SUBJECT: METHOD OF PREPARING SUBROUTINES FOR THE SUBROUTINE LIBRARY

To: Applications Group

From: John Carr and John T. Gilmore, Jr.

Date: September 24, 1951

Abstract: The form in which a programmer submits a subroutine for the library, as well as the procedure and conventions for writing these subroutines, is described below. (This memorandum is presented for immediate use, and will be later superseded by a more thorough Engineering Note. Criticisms and corrections will be appreciated.)

Whirlwind is completely dependent on its library of subroutines for efficient use of the machine. So far most coding has of necessity been from the beginning for each individual problem. Use of the subroutines, pretested, will enable a programmer to save many hours of work. Building a library of subroutines must be a cooperative task, with the writing shared by everyone, just as its use will be.

What subroutines should be included in the library? A glance at the I.D..C programs will show that the Cambridge University group has assembled a very complete set of subroutines covering many different eventualities. If a programmer should happen to have a portion of a main program that he thinks might later on be useful, it probably should be in the subroutine library. If in doubt, he should consult one of the editors; but it will usually pay to err on the side of more subroutines.

From time to time individuals may be asked to write a particular subroutine not then in the library. Such work will be distributed as evenly as possible among all programmers.

The subroutine library will be catalogued and filed in four forms:

1. Table of Contents
2. Subroutine Specifications
3. Complete Program
4. Paper Tape
The Table of Contents will be filed on single sheets by categories for insertion in a loose-leaf binder. Each programmer will have a complete out of these. Subroutines will be filed under different categories as listed in Table I. The Table of Contents will attempt to give all the subroutines cataloged under Matrices, for example, on one or two sheets, listing the subroutine and the specifications that must be known in order to use it in a main program. A programmer may thus look under Matrices and select the subroutine most nearly fitting his requirements.

The Subroutine Specifications sheets will be kept by the programmer in a separate loose-leaf folder or in a separate section of the same folder. These contain a somewhat more thorough set of specifications than in the Table of Contents. Programmers should be able to code using these sheets without the necessity of referring to the actual coded subroutine. Copies of the subroutine, in coded form with detailed description, will be issued on request.

The paper tape containing the subroutine will be available only to workers in the tape preparation room.

In preparing a subroutine for the library, the programmer should obtain one of the blank Subroutine Specifications forms (similar to the form attached) and fill it out completely. The box marked "Classification" is to be filled in using one of the categories: closed, open, interpretive or special.

We follow the EDSSC notation here. A "closed" subroutine is one called in by the order sn.n, where n is the first address of the subroutine in the memory. This order is one of the orders in the main program. The subroutine is designed so that after this it returns control to the register immediately following the sn.n order, or if any program parameters are present, to the register after the last program parameter.

An "open" subroutine is similar to a "closed" subroutine but does not return control to the registers after the entry order. It instead delivers control to the order following the last order of the subroutine.

An "interpretive" subroutine, although its form is similar to that of a "closed" subroutine, has a different purpose. These subroutines interpret a series of program parameters following the entry order as "orders" on a higher level, instructing the machine to perform a different combination of operations whenever a different "order" appears. Examples of interpretive subroutines now being coded are subroutines for floating point and double-precision arithmetic.

A "special" subroutine is one that is not necessarily open or closed, but is used for certain special purposes and usually not included in main programs. An example is the "post mortem" routine, which prints
out various contents of storage after a program has been completed (for
the purpose of finding a programming error).

The "No. of Registers in the Subroutine" will be the total
number of registers, other than temporary storage, occupied by the sub-
routine. The "Temporary Registers Used" will be the numbers (e.g., d, 1t - 5t
or 1t = St) of temporary registers used during a reference to the subroutine
but free for any other use between references.

The "average time" may be a judicious guess as to the mean length
of the path of the control of the program considering all cycles. The
"maximum time" again may be difficult to determine exactly, but an approxi-
mation should at least be given. If either value is in doubt or needs
further explanation, the programmer should elaborate in the "Description".
Time is measured in terms of the number of orders, which will provide a
reasonably accurate indication even though different orders require
different amounts of time.

Two types of parameters will generally be encountered in the
use of subroutines. "Preset" parameters will be those parameters which
are fixed once and for all at the start of a main program, and which will
be fixed in the subroutine at the time of input. An example is the
"digit length constant", which for most output print routines usually
must be decided by the programmer once and for all. This will be placed
in the proper position in the subroutine upon conversion of the Flexo-
writer standard tape.

"Program" parameters are parameters which may have different
values at different points along the control path of a main program.
Program parameters are normally placed after the entry order sp n, where
n is the address of the first order of the subroutine. An example is the
number to be printed in some of the short output print routines, which is
stored immediately following the entry order. Sometimes more than one
program parameter is used; sometimes a program parameter is placed in the
accumulator, rather than the registers following the entry order, which
are known as ul, u2, etc.

The form, range of values, and significance of any preset
parameters should be inserted in the space provided to the right of the
preset parameter register numbers "vl", "v2", etc. Any program parameters
contained in any of the registers of the arithmetic element, in temporary
storage registers or the registers following the sp order which enters
the subroutine (these registers are designated as "ul", "u2", etc.)
should be entered in the proper space. Any results which upon leaving
the subroutine are available in the registers of the arithmetic element,
in temporary storage registers, or in registers ul, u2, etc., should be
listed. Note that in closed subroutines, the contents of the A-Register
will always be the address ul on entering the subroutine. This will
consequently be immaterial, and need not be specified.
The "Description" of the subroutine is important for several reasons. It should give a description of what the machine does in this subroutine, as clearly and concisely as possible. It also should detail how the machine does this so that without the program an interested reader may understand the general idea of the method. It should be explained thoroughly enough so that any reasonably experienced programmer can follow the argument.

The description should also contain if possible the kind of alarms that could occur during the subroutine, where they might occur, and why.

A statement of how the subroutine was tested is also important, so that failures at a later date due to lack of sufficient testing may be avoided. This should be written up on a Subroutine Test Report form and given, along with any output results in the form of photographs or printed data, to the editors. This report will be filed by the editors of the program library for reference in case of later subroutine failures.

The subroutine itself, in coded form, should be preceded by a short abstract, which may be a duplicate of that part of the description stating what the subroutine does.

For use with the present Conversion Program (T 464-5), the following rules must be adhered to in order for a program to be converted correctly:

1. All subroutines will be written with zero as the location of the first word. Address sections of instructions will be expressed to the base ten.

2. All instructions whose address sections are relative to the position of the subroutine in storage will be followed by the letter "r".

3. All instructions requiring the addition of a preset parameter will be followed by the letter "a" with any digit 1 through 7, the digit being chosen to correspond with the desired one of the seven possible preset parameters. All instructions requiring the subtraction of a preset parameter will be followed by the letter "a" and any digit 1 through 7.

4. If the programmer using the subroutine does not assign a value to a given preset parameter its value will automatically be zero. Consequently, when a subroutine is written, all parameters should be so chosen that zero is the most likely value of each, so that values of parameters will frequently not need to be specified by a programmer.
5. A block of registers is to be set aside by the programmer for use as temporary storage by all subroutines and by the main program as needed. These temporary registers will be designated in the following manner and assigned consecutively:

\[
\begin{align*}
    d &= \text{address of a register of which only the address section is temporary, digital positions 0-4 being always zero.} \\
    1t &= \text{address of first regular temporary register.} \\
    2t &= \text{address of second regular temporary register.} \\
    3t &= \text{address of third regular temporary register.} \\
    nt &= \text{address of } n\text{-th regular temporary register.}
\end{align*}
\]

Registers to be used as temporary storage by interpretive subroutines and by no other subroutines or the main program will be designated as follows and assigned consecutively:

\[
\begin{align*}
    dx &= \text{address of a register of which only the address section is temporary, digital positions 0-4 being zero.} \\
    ltx &= \text{addresses of the sections of the multiple 2tx register accumulator used by the interpretive subroutine, followed by other 3tx etc. registers used for special purposes by the interpretive subroutines.}
\end{align*}
\]

The parameter \(d\), that is the address of the zero-th temporary register, is to be assigned by the programmer and indicated in the title of the program. It remains constant throughout the program. This will also be the case for \(dx\) if (and only if) an interpretive subroutine is used. The programmer must examine the specifications of all of his subroutines to determine the number of temporary registers used by each one and provide as many consecutive temporary registers as needed to include all registers that are used by at least one subroutine. The initial values of the contents of the registers assigned as temporary need not be specified except that the zero-th register must be set to contain \(r10\) (or \(p0\)) during input. The registers assigned as \(dx\) and \(tx\) registers can never be the same as those assigned as \(d\) and \(t\) registers. If none of the subroutines refer to the \(d\) (temporary address) register, it may of course be used for some other purpose. This is true also for register \(dx\).
Should the following situations arise, the conversion program will be able to take care of them:

1. A relative instruction which requires the addition of one or more preset parameters as well as the relative factor.

2. A preset parameter which requires the addition of a preset parameter (only if the second parameter has preceded it numerically in the series \( v_1 - v_6 \)). Thus parameter \( v_2 \) may be added to \( v_3 \), for example, but not vice versa.

3. A preset parameter which is relative itself.

4. A gap in the sequence of a subroutine.

The following conventions should always be followed in actually writing the subroutines. The same conventions may also be convenient in main programs.

1. A brief description of the action of every order should be given. Orders may, however, be described singly or in blocks.

2. Horizontal dotted lines should be drawn after orders where changes of control are conditional, i.e., \( \text{cp} \) orders.

3. Horizontal solid lines should be drawn after orders where definite changes of control occur, i.e., \( \text{sp} \) orders.

4. Arrows with heads pointing right should indicate points of entry in the program from other positions, i.e., from \( \text{cp} \) and \( \text{sp} \) orders. These should be at the left of the register number where entry occurs. At the left of the arrows the order or orders from which control is transferred should be written.

5. Orders used only as constants, i.e., so-called "pseudo-orders" should have double vertical bars at the left.

6. Orders used for control purposes and also as "pseudo-orders" should have single vertical bars at the left.

7. Orders which may be changed during the course of control should be enclosed in parentheses.

These conventions are illustrated by included examples. The first is a subroutine for printing out the value \( n \), when a register contains \( n \times 2^{-15} \), i.e., for printing out "integers". This routine suppresses first zeroes in the number, restoring zero print after the first non-zero digit is printed.
The second is a subroutine for printing out "short" decimals in the range \(-0.99999 \leq d \leq +0.99999\). A variable preset parameter allows only the first \(n\) decimal digits to be printed if so wanted, where \(1 \leq n\).

Since all subroutines are intended for use later when electrostatic storage may replace the present test storage, the following rules must be obeyed:

1. Register zero will always contain zero. Thus "ca0" will indicate that the accumulator is now to contain zero.

2. No transfers to or exchanges with test storage as a "waste store" shall be made. Temporary storage must be used for this.

The mechanics of putting a subroutine into the library is as follows: subroutines, after thorough testing on the machine, should be prepared in the proper form and given to Dorothy Lenihan. She will assign a number, catalogue the routine in the Table of Contents, edit the English, and check the description of the code for thoroughness and ability to be understood. Another editor will then check the accuracy of the programming. The programming will also be examined in an attempt to reduce the number of registers. The subroutine will then be typed and stored in the library.

The following is the vocabulary of the present Conversion Program (T 464-5).

<table>
<thead>
<tr>
<th>Orders</th>
<th>Decimal Value</th>
<th>Orders</th>
<th>Decimal Value</th>
<th>Orders</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ri</td>
<td>0</td>
<td>ck</td>
<td>11</td>
<td>ao</td>
<td>22</td>
</tr>
<tr>
<td>rs</td>
<td>1</td>
<td>qa</td>
<td>12</td>
<td>qf</td>
<td>23</td>
</tr>
<tr>
<td>rf</td>
<td>2</td>
<td>qe</td>
<td>13</td>
<td>mr</td>
<td>24</td>
</tr>
<tr>
<td>rb</td>
<td>3</td>
<td>qp</td>
<td>14</td>
<td>mh</td>
<td>25</td>
</tr>
<tr>
<td>rd</td>
<td>4</td>
<td>sq</td>
<td>15</td>
<td>dv</td>
<td>26</td>
</tr>
<tr>
<td>rc</td>
<td>5</td>
<td>sa</td>
<td>16</td>
<td>sl</td>
<td>27</td>
</tr>
<tr>
<td>qh</td>
<td>6</td>
<td>oe</td>
<td>17</td>
<td>sr</td>
<td>28</td>
</tr>
<tr>
<td>qd</td>
<td>7</td>
<td>ad</td>
<td>18</td>
<td>sf</td>
<td>29</td>
</tr>
<tr>
<td>ts</td>
<td>8</td>
<td>su</td>
<td>19</td>
<td>qr</td>
<td>30</td>
</tr>
<tr>
<td>td</td>
<td>9</td>
<td>om</td>
<td>20</td>
<td>qp</td>
<td>31</td>
</tr>
<tr>
<td>ta</td>
<td>10</td>
<td>sa</td>
<td>21</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\{cr, cl, qm, qa, si, dm, lm, ql\} possible new orders not yet available
(All orders are made up of one of these letter pairs, followed by an address, followed by any number of suffixes (see below) followed by a terminal character.)

NUMBERS

Four-digit decimal constants

\[
\pm n_1 n_2 n_3 n_4 \quad (0 \leq n_i \leq 9)
\]

Five-digit octal constants

\[
\pm n_1 n_2 n_3 n_4 n_5 \quad (0 \leq n_i \leq 7)
\]

(7's complement of absolute magnitude)

Positive decimal integers \( \times 2^{-15} \)

\[ p \alpha, \alpha \text{ containing any number of digits, } 0 \leq \alpha \leq 32768 \]

Negative decimal integers \( \times 2^{-15} \)

\[ n \alpha, \alpha \text{ containing any number of digits, } 0 \leq \alpha \leq 32768 \]

(All numbers are followed by any suffixes desired and by a terminal character which instruct the conversion program and do not appear explicitly in the result.)

CONVERSION CONTROL COMBINATIONS

\( i \alpha \quad \text{indicates that } \alpha \text{ is the address of the register containing the first word of the program (requires terminal character).} \)

\( j \alpha \quad \text{indicates that } \alpha \text{ is the address of the register containing the first word after an interruption in the sequence of a program (requires terminal character).} \)

\( f \alpha \quad \text{indicates that } \alpha \text{ is the address of the register containing the starting instruction of a program (requires terminal character).} \)

\( b \quad \text{indicates the end of a block of words (requires terminal character).} \)

\( 0/ \quad \text{will precede the first word of a subroutine (/ terminates).} \)

\( \alpha/ \quad \text{will precede a subroutine instruction (} \alpha \text{ the address of the subroutine relative to the beginning) (/ terminates).} \)

\( \alpha n/ \quad \text{will indicate the number of the preset parameter (} n = 0, 1, \ldots 7, x) (/ \text{ terminates).} \)
SPECIAL CHARACTERS

(special terminal character) terminates a word and indicates that word is a preset parameter.

8 or 10 (occurs only at beginning of each program) indicates that address sections of all orders and control combinations, except those preceded by 0/ or a/, are to be converted octally (8) or decimally (10).

an (an) (suffix) will follow a word requiring the addition (subtraction) of the n-th preset parameter (0 ≤ n ≤ 7).

t or d (suffix) will follow an instruction requiring the addition of the address assigned to the d register, which address is stored as the zero-th preset parameter (see Page 5).

 tx or dx (suffix) will follow an instruction requiring the addition of the address assigned to the dx register, which address is stored as the x-th (9th) preset parameter (see Page 5).

r (suffix) will follow an instruction whose address is relative to the beginning of the subroutine.

\ or \ (prefix) will be ignored by the conversion program.

\ or \ (terminal character) will terminate an instruction, number, or control combination.

Nullify, stop and space characters will always be ignored by the conversion program.

(The symbols \ and \ used here signify "space" and "carriage return").

Signed John W. Carr III

Signed John T. Gilmore Jr.

Approved Charles W. Adams

JWC: JTG/cm

Attached: DL 291 DL 293
DL 292 DL 294
**DIGITAL COMPUTER LABORATORY**
**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**
**WHIRLWIND SUBROUTINE SPECIFICATION**

**TIME:** Print Positive Decimal Integer \(< 2^{15}\)
No Page Layout, Zeros Appear as Spaces

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Subroutine</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>(1t - 3t)</td>
<td>(~100)</td>
<td>106</td>
</tr>
</tbody>
</table>

**Presets:**
None

**Program Parameters** on entering Subroutine

ul: No. to be printed

**Results** on leaving Subroutine

ac: \(+1 \times 2^{-15}\)

ul: No. to be printed

**Description:**
This subroutine prints out the contents of ul, considered as an integer. If ul contains \(n \times 2^{-15}\), \(0 \leq n \leq 32767\), then \(n\) itself is printed out in decimal form. The first zeros of an integer appear as spaces. Thus 00326 will be printed as \(000326\). Zero itself will be printed as five spaces. Only positive numbers can be printed.

This conversion and printing is accomplished by multiplying the contents of ul by \(2^{15}/10\), and using the resulting four most significant binary digits of value \(n_1\) to enter the \(n_1\)-th member of a decimal table \((0 \leq n_1 \leq 9)\). The remainder after subtraction of \(n_1 \times 2^{-4}\) is shifted left four places, multiplied by 10/16, and the process repeated to obtain \(n_2\). After 5 digits the operation ceases. This process may be written arithmetically:

\[
\begin{align*}
\nu_1 \times 10^{-4} \times 2^{15/16} &= \nu_1 \times 2^{-15/16} = \nu_2 \times 2^{-4} \left(\nu_3 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_2 < 1 \\
\nu_2 \times 10^{1/0} &= \nu_2 \times 2^{-4/0} = \nu_3 \times 2^{-4} \left(\nu_4 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_3 < 1 \\
\nu_3 \times 10^{0/0} &= \nu_3 \times 2^{-0/0} = \nu_4 \times 2^{-4} \left(\nu_5 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_4 < 1 \\
\nu_4 \times 10^{0/0} &= \nu_4 \times 2^{-0/0} = \nu_5 \times 2^{-4} \left(\nu_6 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_5 < 1 \\
\nu_5 \times 10^{0/0} &= \nu_5 \times 2^{-0/0} = \nu_6 \times 2^{-4} \left(\nu_7 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_6 < 1 \\
\nu_6 \times 10^{0/0} &= \nu_6 \times 2^{-0/0} = \nu_7 \times 2^{-4} \left(\nu_8 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_7 < 1 \\
\nu_7 \times 10^{0/0} &= \nu_7 \times 2^{-0/0} = \nu_8 \times 2^{-4} \left(\nu_9 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_8 < 1 \\
\nu_8 \times 10^{0/0} &= \nu_8 \times 2^{-0/0} = \nu_9 \times 2^{-4} \left(\nu_0 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_9 < 1 \\
\nu_9 \times 10^{0/0} &= \nu_9 \times 2^{-0/0} = \nu_{10} \times 2^{-4} \left(\nu_1 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_{10} < 1 \\
\nu_{10} \times 10^{0/0} &= \nu_{10} \times 2^{-0/0} = \nu_{11} \times 2^{-4} \left(\nu_2 \times 2^{-4}\right)_{10}/10 & 0 \leq \nu_{11} < 1
\end{align*}
\]

No alarms should occur.

JWC III
Sept. 24, 1951
TITLE: Print Positive Decimal Integer $< 2^{15}$
No Page Layout, Zeros Appear as Spaces

Abstract: This subroutine prints out the contents of ul, considered as an integer. If ul contains $n \times 2^{-15}$, $0 \leq n \leq 2^{27}$, then n itself is printed out in decimal form. The first zeros of an integer are suppressed. Thus 00326 will be printed as $0000326$. Zero itself will be printed as five spaces. Only positive numbers can be printed.

1t digit counter
2t temp. storage for number
3t suppressor

00 ta 7r Plant ul address 24 ca 35r} Eliminate
01 ta 15r} Set return 25 qa 3t} suppression
22 ao 15r} address 31 qa 2t} Multiply remainder
03 (ca 0} Set suppressor to 27 mh 35r} by 10/16
04 qa 3t} print spaces 28 sn 26r
05 ca 32r} Set counter 18 qa 29} ca 30r} Print space
06 ta 1t} 30 \{qp 8
07 (ca 0} Multiply c (ul) by 31 sn 26r
08 mh 33r} 2 10/16
09 sa 11r} Store digit to 32 p5: Digit length constant
10 ts 22r} be printed 33 \{ca 37r} Gives table entry
11 sa 15r} Store remainder 34 \{ta 0 10/16
12 ts 2t} 36 1.77776 Suppression eliminator
13 ao 1t} Count and 37 p45
14 cp 16r} check counter 38 p36
15 (sp 0} Link 39 p39
16 ca 22r} Check 40 p3
17 su 3t} \{digit for 41 p31 Decimal Digit Table
18 cp 29r} zero 42 p33
19 ca 22r} Get printer 43 p43
20 ad 34r} ready for 44 p15
21 ts 22r} printing 45 p13
22 (ca 0} Print 46 p49
23 qp 0} digit

JWC III
Sept. 24, 1951
Subroutine Test Report Form

(Please give a thorough explanation, for the benefit of the editors of the Subroutine Library, of just exactly how the submitted subroutine was tested. Include any output results (Flexowriter output, photographs, etc.) by stapling to this sheet.)

TITLE: Print Positive Decimal Integer < 15, No Page Layout, Zeros Appear as Spaces

This subroutine was tested by first printing out 0 - 100, 16,000 - 16,100, and 32,000 - 32,767. Then it was tested throughout its entire range by comparing the last digits of the number to be printed with a similar digit derived by a simple counting process (modulo 10). If the digits were not the same, the number was printed out; if the same, nothing was printed. The value 0.15066 for 2% yielded several hundred errors in the final digit; the value of 0.15067 yielded no errors, and thus gave correct values throughout the entire range.

JWC III
Sept. 24, 1961


**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE:</th>
<th>Print, Decimal Fraction, Single Column, with Sign and Point, No Roundoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Regs. in Subroutine</td>
<td>37</td>
</tr>
<tr>
<td>Temp. Regs. used by Subroutine</td>
<td>1t - 2t</td>
</tr>
<tr>
<td>Average Time (operations)</td>
<td>261.5</td>
</tr>
<tr>
<td>Max. Time (operations)</td>
<td>262</td>
</tr>
</tbody>
</table>

**Preset Parameters**

- vl (Digit length -5)

**Program Parameters**

- on entering Subroutine
  - ac: Number to be printed

**Results**

- on leaving Subroutine
  - ac: 0

**Description:**

As control is transferred to this routine, it is assumed that the number to be printed is in the accumulator. The normal procedure will be to assign vl = 0 and cause a five-digit decimal constant print-out. The sign and decimal point will be printed also. After the word is printed a carriage return will be produced so that continual use of this routine will produce a column of numbers. The last printed digit does not include roundoff of additional digits.

---

**JIC**

**Sept. 24, 1951**
Abstract: This subroutine prints the decimal equivalent of a word with sign and decimal point and then a carriage return.

Upon entering the subroutine:
ac = word to be printed
It = register used to store the remainder of the word
2t = register used to store counter
vl = (digit length counter) This value is set at 0 for a five-digit decimal constant, i.e. vl = digit length -5.

Enter 
00  ta 20r  Set return address
01  ts 1t  Store value
02  cp 21r  Is word negative?
03  ca 25r  No: print +.
22  qa 6  Printing
05  {qa 27r}  instructions
06  ca 24r  Set
t0  ts 2t  counter
17  cm 1t  /Value/ in ac
09  mh 23r  /Value/ 10 x 2^-15
10  ad 5r  Add start of
11  td 14r  Number Table
12  sl 1s  Store remainder
13  ts 1t \Pat Flexo code for
14  {ca --}  digit in ac
15  qa 0  Print digit
16  ac 2t  Have all digits been printed? No.
17  cp 8r

JTG
Sept. 24, 1951
Subroutine Test Report Form

(Please give a thorough explanation, for the benefit of the editors of the Subroutine Library, of just exactly how the submitted subroutine was tested. Include any output results (Flexowriter output, photographs, etc.) by stapling to this sheet.)

TITLE: Print, Decimal Fraction, Single Column, with Sign and Point, No Roundoff

This routine was tested by printing out a series of words whose decimal equivalents were already known. The last digit is not rounded off: i.e., +0602367321 becomes +060236.
CATEGORIES FOR SUBROUTINES
(Tentative)

AD 0.0  Algebraic Differentiation
AI 0.0  Algebraic Integration
CA 0.0  Complex Arithmetic
DE 0.0  Differential Equations
ED 0.0  Error Diagnosis
EX 0.0  Exponents
HF 0.0  Hyperbolic Functions
IN 0.0  Inverse Hyperbolic Functions
IP 0.0  Interpolation
IT 0.0  Inverse Trigonometric Functions
LG 0.0  Logarithms
MI 0.0  Miscellaneous
ND 0.0  Numerical Differentiation
NI 0.0  Numerical Integration
NR 0.0  n-th Root
OC 0.0  Output Camera
OP 0.0  Output Punch
OT 0.0  Output Typewriter
PA 0.0  Programmed Arithmetic
PM 0.0  Post Mortem
PS 0.0  Power Series
SF 0.0  Special Functions
TF 0.0  Trigonometric Functions
VM 0.0  Vectors and Matrices

Table I
# WHIRLWIND SUBROUTINE SPECIFICATION

## TITLE:

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tape</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Preset Parameters

- v1
- v2
- v3
- v4
- v5
- v6
- v7

### Program Parameters

- on entering Subroutine
  - ac:
  - br:
  - ar:
  - u1:
  - u2:
  - u3:

- on leaving Subroutine
  - ac:
  - br:
  - ar:
  - u1:
  - u2:
  - u3:

### Description:

FORM DL-291
<p>| | | | | | | | | | | | | | | | | | | | |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |</p>
<table>
<thead>
<tr>
<th>TITLE</th>
<th>NUMBER</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>66</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>69</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>71</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>73</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td>78</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td>79</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td>82</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td>83</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>85</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td>86</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td>87</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td>88</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td>89</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td>93</td>
</tr>
<tr>
<td>59</td>
<td></td>
<td>94</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>95</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td>98</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>NAME</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>138</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>139</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>149</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>151</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>154</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>169</td>
<td></td>
</tr>
</tbody>
</table>