DF 57 96 90 4C 97 90 81 46
S1 13 0610 18 25 02 8D 0C DE 67 08 DF 67 DE 65 33 5A 26 D4 A1
S1 13 0620 39 96 90 27 03 BD 16 A5 CE 00 93 DF 57 DE 24 DF 4D
S1 13 0630 91 7F 00 90 39 07 BA 0B 93 0C 12 09 BF 01 ED 09 A1
S1 13 0640 C3 0A 75 09 0F 08 68 09 69 0B 32 0A EC 02 42 0C E7
S1 13 0650 9A 0A 2A 0A 18 0A 21 06 AC 0B 1A 06 6D 09 48 02 DE
S1 13 0660 F7 09 0B 08 64 09 6D 09 8B 09 74 07 A3 DE 31 C6 09
S1 13 0670 01 BD 03 2E 25 34 81 4E 27 17 5C 81 48 27 12 58 6B
S1 13 0680 81 53 27 0D 58 81 49 27 08 81 2C 27 08 86 1A 0C 85
S1 13 0690 39 DA 3D D7 3D 08 20 D7 08 A6 00 BD 0B 04 24 08 4D
S1 13 06A0 4D 26 03 5D 26 04 86 15 0C 39 0D 39 BD 03 0D 86 D0
S1 13 06B0 5A 16 BD 13 12 7F 00 5E A6 00 BD 03 1A 25 03 86 D9
S1 13 06C0 0A 39 E6 01 C1 24 27 08 7C 00 5E BD 10 DB 20 02 44
S1 13 06D0 08 08 BD 12 E7 24 09 7D 00 2E 26 0A 86 13 0C 39 6A
S1 13 06E0 A7 00 E7 01 DF 5F DE 31 A6 00 81 28 26 1F 7F 00 17
S1 13 06F0 8F 8D A5 24 1B D7 8E A6 00 81 2C 26 0C D6 5E 27 B1
S1 13 0700 0C 8D 95 24 0B D7 8F A6 00 81 29 27 04 86 14 0C 01
S1 13 0710 39 08 DF 31 96 2E 26 2E DE 5F 96 8E D6 8F A7 04 FB
S1 13 0720 E7 05 8D 3D A7 02 E7 03 BD 13 46 D6 8E 96 8F 27 B6
S1 13 0730 02 8D 5B 8D 27 7D 00 5E 26 08 8D 20 8D 1E 8D 1C 0D
S1 13 0740 8D 1A 8D 22 24 15 DE 31 BD 03 2E 25 0E BD 03 3B EB
S1 13 0750 24 06 BD 03 12 7E 06 B5 86 10 0C 39 58 49 25 28 97
S1 13 0760 39 96 22 D6 23 39 37 36 30 8D F6 EB 01 A9 00 25 88
S1 13 0770 17 97 22 D7 23 96 2E 26 0C 96 69 D6 6A EB 01 A9 E1
S1 13 0780 00 97 69 D7 6A 0D 20 03 86 20 0C 31 31 39 37 C6 AA
S1 13 0790 08 37 30 5F 58 49 24 04 EB 01 89 00 6A 00 26 F4 C5
S1 13 07A0 31 31 39 BD 04 13 24 0E 81 43 26 03 7E 15 A3 81 00
S1 13 07B0 0D 26 03 7E 15 9F 86 21 0C 39 31 31 BD 15 AB 96 6C
S1 13 07C0 2E 26 03 7E 01 02 BD 06 21 BD 08 41 CE 08 4C 8D B4
S1 13 07D0 64 96 5D BD 02 19 8D 69 CE 08 5C 8D 58 CE 00 69 A2
S1 13 07E0 8D 58 CE 08 54 8D 4E CE 00 67 8D 4E CE 08 44 8D 64
S1 13 07F0 44 DE 91 09 DF 5F CE 00 5F 8D 3F 8D 44 D6 2E C4 69
S1 13 0800 04 27 23 DE 3E 9C 4F 27 1D A6 00 8D 1C 08 A6 00 4E
S1 13 0810 8D 17 C6 03 BD 03 A5 08 BD 16 E7 BD 16 EA BD 16 B0
```

```
S1 13 0820 EA BD 03 51 20 DF 7E E1 6A 4D 2A 02 86 2A 26 02 B0
S1 13 0830 86 20 7E 03 4E BD 03 42 20 F6 86 24 8D F4 BD 16 29
S1 13 0840 E7 7E 03 62 50 47 4D 20 45 4E 44 0D 45 52 52 4F 8A
S1 13 0850 52 53 20 0D 50 47 4D 20 4C 45 4E 0D 56 41 52 20 C9
S1 13 0860 4C 45 4E 0D 86 01 20 01 4F 97 6F 4C 97 6B BD 08 8B
S1 13 0870 F6 C6 19 8D 7A BD 03 2E 26 04 C6 6D 20 71 BD 04 FB
S1 13 0880 13 24 08 81 47 27 26 81 15 27 5A BD 0E F9 DF 5F F7
S1 13 0890 BD 0D 95 24 0B 4D 27 04 C6 89 20 21 C6 85 20 1D 36
S1 13 08A0 DE 5F DF 31 BD 04 13 24 0D 81 47 26 09 BD 0F 23 0C
S1 13 08B0 24 32 C6 3E 20 07 BD 0F 23 24 29 C6 96 8D 30 BD A1
S1 13 08C0 03 3B 24 06 C6 28 8D 27 20 C1 81 3B 26 0E 08 BD 84
S1 13 08D0 03 2E 26 B7 96 6F 26 15 C6 31 20 13 BD 03 31 27 84
S1 13 08E0 04 86 10 0C 39 96 6F 26 04 C6 6B 20 02 C6 45 4F 49
S1 13 08F0 DB 76 99 75 20 0F DE 87 DF 4D BD 03 0D 7E 13 E7 90
S1 13 0900 4F DB 78 99 77 BD 14 57 7E 03 12 C6 91 20 02 C6 37
S1 13 0910 BB 37 BD E2 33 8D E9 C6 01 D7 6C BD 04 13 24 04 C3
S1 13 0920 81 15 27 22 BD 02 AB 24 1E 96 28 26 08 5F 8D D0 90
S1 13 0930 BD 02 5F 20 07 BD 02 88 C6 6B 8D C4 BD 03 0D 81 57
S1 13 0940 2C 26 03 08 20 DE 0D 39 DE 31 BD 03 12 BD 0F C3 92
S1 13 0950 24 16 4D 26 05 17 8D 19 20 03 BD 14 26 BD 03 12 38
S1 13 0960 BD 03 3B 24 02 20 E3 0D 39 86 39 20 02 86 3B DE 99
S1 13 0970 31 7E 14 14 86 A0 C6 0E BD 13 D4 86 7C DE 31 BD 30
S1 13 0980 13 E9 BD 15 A3 BD 15 95 7E 15 9F BD 0A F6 24 1F 59
S1 13 0990 4D 26 1C C1 10 24 18 86 86 BD 14 26 DF 31 BD 04 E3
S1 13 09A0 13 24 0C 81 43 26 04 C6 2C 20 0C 81 0D 27 06 86 B3
S1 13 09B0 0E 0C 39 C6 2C C6 35 86 E3 20 6C 86 CE 20 06 86 FE
S1 13 09C0 BD 20 02 86 7E 97 89 8D 0B 24 13 97 8A D7 8B C6 08
S1 13 09D0 03 7E 05 B7 BD 03 0D BD 05 72 25 03 86 0A 39 DF 05
S1 13 09E0 31 7D 00 2E 26 20 BD 03 BE 25 17 BC 00 3E 26 04 03
S1 13 09F0 86 0D 0C 39 A7 00 E7 01 6F 03 6F 02 08 08 08 08 89
S1 13 0A00 DF 2F DE 31 0D 39 BD 03 BE 24 09 A6 02 E6 03 26 1D
S1 13 0A10 F1 4D 26 EE 86 07 0C 39 DE 31 86 E1 C6 6A 7E 13 77
S1 13 0A20 C9 BD 0A F6 25 01 39 7E 14 57 86 8E 97 89 BD 0A F9
S1 13 0A30 FB 24 F3 20 96 96 2C 26 03 BD 0E A1 86 0C BD 0E 36
S1 13 0A40 B6 4F C6 27 BD 15 4B 4F C6 20 7E 15 52 8D E6 8D 79
S1 13 0A50 73 24 31 8D 70 86 24 C1 04 27 05 4C C1 03 26 24 D8
S1 13 0A60 97 5E BD 0D 95 24 BF 86 0B BD 0E B6 96 5E C6 03 7C
S1 13 0A70 BD 14 26 20 36 BD 03 0D BD 0D 95 25 D0 BD 0F 21 17
S1 13 0A80 8D 42 25 03 86 0F 39 8D 3C D7 5E 7C 00 51 7F 00 53
S1 13 0A90 2D BD 0F 32 C6 01 96 2D 27 04 BD 14 1B 5F BD 10 5A
S1 13 0AA0 BC CE 0A E0 D6 5E 5A 58 BD 13 66 DF 31 BD 04 13 CE
S1 13 0AB0 24 0E 81 40 26 03 7E 09 C3 81 03 26 03 7E 09 BF D9
S1 13 0AC0 86 04 0C 39 5F A6 00 81 3D 26 04 CA 04 20 0E 81 E9
S1 13 0AD0 3C 26 04 CA 02 20 06 81 3E 26 E7 CA 01 08 0D 39 D5
S1 13 0AE0 15 7F 15 64 15 85 15 6E 15 5D 15 68 8D 0D 24 D3 58
S1 13 0AF0 97 22 D7 23 0D 39 BD 03 0D 20 09 DE 31 8D 11 24 32
S1 13 0B00 0E BD 03 12 BD 05 72 25 06 8D 3F 25 02 86 0B 39 E5
S1 13 0B10 BD 03 12 81 3D 26 7A 08 0D 39 8D DA 24 F1 BD 02 18
S1 13 0B20 DC 7F 00 26 8D EA 24 E7 BD 0F 23 96 55 D6 56 7E 3A
S1 13 0B30 15 3C 8D C7 24 D9 97 24 D7 25 96 2E 27 07 DF 31 56
S1 13 0B40 BD 06 21 DE 31 0D 39 BD 03 0D 81 24 26 43 08 8D F8
S1 13 0B50 2A 24 3E C6 04 D7 3C 5F D7 4D D7 4E 8D 1D 24 15 9D
S1 13 0B60 C6 04 78 00 4E 79 00 4D 5A 26 F7 9A 4E 97 4E 08 DF
S1 13 0B70 7A 00 3C 26 E7 96 4D D6 4E 0D 39 A6 00 80 30 25 E6
S1 13 0B80 10 81 09 23 0A 80 07 81 0A 25 06 81 0F 22 02 0D 9C
S1 13 0B90 39 0C 39 BD 03 0D DF 31 BD 10 DB 25 03 86 05 39 62
```

```
S1 13 0BB0 24 ED BD 04 13 24 04 81 41 27 04 86 04 0C 39 96 D2
S1 13 0BC0 5B D6 5C 8A 80 BD 0C 8B 24 D5 7F 00 5E BD 04 13 8C
S1 13 0BD0 24 12 81 42 26 0E 96 5B D6 5C CA 80 BD 0C 8B 24 FF
S1 13 0BE0 BE 7C 00 5E DF 31 DE 59 8C 00 AA 26 04 86 1E 0C 12
S1 13 0BF0 39 96 5B 7D 00 5E 27 02 8A 80 A7 00 09 D6 5C E7 F0
S1 13 0C00 00 09 96 24 A7 00 09 96 25 A7 00 09 DF 59 DE 31 BB
S1 13 0C10 0D 39 BD 03 0D BD 10 DB 25 03 86 05 39 97 5B D7 60
S1 13 0C20 5C DF 31 DE 59 8C 00 EA 26 04 86 0C 0C 39 08 A6 F8
S1 13 0C30 00 97 56 08 A6 00 97 55 08 E6 00 08 A6 00 DF 59 55
S1 13 0C40 DE 31 97 5E 84 7F 91 5B 26 E0 D1 5C 26 DC BD 13 A8
S1 13 0C50 12 36 37 BD 15 02 96 5E 84 80 27 0E 96 5B D6 5C ED
S1 13 0C60 CA 80 BD 13 12 BD 14 FC 20 06 C6 01 4F BD 14 C9 B1
S1 13 0C70 33 32 BD 14 F3 96 5B D6 5C 8A 80 BD 13 12 BD 15 66
S1 13 0C80 08 BD 15 68 96 55 D6 56 7E 13 C9 BD 13 12 97 55 DF
S1 13 0C90 D7 56 5F D7 26 D7 28 7E 02 59 BD 04 13 24 03 7E 76
S1 13 0CA0 0D 04 BD 0F 21 24 23 86 4D 8D 56 86 27 C6 02 8D 43
S1 13 0CB0 4B 86 C6 C6 FF 8D 45 BD 03 0D BD 03 EF 5F 81 05 A1
S1 13 0CC0 27 09 5C 81 03 27 04 86 04 0C 39 D7 8F 7F 00 8E A3
S1 13 0CD0 86 C1 D6 8E 5C D7 8E 8D 23 BD 15 76 96 8F 26 04 5D
S1 13 0CE0 8D 5B 20 03 BD 09 BB 24 E1 BD 03 0D 81 2C 26 05 CA
S1 13 0CF0 08 DF 31 20 DB 96 8F 27 06 86 AD 5F 7E 14 26 86 BB
S1 13 0D00 0D 7E 14 14 97 8E BD 03 0D BD 03 EF 24 B9 81 05 28
S1 13 0D10 26 B5 96 8E 81 45 26 04 86 06 20 05 81 1B 26 0D 60
S1 13 0D20 4F 36 BD 09 BB 24 A3 86 A0 33 7E 15 52 C6 24 81 49
S1 13 0D30 44 27 05 5C 81 46 26 8F 17 C6 03 8D BF 7E 09 C3 F1
S1 13 0D40 DF 31 BD 03 12 81 22 26 45 08 DF 31 5F 8D 2C 25 5A
S1 13 0D50 FC 4D 27 05 86 11 7E 10 57 5C 86 20 BD 14 26 96 0F
S1 13 0D60 24 D6 25 37 36 DE 31 8D 12 24 05 BD 14 14 20 F7 20
S1 13 0D70 BD 14 14 32 33 BD 13 D4 7E 13 E7 A6 00 BD 03 31 72
S1 13 0D80 27 0C 81 22 26 0B A6 01 81 22 27 04 4F 08 0C 39 47
S1 13 0D90 08 08 5C 0D 39 8D 1D 24 1A 96 2C 27 0E 96 85 D6 CD
S1 13 0DA0 86 BD 13 D4 BD 13 E7 86 02 0D 39 96 29 81 01 27 28
S1 13 0DB0 01 4F 0D 39 A6 00 8D 88 25 63 BD 03 1A 24 F4 20 44
S1 13 0DC0 05 86 1C 7E 10 57 E6 01 C1 24 26 3A 08 08 BD 03 97
S1 13 0DD0 06 26 11 37 BD 0E 7D 33 BD 11 1A 24 E6 BD 11 09 57
S1 13 0DE0 C6 03 20 08 BD 13 0D BD 13 E7 C6 02 D7 29 D6 2C B0
S1 13 0DF0 27 03 BD 0E A1 BD 03 12 81 2B 26 B6 08 D6 2C 26 CF
S1 13 0E00 B3 BD 0E A1 20 AE DF 31 DF 5F BD 04 13 24 16 81 14
S1 13 0E10 15 27 04 DE 5F 20 0E 96 87 D6 88 8D 84 C6 01 20 B0
S1 13 0E20 CB 86 1D 20 9E BD 04 07 25 05 96 29 26 93 39 36 B9
S1 13 0E30 8D 54 BD 03 06 26 8A 08 32 36 81 03 22 17 BD 15 58
S1 13 0E40 99 86 01 BD 0E B6 BD 0D B4 24 D6 33 37 5D 27 20 77
S1 13 0E50 BD 03 3B 24 CC 8D 2A 33 37 C1 03 26 13 A6 00 BD 22
S1 13 0E60 03 3B 24 BD 86 37 BD 14 14 8D 16 86 32 BD 14 14 7D
S1 13 0E70 A6 00 BD 10 66 32 8B 04 8D 3C 7E 0D F5 5F 7E 0F 9F
S1 13 0E80 1A 8D FA 7E 11 B5 96 2C 26 16 D6 85 86 86 BD 14 43
S1 13 0E90 26 D6 86 86 C6 BD 14 26 4F C6 27 D7 2C 7E 14 F3 C5
S1 13 0EA0 39 96 29 26 04 8D E3 20 02 8D DB 96 29 26 01 39 03
S1 13 0EB0 4C 81 04 26 01 4A C6 01 D7 6D 5F DF 31 CE 0E D5 C1
S1 13 0EC0 11 27 05 5C 08 08 20 F8 E6 01 A6 00 DE 31 DB 7A 6C
S1 13 0ED0 99 79 7E 14 57 00 38 00 00 00 98 00 A1 00 1D 00 85
S1 13 0EE0 7D 00 4E 00 62 00 BD 01 2D 00 12 01 0E 00 C2 01 02
S1 13 0EF0 3F 00 9C 00 0E 00 C8 00 31 DF 31 CE 1C 07 DF 53 D9
S1 13 0F00 86 71 36 36 30 68 00 68 01 EE 00 31 31 A6 01 27 5B
S1 13 0F10 02 97 3E DE 31 5F D7 2C D7 29 D7 2D D7 51 D7 52 30
```

```
S1 13 0F40 20 F2 81 2D 27 02 0D 39 96 2D 36 4F 97 2D 8D 17 BE
S1 13 0F50 96 2D 33 D7 2D 4D 27 05 BD 14 1B 20 DF C6 01 BD AB
S1 13 0F60 10 BC 20 D0 7E 03 12 08 7C 00 51 8D 32 8D F5 81 97
S1 13 0F70 2A 26 09 8D 26 C6 02 BD 10 BC 20 F1 81 2F 27 01 27
S1 13 0F80 39 96 2D 36 7F 00 2D 8D 12 96 2D 27 04 C6 04 20 08
S1 13 0F90 02 C6 03 BD 10 BC 32 97 2D 20 D2 7C 00 51 08 8D AF
S1 13 0FA0 39 8D C1 81 26 26 04 86 06 20 0F 81 21 26 04 86 D8
S1 13 0FB0 07 20 07 81 25 27 01 39 86 08 36 8D 19 33 BD 10 8E
S1 13 0FC0 BC 20 DE DF 31 A6 00 BD 05 72 25 09 BD 0B 4A 25 14
S1 13 0FD0 04 DE 31 A6 00 39 7C 00 51 08 8D 88 BD 03 1A 24 33
S1 13 0FE0 25 A6 01 BD 03 1A 24 08 BD 03 FB 24 03 7E 11 BE FC
S1 13 0FF0 A6 00 BD 10 DB BD 03 06 26 06 BD 11 05 24 58 39 25
S1 13 1000 BD 13 12 4F 20 6B 8D BB 24 04 86 01 20 63 81 28 FD
S1 13 1010 27 4D 81 2D 26 14 08 8D AA 24 0A 96 4D D6 4E 40 BC
S1 13 1020 50 82 00 20 14 8D 1E 7E 14 1B 81 23 26 27 08 8D D8
S1 13 1030 92 24 0D 96 4D D6 4E 43 53 BD 13 D4 86 01 20 31 D0
S1 13 1040 8D 03 7E 13 DF 7C 00 51 BD 0F DA BD 0F 29 7C 00 B8
S1 13 1050 2D 7F 00 51 39 86 09 8E 1C 77 DF 31 7E 01 D4 08 3B
S1 13 1060 BD 12 D2 BD 0F 32 81 29 27 04 86 08 20 E9 08 0D 5C
S1 13 1070 39 D6 51 C1 01 23 0F D6 52 27 0B 7F 00 52 7F 00 6E
S1 13 1080 51 36 BD 14 0E 32 DF 31 DE 53 A7 00 09 D6 4E E7 C8
S1 13 1090 00 09 D6 4D E7 00 09 DF 53 DE 31 0D 39 DF 31 DE BB
S1 13 10A0 53 8C 1C 07 27 11 08 A6 00 97 4D 08 A6 00 97 4E DD
S1 13 10B0 08 E6 00 DF 53 20 02 C6 02 DE 31 39 37 96 52 26 95
S1 13 10C0 12 8D DA 37 36 96 4D 36 8D 30 32 97 4D 32 97 4E 33
S1 13 10D0 33 20 02 8D C8 32 BD 13 58 20 35 BD 03 1A 24 0B AA
S1 13 10E0 A6 01 BD 03 24 25 0B C1 24 26 02 0C 39 5F A6 00 EA
S1 13 10F0 20 05 A6 00 E6 01 08 08 0D 39 BD 10 9D 7C 00 52 AC
S1 13 1100 86 05 7E 13 58 8D 18 24 10 96 7D D6 7E BD 14 57 FF
S1 13 1110 4F 97 51 4C 97 52 97 2D 0D 39 37 C6 20 20 03 37 DE
S1 13 1120 C6 02 D7 35 33 BD 12 E7 24 0A A6 02 E6 03 37 36 D2
S1 13 1130 A6 04 26 04 86 17 0C 39 97 70 A6 05 26 22 D6 35 F0
S1 13 1140 37 DE 31 BD 12 D2 BD 10 45 33 D7 35 C1 02 26 05 75
S1 13 1150 BD 15 8D 20 02 8D 50 32 33 D0 35 82 00 7E 14 C9 E6
S1 13 1160 36 DE 31 08 8D 4F 32 97 36 A6 00 81 2C 27 06 31 A2
S1 13 1170 31 86 18 0C 39 08 D6 36 8D 2D 7C 00 51 BD 0F 32 BE
S1 13 1180 4F BD 10 BC BD 15 8D D6 36 4F CB 01 89 00 58 49 D3
S1 13 1190 40 50 82 00 DF 31 30 EB 01 A9 00 31 31 DE 31 BD 36
S1 13 11A0 14 C9 A6 00 7E 10 66 86 86 97 6E BD 14 26 96 7B AB
S1 13 11B0 D6 7C 7E 14 57 BD 12 D2 BD 0F 32 7E 10 4B 4D 26 05
S1 13 11C0 0E 4F C6 26 BD 13 D4 BD 12 D2 BD 15 43 20 2E 81 A9
S1 13 11D0 01 26 09 4F C6 05 BD 13 D4 7E 10 71 81 02 26 30 45
S1 13 11E0 97 72 BD 03 06 26 0A BD 10 45 8D 1B BD 13 F7 20 5B
S1 13 11F0 05 BD 12 D2 8D 11 96 7F D6 80 BD 14 57 4F 97 51 DD
S1 13 1200 4C 97 2D 97 52 0D 39 86 52 16 BD 13 12 7E 13 E7 53
S1 13 1210 81 03 26 17 BD 03 06 27 05 86 19 7E 10 57 BD 10 C6
S1 13 1220 45 86 2A C6 04 BD 14 26 7E 14 1B 81 04 26 18 BD D7
S1 13 1230 03 06 26 E5 08 BD 03 12 BD 0F C3 24 DC 36 A6 00 51
S1 13 1240 BD 10 66 32 7E 11 C4 36 BD 12 D2 D6 2C 37 27 08 A3
S1 13 1250 BD 15 99 86 01 BD 0E B6 A6 00 BD 03 06 26 BA 08 C3
S1 13 1260 BD 0D B4 24 B4 96 2C 26 03 BD 0E A1 33 32 36 37 FB
S1 13 1270 81 08 26 0D A6 00 BD 03 3B 24 9E BD 0A 35 BD 0D 85
S1 13 1280 B4 A6 00 BD 10 66 33 37 5D 27 07 86 0F BD 0E B6 C2
S1 13 1290 20 03 BD 0D 99 33 32 37 81 05 26 04 86 0A 20 1B AD
S1 13 12A0 81 06 26 04 86 0D 20 13 81 07 26 0D 86 E6 5F BD 80
S1 13 12B0 14 26 86 4F BD 14 14 20 05 86 10 BD 0E B6 33 5D 6A
S1 13 12C0 27 07 86 0E BD 0E B6 20 05 4F 97 2C 97 29 86 01 59
```

```
S1 13 12D0 20 0F 96 52 27 08 BD 14 0E 86 02 BD 10 86 4F 97 24
S1 13 12E0 51 97 52 97 2D 0D 39 DF 31 DE 3E 20 06 08 08 08 4C
S1 13 12F0 08 08 08 9C 4F 26 02 0C 39 A1 00 26 F0 E1 01 26 BB
S1 13 1300 EC 0D 39 37 C6 61 20 0D 37 C6 FF 20 08 37 C6 20 BB
S1 13 1310 20 03 37 C6 02 D7 35 33 8D CD 24 06 A6 02 E6 03 53
S1 13 1320 20 1C A7 00 E7 01 96 22 D6 23 A7 02 E7 03 6F 04 37
S1 13 1330 6F 05 8D 12 37 36 4F D6 35 BD 07 66 32 33 97 4D 5C
S1 13 1340 D7 4E DE 31 0D 39 9C 40 26 05 86 0D 7E 10 57 08 98
S1 13 1350 08 08 08 08 08 DF 4F 39 DF 61 48 97 43 48 9B 43 72
S1 13 1360 58 1B 16 CE 13 7B DF 43 4F DB 44 99 43 37 36 30 8B
S1 13 1370 EE 00 EE 00 31 31 AD 00 DE 61 39 14 FC 14 CC 14 02
S1 13 1380 9C 15 08 14 D2 14 A2 14 7D 14 74 14 6E 14 4D 14 F4
S1 13 1390 34 14 2E 14 65 14 5E 14 39 15 02 13 B1 15 95 14 02
S1 13 13A0 E1 14 B7 14 8A 14 E7 14 BD 14 90 14 ED 14 C3 14 93
S1 13 13B0 96 D6 4D 26 06 86 4F 8D 5B 20 04 86 86 8D 67 86 DD
S1 13 13C0 C6 D6 4E 26 61 86 5F 20 4B 8D 09 86 7E 20 1A 86 FE
S1 13 13D0 EE 5F 20 52 97 4D D7 4E 39 86 31 8D 37 20 35 86 B2
S1 13 13E0 43 8D 31 86 53 20 2D 86 CE 97 89 96 4D 97 8A 96 C4
S1 13 13F0 4E 97 8B C6 03 20 21 86 A7 C6 E7 20 04 86 A6 C6 7F
S1 13 1400 E6 37 5F 8D 21 32 C6 01 20 1C 86 30 20 06 86 37 E0
S1 13 1410 8D 02 86 36 97 89 C6 01 7E 05 B7 86 40 8D F5 86 8E
S1 13 1420 50 8D F1 86 82 5F 97 89 D7 8A C6 02 20 EA 8D DA C9
S1 13 1430 8D 1D 20 A5 8D D8 BD 13 B1 8D D3 8D CD 86 A6 C6 A7
S1 13 1440 02 8D E3 86 E6 C6 03 8D DD 8D E5 20 8C 8D 98 86 BE
S1 13 1450 01 97 73 96 81 D6 82 BD 13 D4 86 BD 20 8B 8D AE 41
S1 13 1460 BD 13 B1 20 C9 8D A7 BD 13 E7 8D 91 20 C0 8D 9A FE
S1 13 1470 8D 0E 20 BE 8D 98 BD 13 B1 8D 8F 20 F3 BD 13 E7 63
S1 13 1480 86 01 97 74 96 83 D6 84 20 CD 86 E4 C6 A4 20 16 5C
S1 13 1490 86 EA C6 AA 20 10 86 E8 C6 A8 20 0A 86 EB C6 A9 52
S1 13 14A0 20 04 86 E0 C6 A2 37 36 BD 14 0A 32 C6 01 8D 2E 4A
S1 13 14B0 32 5F 8D 2A 7E 14 32 86 C4 C6 84 20 19 86 CA C6 39
S1 13 14C0 8A 20 13 86 C8 C6 88 20 0D BD 13 D4 86 CB C6 89 4E
S1 13 14D0 20 04 86 C0 C6 82 37 D6 4E 8D 03 32 D6 4D 7E 14 84
S1 13 14E0 26 86 F4 C6 B4 20 27 86 FA C6 BA 20 21 86 F8 C6 12
S1 13 14F0 B8 20 1B BD 13 D4 86 F7 C6 B7 20 12 86 FB C6 B9 25
S1 13 1500 20 0C 86 F6 C6 B6 20 06 86 F0 C6 B2 20 00 37 D6 72
S1 13 1510 4D 26 10 D6 4E 5C 27 0B 8D 03 32 D6 4E 84 DF 7E CB
S1 13 1520 14 26 37 7C 00 4E 26 03 7C 00 4D 8D 0C 7D 00 4E 26
S1 13 1530 26 03 7A 00 4D 7A 00 4E 32 7E 13 E9 BD 13 D4 86 19
S1 13 1540 F7 20 14 86 F6 8D 10 86 4F 20 31 BD 13 D4 86 FE 05
S1 13 1550 20 05 BD 13 D4 86 FF D6 4D 26 DE 20 BE 86 2D C6 BB
S1 13 1560 03 7E 14 26 86 2C 20 F7 86 2D C6 05 8D F3 86 26 49
S1 13 1570 C6 06 8D ED 8D 04 86 26 20 E5 86 5D 7E 14 14 86 D0
S1 13 1580 2D C6 08 8D DC 8D EF 8D F1 86 27 20 D2 86 58 8D EF
S1 13 1590 EB 86 49 20 E7 86 3F 20 E3 86 34 8D DF 20 DD 86 15
S1 13 15A0 0E 20 D9 86 01 8D D5 86 0F 20 D1 BD 16 93 96 6C 59
S1 13 15B0 9A 6B 27 2D DE 22 DF 87 86 49 C6 4F BD 13 03 96 1B
S1 13 15C0 6C 27 0F DE 24 DF 77 CE 18 4D C6 FF 8D 42 C6 0E 82
S1 13 15D0 8D 3E 96 6B 27 0B DE 24 DF 75 CE 17 60 C6 ED 8D 2E
S1 13 15E0 2F 96 74 27 0B DE 24 DF 83 CE 19 5A C6 74 8D 20 00
S1 13 15F0 96 73 27 0B DE 24 DF 81 CE 19 CE C6 76 8D 11 96 25
S1 13 1600 6E 27 10 DE 24 DF 7B CE 07 8E C6 15 8D 02 20 03 E5
S1 13 1610 7E 05 F4 96 72 27 0B DE 24 DF 7F CE 1A 44 C6 1A A9
S1 13 1620 8D EE 96 6D 27 36 DE 22 DF 85 86 53 C6 54 BD 13 B4
S1 13 1630 08 DE 24 DF 79 CE 1A 5E C6 FF 8D D4 C6 61 8D D0 54
S1 13 1640 96 2E 36 84 FD 97 2E 96 77 D6 78 BD 13 C9 96 75 57
S1 13 1650 D6 76 CB 96 89 00 BD 13 C9 32 97 2E 96 70 27 24 6F
```

```
S1 13 1660 DE 24 DF 7D 96 2E 36 84 FD 97 2E 86 5A 16 BD 13 12
S1 13 1670 12 37 36 BD 14 F6 32 33 BD 15 4B BD 13 FD BD 09 0B
S1 13 1680 69 32 97 2E 8D 14 BD 06 21 86 53 8D 54 86 39 8D 6B
S1 13 1690 50 20 71 8D 05 8D 5C D7 42 39 96 2E 27 04 84 08 1D
S1 13 16A0 27 02 31 31 39 8D F3 8D 4A 86 53 8D 34 86 31 8D 3D
S1 13 16B0 30 CE 00 90 E6 00 37 CB 03 E7 00 5F 8D 20 8D 1E 0F
S1 13 16C0 8D 1C 8D 1A 32 4A 27 03 36 20 F7 53 37 30 8D 0E 7E
S1 13 16D0 31 86 0D 8D 0C 86 0A 8D 08 96 42 26 27 39 7E E0 C8
S1 13 16E0 EA 7E E3 A6 7E E3 50 7E E0 C8 7E E0 CA 7E E1 41 66
S1 13 16F0 7E E0 7E 86 12 C6 3C 8D 0F D6 42 27 06 4F 8D E1 D2
S1 13 1700 5A 26 FA 39 86 14 C6 34 8D D7 F7 80 07 01 39 8D E5
S1 13 1710 DC CE 17 5A 8D DA 20 CC C6 48 8D 27 37 DE F3 A6 E7
S1 13 1720 00 26 04 8D 25 20 F6 08 DF F3 33 81 1A 26 02 0C E7
S1 13 1730 39 DE 31 A7 00 08 DF 31 81 0D 27 07 5A 26 DD 86 FF
S1 13 1740 0D A7 00 CE 1C F9 DF 31 0D 39 8D 03 BD 18 DE CE 97
S1 13 1750 1C 78 DF F3 39 8D F8 6F 00 39 50 41 53 53 32 04 4C
S1 13 1760 86 20 DF 20 DE 22 A7 00 96 26 81 80 22 02 4C 08 F4
S1 13 1770 6F 00 97 26 DF 22 DE 20 39 DF 24 20 04 86 01 97 BC
S1 13 1780 26 DE 24 DF 22 39 8D D8 96 26 84 07 81 01 26 F6 A9
S1 13 1790 39 8D EE A6 00 27 0B 8D 2F 08 20 F7 8D C2 D1 26 98
S1 13 17A0 22 FA 39 20 BD 86 12 C6 3C 8D 16 C6 5A 4F 8D 18 B2
S1 13 17B0 5A 26 FA 86 02 8D 11 8D D8 86 03 8D 0B 86 14 C6 9F
S1 13 17C0 34 F7 80 07 8D 02 01 39 7E E3 A6 8D C4 8D AE 7E 89
S1 13 17D0 E1 41 36 A6 00 26 04 6D 01 27 06 8A 30 8D 83 6C 0C
S1 13 17E0 01 6F 00 32 39 C6 20 20 02 C6 80 A6 00 27 06 8D 6C
S1 13 17F0 B2 08 5A 26 F6 39 4D 2A 0A 36 86 2D 8D A5 32 40 6E
S1 13 1800 50 82 00 34 34 30 6F 00 6F 01 C0 10 82 27 25 04 E9
S1 13 1810 6C 00 20 F6 CB 10 89 27 8D B8 C0 E8 82 03 25 04 1C
S1 13 1820 6C 00 20 F6 CB E8 89 03 8D A8 C0 64 82 00 25 04 EF
S1 13 1830 6C 00 20 F6 CB 64 8D 9A C0 0A 25 04 6C 00 20 F8 55
S1 13 1840 CB 0A 8D 8E E7 00 6C 01 8D 88 31 31 39 4F 36 36 E5
S1 13 1850 36 36 86 05 36 30 8D 6F 81 20 27 FA 81 2D 26 0E 87
S1 13 1860 6C 04 8D 63 24 36 81 20 27 32 6D 00 27 41 80 30 3B
S1 13 1870 25 3D 81 09 22 39 A7 01 A6 02 E6 03 58 49 A7 02 9A
S1 13 1880 E7 03 58 49 58 49 25 27 EB 03 A9 02 25 21 EB 01 11
S1 13 1890 89 00 25 1B A7 02 E7 03 6A 00 20 C6 32 81 05 27 B9
S1 13 18A0 0F 31 32 33 6D 04 27 04 40 50 82 00 31 0C 39 31 3A
S1 13 18B0 31 31 31 31 4F 5F 0D 39 C6 20 8D 0B 24 06 A7 00 1D
S1 13 18C0 08 5A 26 F6 6F 00 39 DF 20 DE 22 A6 00 27 09 81 98
S1 13 18D0 2C 27 04 8D 02 0D 39 08 DF 22 DE 20 0C 39 86 11 F5
S1 13 18E0 C6 3C 8D 1D BD E3 50 81 02 26 F9 5F BD E3 50 81 E6
S1 13 18F0 03 27 08 A7 00 08 5C C1 80 25 F1 6F 00 86 13 C6 82
S1 13 1900 34 BD E3 A6 F7 80 07 39 86 3F BD E3 A6 BD E0 CC 2E
S1 13 1910 DF 22 DF 24 C6 80 BD E3 50 81 0F 26 0C 9C 24 27 E0
S1 13 1920 0C 09 A6 00 BD E3 A6 20 ED 81 18 26 17 8D 07 20 1B
S1 13 1930 2A 44 45 4C 2A 04 30 EE 00 31 31 BD E0 7E 8D 0C 42
S1 13 1940 DE 24 20 CC A7 00 81 0D 26 09 6F 00 86 01 97 26 8E
S1 13 1950 7E E1 41 5D 27 C0 08 5A 20 BC 37 36 E6 01 37 5F 77
S1 13 1960 A6 00 2A 07 5C 30 40 60 00 82 00 36 37 C6 10 37 74
S1 13 1970 30 A6 04 2A 09 40 60 05 82 00 A7 04 6C 01 4F 5F 69
S1 13 1980 66 02 66 03 24 04 EB 05 A9 04 46 56 66 02 66 03 50
S1 13 1990 6A 00 26 F0 4D 26 11 5D 26 0E A6 02 E6 03 66 01 B6
S1 13 19A0 24 25 40 50 82 00 20 1F 66 01 24 12 63 03 63 02 31
S1 13 19B0 53 43 6C 03 26 08 6C 02 26 04 CB 01 89 00 97 2B 41
S1 13 19C0 D7 2C E6 03 A6 02 0D 31 31 31 31 31 31 39 37 36 A6
S1 13 19D0 F6 01 37 5F A6 00 2A 07 5C 30 40 60 00 82 00 36 CB
```

```
S1 13 19F0 2A 0A 60 04 60 05 24 02 6A 04 6C 01 86 01 6D 02 EF
S1 13 1A00 2B 0B 4C 68 03 69 02 2B 04 81 11 26 F5 A7 00 A6 51
S1 13 1A10 04 E6 05 6F 04 6F 05 E0 03 A2 02 24 07 EB 03 A9 A3
S1 13 1A20 02 0C 20 01 0D 69 05 69 04 64 02 66 03 6A 00 26 3C
S1 13 1A30 E6 A6 04 E6 05 66 01 24 04 40 50 82 00 31 31 31 F3
S1 13 1A40 31 31 31 39 A6 01 48 AB 00 E6 01 58 49 58 49 EB 18
S1 13 1A50 01 A9 00 CB 19 89 36 A7 00 E7 01 44 56 39 DE 27 CE
S1 13 1A60 DF 29 30 96 29 A7 02 96 2A A7 03 39 30 EE 02 39 D6
S1 13 1A70 5F 6D 00 27 04 08 5C 20 F8 4F 39 30 EE 02 8D F0 CA
S1 13 1A80 5D 27 66 09 A6 00 08 81 20 26 5E 09 5A 20 F1 C6 52
S1 13 1A90 80 8D 05 6F 00 39 C6 20 DF 29 DE 27 A6 00 08 DF 08
S1 13 1AA0 27 DE 29 A7 00 27 04 08 5A 26 ED 39 30 EE 02 DF 85
S1 13 1AB0 27 37 8D BC 17 33 27 31 10 23 2E 4C DE 27 20 03 04
S1 13 1AC0 30 EE 02 DF 27 4D 27 21 6D 00 27 2E 4A 27 03 08 19
S1 13 1AD0 20 F6 5D 27 25 8D 2A 20 14 20 95 30 EE 02 5D 27 FF
S1 13 1AE0 08 A6 00 27 04 08 5A 20 F5 DF 27 6F 00 30 EE 00 0F
S1 13 1AF0 31 31 31 31 6E 00 C6 80 20 07 30 EE 02 20 EA C6 53
S1 13 1B00 20 A6 00 8D 08 4D 27 12 08 5A 26 F5 17 DF 29 DE 76
S1 13 1B10 27 A7 00 27 01 08 DF 27 DE 29 39 8D EF 4F 20 ED A5
S1 13 1B20 DE 27 6F 00 20 EF C6 01 D7 2F DF 27 8D AB 5D 27 9F
S1 13 1B30 2A D7 2C DE 20 8D A2 D7 2B D6 2C D1 2B 23 03 5F C2
S1 13 1B40 4F 39 DE 27 DF 29 DE 20 DF 2D DE 29 A6 00 08 DF 5E
S1 13 1B50 29 DE 2D A1 00 26 08 08 5A 26 ED D6 2F 4F 39 DE 9E
S1 13 1B60 20 08 DF 20 7C 00 2F D6 2B 5A 20 CB DF 27 DE 20 55
S1 13 1B70 A6 00 27 0F 08 DF 20 DE 27 E6 00 27 04 08 11 27 28
S1 13 1B80 EB 0C 39 DE 27 6D 00 26 F8 0D 39 36 8D 24 DE 27 5F
S1 13 1B90 DF 22 32 8D 2C DE 22 DF 27 6F 00 20 0A DF 29 8D 21
S1 13 1BA0 11 DE 29 DF 22 8D 17 36 96 2B 97 26 DE 2D DF 22 B4
S1 13 1BB0 32 39 96 26 97 2B 7F 00 26 DE 22 DF 2D 39 D9 DC 99
```

Page   RT/EDIT Tape Dump V1.0C

```
S1 13 0100 4F C6 32 D7 31 97 30 4F C6 01 D7 33 97 32 D6 31 E5
S1 13 0110 96 30 D7 35 97 34 D6 33 96 32 58 49 CB 00 89 13 65
S1 13 0120 37 36 4F 5F 30 EE 00 31 31 A7 00 E7 01 D6 33 96 02
S1 13 0130 32 CB 01 89 00 D7 33 97 32 D0 35 92 34 2D 05 26 3E
S1 13 0140 06 5D 26 03 7E 01 16 CE 10 00 BD 0D 7F BD 0D D3 C6
S1 13 0150 CE 10 00 BD 0D 14 86 10 C6 81 D7 28 97 27 34 34 DD
S1 13 0160 BD 0E 68 CE 10 00 BD 0F 00 4F C6 02 BD 0E E5 CE 19
S1 13 0170 10 81 DF 27 CE 00 36 BD 0E A0 CE 00 36 86 10 C6 15
S1 13 0180 81 D7 28 97 27 BD 0F 09 BD 0F 2A DE 27 DF 20 20 3E
S1 13 0190 03 52 54 00 CE 01 91 BD 0F 00 CE 10 81 BD 0F 76 E5
S1 13 01A0 25 03 7E 01 A8 7E E1 6A CE 00 36 86 10 C6 81 D7 7B
S1 13 01B0 28 97 27 BD 0F 09 BD 0F 2A DE 27 DF 20 20 03 4D 16
S1 13 01C0 45 00 CE 01 BF BD 0F 00 CE 10 81 BD 0F 76 24 03 C4
S1 13 01D0 7E 09 87 CE 00 36 86 10 C6 B1 D7 28 97 27 BD 0F A3
S1 13 01E0 09 BD 0F 2A DE 27 DF 20 20 03 4C 49 00 CE 01 EA 97
S1 13 01F0 BD 0F 00 CE 10 81 BD 0F 76 24 03 7E 04 B6 CE 00 61
S1 13 0200 36 86 10 C6 81 D7 28 97 27 BD 0F 09 BD 0F 2A DE 71
S1 13 0210 27 DF 20 20 03 46 2F 00 CE 02 15 BD 0F 00 CE 10 8D
S1 13 0220 81 BD 0F 76 24 03 7E 08 90 CE 00 36 86 10 C6 81 E9
S1 13 0230 D7 28 97 27 BD 0F 09 BD 0F 2A DE 27 DF 20 20 03 0B
S1 13 0240 4C 4F 00 CE 02 40 BD 0F 00 CE 10 81 BD 0F 76 24 6E
S1 13 0250 03 7E 06 CA CE 00 36 86 10 C6 81 D7 28 97 27 BD EE
S1 13 0260 0F 09 BD 0F 2A DE 27 DF 20 20 03 53 2C 00 CE 02 06
S1 13 0270 6B BD 0F 00 CE 10 81 BD 0F 76 24 03 7E 04 E8 CE 43
S1 13 0280 00 36 86 10 C6 81 D7 28 97 27 BD 0F 09 BD 0F 2A CF
S1 13 0290 DE 27 DF 20 20 03 53 41 00 CE 02 96 BD 0F 00 CE 9F
S1 13 02A0 10 81 BD 0F 76 24 03 7E 04 D6 CE 00 36 86 10 C6 98
S1 13 02B0 81 D7 28 97 27 BD 0F 09 BD 0F 2A DE 27 DF 20 20 0D
S1 13 02C0 03 45 4E 00 CE 02 C1 BD 0F 00 CE 10 81 BD 0F 76 96
S1 13 02D0 24 03 7E 04 F4 CE 00 36 86 10 C6 81 D7 28 97 27 DF
S1 13 02E0 BD 0F 09 BD 0F 2A DE 27 DF 20 20 03 4E 45 00 CE B7
S1 13 02F0 02 EC BD 0F 00 CE 10 81 BD 0F 76 24 03 7E 01 00 F9
S1 13 0300 CE 00 36 86 10 C6 81 D7 28 97 27 BD 0F 09 BD 0F AA
S1 13 0310 2A DE 27 DF 20 20 03 4C 2C 00 CE 03 17 BD 0F 00 5C
S1 13 0320 CE 10 81 BD 0F 76 24 03 7E 06 8B CE 10 00 86 10 7E
S1 13 0330 C6 81 D7 28 97 27 BD 0F 00 CE 10 81 BD 0F A7 D7 40
S1 13 0340 57 97 56 D6 57 96 56 C0 00 82 00 2D 08 26 03 5D 4F
S1 13 0350 27 03 7E 03 6E CE 10 00 BD 0D 7F 20 05 57 48 41 54
S1 13 0360 54 00 CE 03 5D BD 0D EF BD 0D D1 7E 01 47 BD 0C 24
S1 13 0370 40 D6 59 96 58 C0 40 82 00 2D 05 26 06 5D 26 03 B6
S1 13 0380 7E 03 A5 CE 10 00 BD 0D 7F 20 0E 4C 49 4E 45 20 A6
S1 13 0390 54 4F 4F 20 4C 4F 4E 47 00 CE 03 8B BD 0D EF BD 45
S1 13 03A0 0D D1 7E 01 47 BD 0A C8 D6 33 96 32 D0 31 92 30 82
S1 13 03B0 2D 08 26 03 5D 27 03 7E 03 E6 D6 33 96 32 58 49 7B
S1 13 03C0 CB 00 89 13 37 36 4F 5F 30 EE 00 31 31 A7 00 E7 99
S1 13 03D0 01 D6 59 96 58 C0 04 82 00 2D 08 26 03 5D 27 03 D0
S1 13 03E0 7E 03 E6 7E 01 47 BD 0B 18 D6 33 96 32 37 36 D6 E8
S1 13 03F0 31 96 30 CB 01 89 00 40 50 82 00 30 EB 01 A9 00 D6
S1 13 0400 31 31 2C 03 7E 04 27 CE 10 00 BD 0D 7F 20 0C 4D 0E
S1 13 0410 45 4D 4F 52 59 20 46 55 4C 4C 00 CE 04 0F BD 0D 4E
S1 13 0420 EF BD 0D D1 7E 01 47 D6 33 96 32 58 49 CB 00 89 B2
S1 13 0430 13 37 36 D6 57 96 56 30 EE 00 31 31 A7 00 E7 01 10
S1 13 0440 D6 33 96 32 86 20 BD 0E 53 CB 46 89 13 37 36 86 73
S1 13 0450 10 C6 81 D7 28 97 27 34 34 BD 0E 68 CE 10 00 BD 4E
S1 13 0460 0F 00 4F C6 20 BD 0E E5 CE 10 81 DF 27 30 EE 00 11
S1 13 0470 31 31 BD 0E A0 D6 33 96 32 DB 31 99 30 86 20 BD A2
S1 13 0480 0E 53 CB 46 89 13 37 36 86 10 C6 81 D7 28 97 27 53
```

Page 2 RT/EDIT V1.0C

```
S1 13 0490 34 34 BD 0E 68 CE 10 00 BD 0F 00 4F C6 21 37 4F 57
S1 13 04A0 C6 20 32 BD 0E CA CE 10 81 DF 27 30 EE 00 31 31 B6
S1 13 04B0 BD 0E A0 7E 01 47 4F C6 01 D7 5B 97 5A BD 09 B7 51
S1 13 04C0 D6 5D 96 5C C0 10 82 27 26 06 5D 26 03 7E 01 47 12
S1 13 04D0 BD 0A 86 7E 04 BD 4F C6 01 D7 5B 97 5A 86 27 C6 E0
S1 13 04E0 0F D7 5F 97 5E 7E 04 EB BD 0B 5B 4F 5F D7 61 97 C1
S1 13 04F0 60 7E 05 0A 4F C6 01 D7 61 97 60 4F C6 01 D7 5B 7E
S1 13 0500 97 5A 86 27 C6 0F D7 5F 97 5E BD 06 70 BD 09 B7 99
S1 13 0510 D6 5B 96 5A 37 36 D6 5F 96 5E CB 01 89 00 40 50 3B
S1 13 0520 82 00 30 EB 01 A9 00 31 31 2D 08 26 03 5D 27 03 39
S1 13 0530 7E 05 40 D6 5D 96 5C C0 10 82 27 2C 03 7E 05 97 0D
S1 13 0540 D6 61 96 60 C0 00 82 00 26 06 5D 26 03 7E 05 7B 88
S1 13 0550 CE 10 00 86 10 C6 81 D7 28 97 27 BD 0F 00 4F C6 3E
S1 13 0560 1A BD 0F 25 CE 10 81 DF 27 CE 10 00 BD 0E 99 D6 FF
S1 13 0570 59 96 58 CB 01 89 00 D7 59 97 58 D6 59 96 58 C0 DF
S1 13 0580 00 82 00 26 06 5D 26 03 7E 01 47 CE 10 00 BD 0D C5
S1 13 0590 7F BD 0D AB 7E 01 47 D6 65 96 64 86 20 BD 0E 53 A4
S1 13 05A0 CB 46 89 13 BD 0F CE 86 10 C6 81 D7 28 97 27 BD A9
S1 13 05B0 0F 09 D6 65 96 64 DB 31 99 30 86 20 BD 0E 53 CB 86
S1 13 05C0 46 89 13 BD 0F CE BD 0F 09 CE 10 81 BD 0E 7A D7 5B
S1 13 05D0 63 97 62 D6 67 96 66 DB 63 99 62 CB 01 89 00 C0 34
S1 13 05E0 80 82 00 2D 08 26 03 5D 27 03 BD 06 67 CE 10 00 18
S1 13 05F0 86 10 C6 81 D7 28 97 27 BD 0F 00 D6 65 96 64 86 D6
S1 13 0600 20 BD 0E 53 CB 46 89 13 BD 0F CE BD 0F 09 D6 65 51
S1 13 0610 96 64 DB 31 99 30 86 20 BD 0E 53 CB 46 89 13 BD D9
S1 13 0620 0F CE BD 0F 09 4F C6 0D BD 0F 25 CE 10 81 DF 27 9C
S1 13 0630 CE 10 00 BD 0E 99 D6 67 96 66 DB 63 99 62 CB 01 36
S1 13 0640 89 00 D7 67 97 66 D6 59 96 58 CB 01 89 00 D7 59 40
S1 13 0650 97 58 D6 59 96 58 C0 03 82 00 2D 08 26 03 5D 27 63
S1 13 0660 03 BD 06 67 7E 05 0D CE 10 00 BD 0D 7F BD 0D AB 2D
S1 13 0670 4F 5F D7 67 97 66 4F 5F D7 59 97 58 20 01 00 CE D1
S1 13 0680 06 7E DF 27 CE 10 00 BD 0E 99 39 BD 0B 5B BD 09 78
S1 13 0690 B7 D6 5D 96 5C C0 10 82 27 26 06 5D 26 03 7E 01 D0
S1 13 06A0 47 D6 5B 96 5A 37 36 D6 5F 96 5E CB 01 89 00 40 B3
S1 13 06B0 50 82 00 30 EB 01 A9 00 31 31 2D 08 26 03 5D 27 5B
S1 13 06C0 03 7E 01 47 BD 0A 86 7E 06 8E BD 0B EA D6 59 96 87
S1 13 06D0 58 C0 04 82 00 2C 03 7E 01 47 CE 10 00 BD 0C EA F2
S1 13 06E0 86 10 C6 81 D7 28 97 27 4F C6 0D BD 0F 25 BD 0F 8D
S1 13 06F0 2A DE 27 DF 20 CE 10 00 BD 0F 00 CE 10 81 BD 0F F3
S1 13 0700 30 D7 59 97 58 D6 59 96 58 C0 00 82 00 26 03 5D B1
S1 13 0710 27 03 7E 07 64 86 10 C6 81 D7 28 97 27 4F C6 1A F9
S1 13 0720 BD 0F 25 BD 0F 2A DE 27 DF 20 CE 10 00 BD 0F 00 30
S1 13 0730 CE 10 81 BD 0F 30 D7 59 97 58 D6 59 96 58 C0 00 5E
S1 13 0740 82 00 26 06 5D 26 03 7E 06 CA CE 10 00 BD 0D 7F FC
S1 13 0750 20 06 2A 45 4F 46 2A 00 CE 07 52 BD 0D EF BD 0D 97
S1 13 0760 D1 7E 01 47 BD 0B 18 D6 33 96 32 DB 31 99 30 86 E2
S1 13 0770 20 BD 0E 53 CB 46 89 13 37 36 20 01 00 CE 07 7C AB
S1 13 0780 DF 27 30 EE 00 31 31 BD 0E A0 D6 33 96 32 58 49 02
S1 13 0790 CB 00 89 13 37 36 CE 10 00 86 10 C6 81 D7 28 97 30
S1 13 07A0 27 BD 0F 00 CE 10 81 BD 0F A7 30 EE 00 31 31 A7 59
S1 13 07B0 00 E7 01 D6 59 96 58 C0 21 82 00 2D 05 26 06 5D 12
S1 13 07C0 26 03 7E 08 08 D6 33 96 32 DB 31 99 30 86 20 BD 65
S1 13 07D0 0E 53 CB 46 89 13 37 36 86 10 C6 81 D7 28 97 27 00
S1 13 07E0 34 34 BD 0E 68 CE 10 00 BD 0F 00 4F C6 21 37 D6 7D
S1 13 07F0 59 96 58 C0 21 82 00 32 BD 0E CA CE 10 81 DF 27 1F
S1 13 0800 30 EE 00 31 31 BD 0E A0 D6 33 96 32 86 20 BD 0E B7
S1 13 0810 53 CB 46 89 13 37 36 86 10 C6 81 D7 28 97 27 34 99
```

```
S1 13 0820 34 BD 0E 68 CE 10 00 BD 0F 00 D6 59 96 58 C0 01 D5
S1 13 0830 82 00 BD 0E E5 CE 10 81 DF 27 30 EE 00 31 31 BD E0
S1 13 0840 0E A0 CE 10 00 86 10 C6 81 D7 28 97 27 BD 0F 00 B2
S1 13 0850 CE 10 81 BD 0E 7A D0 59 92 58 D7 69 97 68 86 10 08
S1 13 0860 C6 81 D7 28 97 27 34 34 BD 0E 68 CE 10 00 BD 0F 3B
S1 13 0870 00 D6 59 96 58 CB 01 89 00 37 D6 69 96 68 32 BD 9F
S1 13 0880 0E CA CE 10 81 DF 27 CE 10 00 BD 0E 99 7E 06 E0 81
S1 13 0890 86 10 C6 81 D7 28 97 27 34 34 BD 0E 68 CE 10 00 41
S1 13 08A0 BD 0F 00 4F C6 03 37 4F C6 20 32 BD 0E CA CE 10 4F
S1 13 08B0 81 DF 27 CE 00 36 BD 0E A0 4F C6 01 D7 65 97 64 F1
S1 13 08C0 D6 31 96 30 D7 6B 97 6A D6 65 96 64 58 49 CB 00 73
S1 13 08D0 89 13 BD 0F CE C0 00 82 00 26 06 5D 26 03 7E 09 63
S1 13 08E0 32 CE 00 36 86 10 C6 81 D7 28 97 27 BD 0F 09 BD A2
S1 13 08F0 0F 2A DE 27 DF 20 D6 65 96 64 86 20 BD 0E 53 CB F3
S1 13 0900 46 89 13 BD 0F CE BD 0F 09 D6 65 96 64 DB 31 99 B8
S1 13 0910 30 86 20 BD 0E 53 CB 46 89 13 BD 0F CE BD 0F 09 C3
S1 13 0920 CE 10 81 BD 0F 30 C0 00 82 00 26 03 5D 27 03 7E F8
S1 13 0930 09 81 D6 65 96 64 CB 01 89 00 D7 65 97 64 D0 6B 2D
S1 13 0940 92 6A 2D 05 26 06 5D 26 03 7E 08 C8 CE 10 00 BD DA
S1 13 0950 0D 7F CE 00 36 86 10 C6 81 D7 28 97 27 BD 0F 09 94
S1 13 0960 20 0D 20 20 20 4E 4F 54 20 46 4F 55 4E 44 00 CE 9B
S1 13 0970 09 62 BD 0F 00 CE 10 81 BD 0D EF BD 0D D1 7E 01 0A
S1 13 0980 47 BD 0A 86 7E 01 47 BD 0B EA CE 10 00 BD 0D 7F 30
S1 13 0990 D6 59 96 58 BD 0D FC 20 12 20 20 4C 49 4E 45 53 83
S1 13 09A0 20 41 56 41 49 4C 41 42 4C 45 00 CE 09 99 BD 0D 68
S1 13 09B0 EF BD 0D D1 7E 01 47 4F C6 01 D7 6D 97 6C 86 27 D9
S1 13 09C0 C6 10 D7 5D 97 5C D6 6D 96 6C 58 49 CB 00 89 13 D9
S1 13 09D0 BD 0F CE D7 6F 97 6E D6 6F 96 6E C0 00 82 00 26 7D
S1 13 09E0 06 5D 26 03 7E 0A 2B D6 5B 96 5A D0 6F 92 6E 2D 37
S1 13 09F0 08 26 03 5D 27 03 7E 0A 2B D6 6F 96 6E D0 5B 92 82
S1 13 0A00 5A 26 06 5D 26 03 7E 0A 69 D6 6F 96 6E D0 5D 92 DD
S1 13 0A10 5C 2D 08 26 03 5D 27 03 7E 0A 2B D6 6F 96 6E D7 BE
S1 13 0A20 5D 97 5C D6 6D 96 6C D7 65 97 64 D6 6D 96 6C CB E6
S1 13 0A30 01 89 00 D7 6D 97 6C D6 6D 96 6C 37 36 D6 31 96 92
S1 13 0A40 30 CB 01 89 00 40 50 82 00 30 EB 01 A9 00 31 31 E4
S1 13 0A50 26 03 5D 27 03 7E 09 C6 D6 5D 96 5C C0 10 82 27 F7
S1 13 0A60 26 03 5D 27 03 7E 0A 79 39 D6 5B 96 5A D7 5D 97 AC
S1 13 0A70 5C D6 6D 96 6C D7 65 97 64 D6 5D 96 5C CB 01 89 20
S1 13 0A80 00 D7 5B 97 5A 39 CE 10 00 BD 0D 7F D6 65 96 64 AA
S1 13 0A90 86 20 BD 0E 53 CB 46 89 13 BD 0F CE 86 10 C6 81 6A
S1 13 0AA0 D7 28 97 27 BD 0F 09 D6 65 96 64 DB 31 99 30 86 20
S1 13 0AB0 20 BD 0E 53 CB 46 89 13 BD 0F CE BD 0F 09 CE 10 FA
S1 13 0AC0 81 BD 0D EF BD 0D D1 39 4F C6 01 D7 33 97 32 D6 55
S1 13 0AD0 31 96 30 D7 35 97 34 D6 57 96 56 37 36 D6 33 96 1F
S1 13 0AE0 32 58 49 CB 00 89 13 BD 0F CE 40 50 82 00 30 EB 01
S1 13 0AF0 01 A9 00 31 31 26 06 5D 26 03 7E 0B 17 D6 33 96 F5
S1 13 0B00 32 CB 01 89 00 D7 33 97 32 D0 35 92 34 2D 05 26 64
S1 13 0B10 06 5D 26 03 7E 0A D7 39 4F C6 01 D7 33 97 32 D6 EE
S1 13 0B20 31 96 30 D7 35 97 34 D6 33 96 32 58 49 CB 00 89 2D
S1 13 0B30 13 BD 0F CE C0 00 82 00 26 06 5D 26 03 7E 0B 5A 2D
S1 13 0B40 D6 33 96 32 CB 01 89 00 D7 33 97 32 D0 35 92 34 DD
S1 13 0B50 2D 05 26 06 5D 26 03 7E 0B 27 39 20 02 2C 00 CE A8
S1 13 0B60 0B 5D 86 10 C6 81 D7 28 97 27 BD 0F 00 BD 0F 2A BD
S1 13 0B70 DE 27 DF 20 34 34 BD 0E 68 CE 10 00 BD 0F 00 4F D9
S1 13 0B80 C6 03 37 4F C6 05 32 BD 0E CA CE 10 81 BD 0F 30 25
S1 13 0B90 D7 71 97 70 86 10 C6 81 D7 28 97 27 34 34 BD 0E 35
S1 13 0BA0 68 CE 10 00 BD 0F 00 4F C6 03 37 4F C6 04 32 BD D8
```

```
S1 13 0BB0 0E CA CE 10 81 BD 0F A7 D7 5B 97 5A 86 10 C6 81 87
S1 13 0BC0 D7 28 97 27 34 34 BD 0E 68 CE 10 00 BD 0F 00 D6 49
S1 13 0BD0 71 96 70 CB 03 89 00 37 4F C6 04 32 BD 0E CA CE 5E
S1 13 0BE0 10 81 BD 0F A7 D7 5F 97 5E 39 4F 5F D7 59 97 58 CC
S1 13 0BF0 4F C6 01 D7 33 97 32 D6 31 96 30 D7 35 97 34 D6 8E
S1 13 0C00 33 96 32 58 49 CB 00 89 13 BD 0F CE C0 00 82 00 01
S1 13 0C10 26 06 5D 26 03 BD 0C 33 D6 33 96 32 CB 01 89 00 FC
S1 13 0C20 D7 33 97 32 D0 35 92 34 2D 05 26 06 5D 26 03 7E C0
S1 13 0C30 0B FF 39 D6 59 96 58 CB 01 89 00 D7 59 97 58 39 A3
S1 13 0C40 CE 10 00 86 10 C6 81 D7 28 97 27 BD 0F 00 CE 10 7E
S1 13 0C50 81 BD 0E 7A D7 59 97 58 39 4F 36 36 36 36 86 05 C0
S1 13 0C60 36 30 8D 6F 81 20 27 FA 81 2D 26 0E 6C 04 8D 63 1A
S1 13 0C70 24 36 81 20 27 32 6D 00 27 41 80 30 25 3D 81 09 AB
S1 13 0C80 22 39 A7 01 A6 02 E6 03 58 49 A7 02 E7 03 58 49 F7
S1 13 0C90 58 49 25 27 EB 03 A9 02 25 21 EB 01 89 00 25 1B CF
S1 13 0CA0 A7 02 E7 03 6A 00 20 C6 32 81 05 27 0F 31 32 33 D9
S1 13 0CB0 6D 04 27 04 40 50 82 00 31 0C 39 31 31 31 31 31 17
S1 13 0CC0 4F 5F 0D 39 C6 20 8D 0B 24 06 A7 00 08 5A 26 F6 5F
S1 13 0CD0 6F 00 39 DF 20 DE 22 A6 00 27 09 81 2C 27 04 8D 2E
S1 13 0CE0 02 0D 39 08 DF 22 DE 20 0C 39 86 11 C6 3C 8D 1D 29
S1 13 0CF0 BD E3 50 81 02 26 F9 5F BD E3 50 81 03 27 08 A7 B5
S1 13 0D00 00 08 5C C1 80 25 F1 6F 00 86 13 C6 34 BD E3 A6 DC
S1 13 0D10 F7 80 07 39 86 3F BD E3 A6 BD E0 CC DF 22 DF 24 A0
S1 13 0D20 C6 80 BD E3 50 81 0F 26 0C 9C 24 27 0C 09 A6 00 25
S1 13 0D30 BD E3 A6 20 ED 81 18 26 17 8D 07 20 2A 44 45 4C D3
S1 13 0D40 2A 04 30 EE 00 31 31 BD E0 7E 8D 0C DE 24 20 CC 4F
S1 13 0D50 A7 00 81 0D 26 09 6F 00 86 01 97 26 7E E1 41 5D 7B
S1 13 0D60 27 C0 08 5A 20 BC 86 20 DF 20 DE 22 A7 00 96 26 52
S1 13 0D70 81 80 22 02 4C 08 6F 00 97 26 DF 22 DE 20 39 DF B3
S1 13 0D80 24 20 04 86 01 97 26 DE 24 DF 22 39 8D D8 96 26 76
S1 13 0D90 84 07 81 01 26 F6 39 8D EE A6 00 27 0B 8D 2F 08 D6
S1 13 0DA0 20 F7 8D C2 D1 26 22 FA 39 20 BD 86 12 C6 3C 8D 89
S1 13 0DB0 16 C6 5A 4F 8D 18 5A 26 FA 86 02 8D 11 8D D8 86 7A
S1 13 0DC0 03 8D 0B 86 14 C6 34 F7 80 07 8D 02 01 39 7E E3 48
S1 13 0DD0 A6 8D C4 8D AE 7E E1 41 36 A6 00 26 04 6D 01 27 A2
S1 13 0DE0 06 8A 30 8D 83 6C 01 6F 00 32 39 C6 20 20 02 C6 1A
S1 13 0DF0 80 A6 00 27 06 8D B2 08 5A 26 F6 39 4D 2A 0A 36 EF
S1 13 0E00 86 2D 8D A5 32 40 50 82 00 34 34 30 6F 00 6F 01 3E
S1 13 0E10 C0 10 82 27 25 04 6C 00 20 F6 CB 10 89 27 8D B8 DA
S1 13 0E20 C0 E8 82 03 25 04 6C 00 20 F6 CB E8 89 03 8D A8 72
S1 13 0E30 C0 64 82 00 25 04 6C 00 20 F6 CB 64 8D 9A C0 0A 3D
S1 13 0E40 25 04 6C 00 20 F8 CB 0A 8D 8E E7 00 6C 01 8D 88 98
S1 13 0E50 31 31 39 37 C6 08 37 30 5F 58 49 24 04 EB 01 89 EA
S1 13 0E60 00 6A 00 26 F4 31 31 39 DE 27 DF 29 30 96 29 A7 BC
S1 13 0E70 02 96 2A A7 03 39 30 EE 02 39 5F 6D 00 27 04 08 71
S1 13 0E80 5C 20 F8 4F 39 30 EE 02 8D F0 5D 27 66 09 A6 00 2C
S1 13 0E90 81 20 26 5F 6F 00 5A 20 F1 C6 80 8D 05 6F 00 39 CE
S1 13 0EA0 C6 20 DF 29 DE 27 A6 00 08 DF 27 DE 29 A7 00 27 C2
S1 13 0EB0 04 08 5A 26 ED 39 30 EE 02 DF 27 37 8D BC 17 33 8C
S1 13 0EC0 27 31 10 23 2E 4C DE 27 20 03 30 EE 02 DF 27 4D 7E
S1 13 0ED0 27 21 6D 00 27 2E 4A 27 03 08 20 F6 5D 27 25 8D 3C
S1 13 0EE0 2A 20 14 20 95 30 EE 02 5D 27 08 A6 00 27 04 08 66
S1 13 0EF0 5A 20 F5 DF 27 6F 00 30 EE 00 31 31 31 31 6E 00 BA
S1 13 0F00 C6 80 20 07 30 EE 02 20 EA C6 20 A6 00 8D 08 4D D8
S1 13 0F10 27 12 08 5A 26 F5 17 DF 29 DE 27 A7 00 27 01 08 1C
S1 13 0F20 DF 27 DE 29 39 8D EF 4F 20 ED DE 27 6F 00 20 EF 1C
S1 13 0F30 C6 01 D7 2F DF 27 8D AB 5D 27 2A D7 2C DE 20 8D 66
```

```
S1 13 0F40 A2 D7 2B D6 2C D1 2B 23 03 5F 4F 39 DE 27 DF 29 E1
S1 13 0F50 DE 20 DF 2D DE 29 A6 00 08 DF 29 DE 2D A1 00 26 F4
S1 13 0F60 08 08 5A 26 ED D6 2F 4F 39 DE 20 08 DF 20 7C 00 F2
S1 13 0F70 2F D6 2B 5A 20 CB DF 27 DE 20 A6 00 27 0F 08 DF 31
S1 13 0F80 20 DE 27 E6 00 27 04 08 11 27 EB 0C 39 DE 27 6D 45
S1 13 0F90 00 26 F8 0D 39 36 8D 24 DE 27 DF 22 32 8D 2C DE 33
S1 13 0FA0 22 DF 27 6F 00 20 0A DF 29 8D 11 DE 29 DF 22 8D 41
S1 13 0FB0 17 36 96 2B 97 26 DE 2D DF 22 32 39 96 26 97 2B 6D
S1 13 0FC0 7F 00 26 DE 22 DF 2D 39 7E 0C 59 7E 0D F4 F7 13 C7
S1 13 0FD0 01 B7 13 00 FE 13 00 A6 00 E6 01 39 BD 03 1A 24 6D
```