Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

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 (This memorandum is presented for immediate use, and will be later superseded by a more thorough Engineering Note.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 EDSAC 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 set 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 catalogued 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 EDSAC notation here. A "closed" subroutine is one called in by the order <u>sp n</u>, 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 <u>sp n</u> 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 subroutine. The "Temporary Registers Used" will be the numbers (e.g., d, 1t - 5t or 1t-3t) 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 approximation 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 Flexowriter 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 <u>sp n</u>, 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 u1, 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 "v1", "v2", etc. Any program parameters contained in any of the registers of the arithmetic element, in temporary storage registers or the registers following the <u>sp</u> order which enters the subroutine (these registers are designated as "u1", "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 u1, u2, etc., should be listed. Note that in closed subroutines, the contents of the A-Register will always be the address u1 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 "s" 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:

d = address of a register of which only the address section is temporary, digital positions 0-4 being always zero.

1t = address of first regular temporary register.

2t = address of second regular temporary register.

3t = address of third regular temporary register.

nt = address of n-th regular temporary register.

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:

dx - address of a register of which only the address section is temporary, digital positions 0-4 being zero.

1tx
2tx
3tx
etc. addresses of the sections of the multiple register accumulator used by the interpretive subroutine, followed by other registers used for special purposes by the interpretive subroutines.

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 v1 - v8). Thus parameter v2 may be added to v3, for example, but not <u>vice versa</u>.

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., cp orders.

3. Horizontal solid lines should be drawn after orders where definite changes of control occur, i.e., sp orders.

4. Arrows with heads pointing right should indicate points of entry in the program from other positions, i.e., from cp and 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 $-.99999 \leq d \leq +.99999$. 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).

ORDERS

Orders	Decimal Value	Orders	Decimal Value	Orders	Decimal Value
ri	0	ck	11	ao	22
rs	1	qs	12	qf	23
rf	2	qe	13	mr	24
rb	3	cp	14	mh	25
rd	4	sp	15	dv	26
rc	5	ca	16	sl	27
qh	6	cs	17	sr	28
qd	7	ad	18	sf	29
ts	8	su	19	qr	30
td	9	cm	20	qp	31
ta	10	sa	21		

$$\left.\begin{array}{l}\text{cr}\\ \text{cl}\\ \text{qm}\\ \text{qc}\\ \text{si}\\ \text{dm}\\ \text{lm}\\ \text{ql}\end{array}\right\} \text{possible new orders not yet available}$$

(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

$+.n_1n_2n_3n_4$ $(0 \leq n_i \leq 9)$

$-.n_1n_2n_3n_4$ (negative of absolute magnitude)

Five-digit octal constants

$0.n_1n_2n_3n_4n_5$ $(0 \leq n_i \leq 7)$

$1.n_1n_2n_3n_4n_5$ (7's complement of absolute magnitude)

Positive decimal integers x 2^{-15}

$p\alpha$, α containing any number of digits, $0 \leq \alpha < 32768$

Negative decimal integers x 2^{-15}

$n\alpha$, α 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

$i\alpha$ --- indicates that α is the address of the register containing the first word of the program (requires terminal character).

$g\alpha$ --- indicates that α is the address of the register containing the first word after an interruption in the sequence of a program (requires terminal character).

$f\alpha$ --- indicates that α is the address of the register containing the starting instruction of a program (requires terminal character).

b indicates the end of a block of words (requires terminal character).

0/ will precede the first word of a subroutine (/ terminates).

α/ will precede a subroutine instruction (α the address of the subroutine relative to the beginning). (/ terminates).

vn/ will indicate the number of the preset parameter (n = 0, 1, .. 7, x). (/ 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 α/, are to be converted octally (8) or decimally (10).
an (sn)	(suffix) will follow a word requiring the addition (subtraction) of the n-th preset parameter ($0 \leq n \leq 7$).
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*
John W. Carr III

Signed *John T. Gilmore Jr.*
John T. Gilmore, Jr.

Approved *CWA.*
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

TITLE: Print Positive Decimal Integer $< 2^{15}$, No Page Layout, Zeros Appear as Spaces		LSR# OT 2.1 Classfication CLOSED	
No. of Regs. in Subroutine 46	Temp. Regs. used by Subroutine 1t - 3t	Average Time (operations) ~100	Max. Time (operations) 106

Preset Parameters
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 →→326. 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 $2^{11}/10^4$, 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:

$$n \times 2^{-15} \times 2^{11}/10^4 = n \times 2^{-4}/10^4 = n_1 \times 2^{-4} + (\nu_1 \times 2^{-4})/10^4 \qquad 0 \leq \nu_1 < 1$$

$$\nu_1/10^4 \times 10/16 = \nu_1 \times 2^{-4}/10^3 = n_2 \times 2^{-4} + (\nu_2 \times 2^{-4})/10^3 \qquad 0 \leq \nu_2 < 1$$

$$\nu_4/10^4 \times 10/16 = \nu_4 \times 2^{-4}/10^0 = n_5 \times 2^{-4} + (\nu_5 \times 2^{-4})/10^0 \qquad 0 \leq \nu_5 < 1$$

No alarms should occur.

JWC III Sept. 24, 1951

TITLE: Print Positive Decimal Integer $\leq 2^{15}$, No Page Layout, Zeros Appear as Spaces NUMBER OT 2.1

Abstract: This subroutine prints out the contents of ul, considered as an integer. If ul contains $n \times 2^{-15}$, $0 \leq n \leq 32768$, then n itself is printed out in decimal form. The first zeros of an integer are suppressed. Thus 00326 will be printed as →→326. 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
      01   ta 15r  }  Set return
      02   ao 15r  }  address
      03   (ca 0)  }  Set suppressor to
      04   qe 3t   }  print spaces
      05   cs 32r  }
      06   ts 1t   }  Set counter
      07   (ca 0)  }  Multiply c (ul) by
      08   mh 33r  }  2 11/10^4
28 -> 09   sr*11   }  Store digit to
      10   ts 22r  }  be printed
      11   sl*15   }
      12   ts 2t   }  Store remainder
      13   ao 1t   }  Count and
      14   cp 16r  }  check counter
      15   (sp 0)     Link
14 -> 16   ca 22r  }  Check
      17   su 3t   }  digit for
      18   cp 29r  }  zero
      19   ca 22r  }  Get printer
      20   ad 34r  }  ready for
      21   ts 22r  }  printing
      22   (ca 0)  }  Print
      23   qp 0    }  digit

      24   ca 36r  }  Eliminate
      25   qe 3t   }  suppression
31 -> 26   ca 2t   }  Multiply remainder
      27   mh 35r  }  by 10/16
      28   sp 9r   }
18 -> 29   ca 30r  }  Print space
      30 / qp 8    }
      31   sp 26r  }
      32   p5         Digit length constant
      33   0.15067    ≈ 2 11/10^4
      34 //ca 37r     Gives table entry
      35 //ta 0       10/16
      36   1.77776    Suppression eliminator
      37   p45     }
      38   p36     }
      39   p39     }
      40   p3      }  Decimal
      41   p31     }  Digit
      42   p33     }  Table
      43   p43     }
      44   p15     }
      45   p13     }
      46   p49     }
```

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 $< 2^{15}$, No Page Layout, Zeros Appear as Spaces — NUMBER OT 2.1

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 $2^{11}/10^{4}$ yielded several hundred errors in the final digit; the value of 0.15067 yield no errors, and thus gave correct values throughout the entire range.

JWC III
Sept. 24, 1951

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print, Decimal Fraction, Single Column, with Sign and Point, No Roundoff			LSR# OT 1.1 Classification CLOSED
No. of Regs. in Subroutine 37	Temp. Regs. used by Subroutine 1t - 2t	Average Time (operations) 261.5	Max. Time (operations) 262

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.

JTG
Sept. 24, 1951

TITLE: Print, Decimal Fraction, Single Column, with Sign and Point, No Roundoff NUMBER OT 1.1

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

1t = 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 →	04	qp ε	Printing
	05	qp 27r	instructions
	06	cs 24r	Set
	07	ts 2t	counter
17 →	08	cm 1t	/Value/ . in ac
	09	mh 23r	/Value/ . 10×2^{-15}
	10	ad 5r	Add start of
	11	td 14r	Number Table
	12	sl 15	Store remainder
	13	ts 1t	
	14	(ca --)	Put Flexo code for digit in ac
	15	qp 0	Print digit
	16	ao 2t	Have all digits been
	17	cp 8r	printed? No.

	18	ca 19r	Yes. Cause a
	19	qp 16	carriage return
	20	(sp --)	Go back to main program
2 →	21	ca 26r	Print -.
	22	sp 4r	
	23	p 10	10×2^{-15}
	24	p 4al	Initial value of counter
	25	0.07143	Number Table
	26	0.07107	
	27	0.00055	
	28	0.00044	
	29	0.00047	
	30	p 3	
	31	0.00025	
	32	0.00041	
	33	0.00053	
	34	0.00017	
	35	0.00015	
	36	0.00061	

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 NUMBER OT 1.1

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., +.602367321 becomes +.60236.

JTG
Sept. 24, 1951

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
IH 0.0	Inverse Hyperbolic Functions
IN 0.0	Interpolation
IP 0.0	Input Programs
IT 0.0	Inverse Trignometric 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	Trignometric Functions
VM 0.0	Vectors and Matrices

Table I

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE:			LSR Tape Classification
No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)	Max. Time (operations)

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

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

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

Description:

FORM DL-291

TITLE ______________________ NUMBER ______________

NAME ______________________ DATE ______________

00		15	
01		16	
02		17	
03		18	
04		19	
05		20	
06		21	
07		22	
08		23	
09		24	
10		25	
11		26	
12		27	
13		28	
14		29	

FORM DI-292

TITLE ______________________ NUMBER ______________

NAME ______________________ DATE ______________

30		65	
31		66	
32		67	
33		68	
34		69	
35		70	
36		71	
37		72	
38		73	
39		74	
40		75	
41		76	
42		77	
43		78	
44		79	
45		80	
46		81	
47		82	
48		83	
49		84	
50		85	
51		86	
52		87	
53		88	
54		89	
55		90	
56		91	
57		92	
58		93	
59		94	
60		95	
61		96	
62		97	
63		98	
64		99	

FORM DL-293

TITLE ______________________________ NUMBER ____________

NAME ______________________________ DATE ____________

100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169

FORM DI-294