Computing Note
M.S./B./200

GUIDE TO PROGRAMMING FOR DEUCE

by

Miss E. R. de Bourcier, M.A.
D. G. Burnett-Hall, B.A.

These computing notes are intended for those learning to programme for DEUCE, and should be used in conjunction with the second issue of the English Electric Company's "Programming Manual for DEUCE".

Computing Note
M.S./B./200

GUIDE TO PROGRAMMING FOR DEUCE

This guide is to be used in conjunction with the "Programming Manual for DEUCE." It is intended that the programmer should have a personal demonstration of the subjects stated when he is about to use DEUCE for the first time.

When doing these examples, it is necessary to do only sufficient examples to master the principles involved. In certain cases one example follows immediately from another; e.g. examples II, 4 and 5 or examples III, 4 and 5. Here the user should do both examples or neither. If at all possible he should get his work checked by an experienced programmer before going on to the next section.

Many of the earlier examples have been adapted from those of the Introduction to Programming for the EDSAC, 1955, of the University Mathematical Laboratory, Cambridge.

Computing Note
M.S./B./200

GUIDE TO PROGRAMMING FOR DEUCE

Manual Read chapters 1, 2, 3, and §§4.1-3 and §§4.7

The multiplier does not give the correct result unless 21_2 contains zero when the multiplier is started. The instruction $30 - 21_2$ sends zero to 21_2. When $0 - 24$ is obeyed the integers in 16 and 21_3 are multiplied together and the result is found in 21_2. The number in 21_3 is lost but that in 16 can be used again.

Examples I In these examples all numbers are small integers; in 2 - 5 they are positive. C(14) denotes the content of TS14.

1. x = C(14), y = C(16). Put x-y in 15, and 2x+y in 13.
2. x = C(14), y = C(16). Put xy in 13.
3. z = x+iy, where x = $C(19_2)$, y = $C(19_3)$. Put the real and imaginary parts of z^2 in 20_3 respectively, and put $|z|^2$ in 15.
4. a = $C(17_0)$, b = $C(17_1)$, c = $C(17_2)$ and x = C(16). Form ax^2+bx+c and place in 14. (Only two multiplications are necessary.)
5. The dimensions of a rectangular block in inches are in $18_{1,2,3}$. Put the surface area in square inches in 13, and the volume in cubic inches in 14.

Manual Read §§4.9, 4.11, 4.12.

Examples II

1. Put $-|C(15)|$ in 14.
2. Given that $C(14) \geq 0$, put the larger of C(14) and $|C(16)|$ in 14.
3. $C(17_0)$, $C(17_1)$, $C(17_2)$ are all positive. If they can represent the lengths of the sides of a plane triangle, put the largest in 14, but if they cannot, sound the buzzer and stop the machine.
4. If $C(16) < 0$ put $C(19_2)$ in 21_3, and if $C(16) > 0$, put $C(19_3)$ in 21_3; in either case reverse the sign of 16.
5. What happens in your programme for question 4 if C(16) = 0? Modify your programme if necessary to sound the buzzer in this case.
6. 18_0 contains either 10 or 3; whichever it is, replace it by the other. You may assume that any constants you need are in 19.
7. Place the numerically greater of C(14) and C(15) in 13.

Manual Read the following paragraphs- 4.4, 4.5, 4.6, 4.7, 4.8, 4.13, 6.1, and 6.2 up to the end of the first paragraph on sheet 26. Do not read through the examples on sheets 26 to 28 yet.

Examples III

1. x is in 15; place 10x in 16 and 1000x in 13, using 14.
2. Interchange C(14) and C(15) without using any other store.
3. Place the numerically greater of C(14) and C(15) in 13, using logical operations.
4. Shift up x = C(14) until $2^{30} \leq 2^n|x| < 2^{31}$ and put nP_{17} in 13, where n is the total number of shifts.
5. Shift x = C(14) till $2^{26} < 2^n|x| < 2^{27}$ and put nP_{17} in 13.
6. $x+2^{16}y$ is in 14, and the signs of x and y are the P_{14} and P_{30} digits respectively. Put x and y as full-length signed numbers in 14 and 15.

Computing Note
M.S./B./200

7. Form the number which has a 1 in any place where C(14) or C(15) or both have a 1.

8. If C(14) has a 1 wherever C(15) has a 1, remove the 1's in 14 corresponding to the 1's in 15; but if a zero in 14 corresponds to a 1 in 15, sound the alarm.

Manual — In chapter 4 read from §4.13 to the end. In chapter 6 read the examples in §6.2 (sheets 26-28) and from §6.6 to the end of the chapter. (Ignore the coding of the examples on sheets 34 and 35.)

Examples IV

All numbers are given to 30 binary places; that is, the binary point comes between the 30th and 31st digits, P_{30} and P_{31}. When you need to do a multiplication, put the two numbers in 14 and 16 and write "MULT" instead of writing the string of orders on sheet 35. Keep your answers to these questions.

1. x = C(16), a = C(15). Sum the polynomial $ax^n+ax^{n-1}+....+ax+a$ and put the result in 14. nP_{17} is given in 19_2. Use the method $\{\{(ax+a)x+a\}x+a\}x+a$ etc.
2. x = C(16), and 2^{-6} is in 15. Sum the polynomial $(x^9+2x^8+3x^7+...+10).2^{-6}$
3. x = $C(20_3)$. Place x^{13} in 16,
   (i) taking the shortest possible machine time
   (ii) using the fewest possible orders.
4. a, b, c are the contents of $18_{0,1,2}$ respectively, and a > 0, b > 0. $C(20_2)$= h > 0 and we define x_n= nh, where n is an integer. Place in 19 the value of x_n for which $y = ax_n^4 - bx_n+c$ is least, and place in 19_3 the corresponding value of y.

Manual — Read chapter 5 except §§5.12 and 5.13.

Examples V — Keep the detailed coding sheets prepared for these questions.

1. The flow diagram of the "successive squares" programme is

2_5 10 - 24
2_7 30 - 13
2_9 27 - 15
2_{11} 30 - 16
2_{13} 2_{16} - 21_2
2_0 15 - 25
2_2 13 - 15
2_4 16 - 25
2_6 13 - 16
2_8 16 - 29 X
2_{10} 27 - 23_2
2_{12} 21_2 - 28
Zero / \ Non Zero
2_{14} 9 - 24 2_{15} 2_{17} - 13
2_1 7 - 24

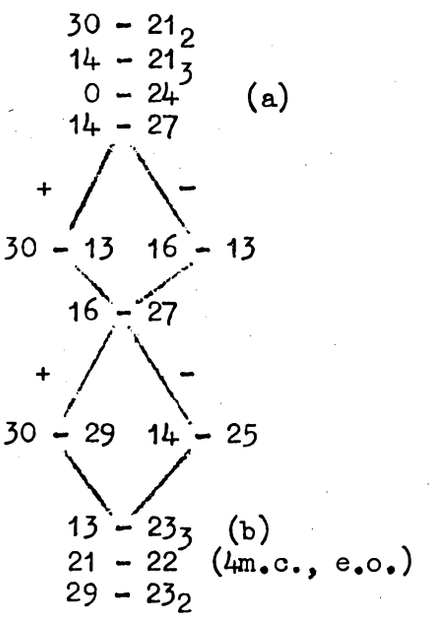
2_{16} contains $20P_1$, 2_{17} contains $2P_1$

Prepare the detailed coding sheet for this problem.

2. Do the detailed coding for the flow diagram on sheet 26. Arrange the orders in one delay line so that the loop of instructions (c) to (i) is obeyed in one major cycle, and then arrange the rest of the programme so that it is as compact as possible. Instruction (1) should have N.I.S. 1_{30}; instruction (d) leads to (e) or (j) depending on the discrim.

3. Do the detailed coding of the flow diagram on sheet 28. The instructions can be arranged in minor cycles 16 - 31 of one delay line, keeping m.c.28 clear, and the loop is obeyed in one major cycle. An order in m.c.28 leads to the first instruction, and the last instruction leads to 1_{30}.

4. The following flow diagram is for signed multiplication of numbers with 30 binary places in 14 and 16. Make the detailed coding as fast as possible, remembering that 65 minor cycles must elapse while the multiplier is running between instructions (a) and (b). The last instruction leads to 1_{30}.



Manual Read SS 5.12, 5.13.

Examples VI

In each case give the flow diagram and the detailed coding. Numbers are given to 30 b.p.

1. Given nP_{17} in 14, clear 9_n. ($n < 32$)
2. $9_0\ 9_1 \ldots 9_{31}\ 10_0 \ldots 10_{31}$ contain a series of numbers. Given nP_{17}, replace $C(9_n)$ by its modulus, where 9_n denotes 10_{n-32} if $n \geq 32$. nP_{17} is given in 14.
3. The quantity y is defined by $y = a_0 + a_1x + a_2x^2 + \ldots + a_{31}x^{31}$. x is given in 16. n is the number of terms needed to produce the required degree of precision, and nP_{17} is given in 14; $n < 32$. The coefficients $a_0, a_1 \ldots$ are given in 9_0, 9_1 .... Place y in 14.
4. Modify your programme for question 3 so that a_0 is stored in A_p, $AP_{10}+pP_{17}$ being given in 13, and the term after that in A_{31} is in A_0.
5. Place in 21_2 the sum of the moduli of the numbers in $9_0, 9_1, \ldots 9_{24}$.

6. Place in 21_3 the sum of the squares of the numbers in $9_0\ 9_1 \ldots 9_{31}\ 10_0 \ldots 10_{17}$. (It can be assumed that the sum is less than 2).
7. Place $C(9_{r+1})$ in 9_r for r=5, 6....14.
8. Test the numbers in $10_0 10_1 \ldots$, until a number x is found satisfying $3/16 \leq x < 4/16$; place x in 14. If none of the numbers from 10_0 up to 12_{31} lies in this range, sound the buzzer.
9. $9_0 \ldots 10_{31}$ contain a series of numbers. If the absolute value of any number is greater than $1/2$ divide all the numbers by 2.

Manual Read chapter 7.

Examples VII

Make flow diagrams and detailed coding for these examples, filling D.L.2 if one delay line is needed, or D.L.2 and 3 if there are more instructions. All numbers have 30 binary places. Assume that subroutines 14 (M.6), 20 (F.1), and 54 (D.4) are in delay lines A, B, C, and that all have been arranged to work with 30 binary-place numbers. You may assume that subroutine 19 (B.1) is in D.L.1 if you need it.
Keep your programmes for future use.

1. Two vectors of length n have their first components in D_p and E_q. Given $DP_{10}+pP_{17}$, $EP_{10}+qP_{17}$ and nP_{17} in suitable short stores, make a subroutine to place the scalar product of the vectors in a TS, assuming that n may be greater than 32. The thirty-second terms will be in D_{p-1} and E_{q-1}. Finally make a list of the short stores used, giving the contents at entry to the subroutine and at exit where they are of interest.

2. A scalar c is given in 16, and a vector of length n (which may be greater than 32) starts in D_p. Make a subroutine to multiply each term of the vector by the scalar, thus replacing a_r (the content of D_{p+r}) by ca_r.
   As a check, accumulate the sum of the products ca_r double length (not rounded off), also the exact sum of the terms a_r, ignoring any overflow which may occur because the sum is greater than 2. At the end of the programme form the product $c.\Sigma a_r$, not rounded off. Then if the less significant half is not the same as the less significant half of Σca_r sound the buzzer and stop the machine.
3. A three dimensional vector is given in $17_{0,1,2}$. Form a subroutine to put the modulus of the vector in 18_3 and the normalised vector in $18_{0,1,2}$. Finally make a list of the short stores used with contents at entry and at exit.

Manual Read S6.3.

Examples VIII

Make flow diagrams and detailed coding for D.L.2(3,4...).
Keep the results for future use.

1. A vector of length n (maybe greater than 32) is punched in binary, one element per row, on a succession of cards.

On a card placed in front of the first card of the vector nP_{17} is punched on the X row and sP_{17} on the 1 row; in the row immediately following the last element of the vector is punched the sum of the elements formed double length and shifted down s places without round off. Make a subroutine to read the vector into A_p A_{p+1}...., where $AP_{10}+pP_{17}$ is given in 13, and stop the reader. If the sum is not correct, sound the buzzer.

2. Modify your programme for the previous question to find, in addition, the largest absolute value of the components of the vector, and place this in a TS. (This can also be done while reading cards).

Manual — Read §§ 8.1, 8.3, 8.4. In reading § 8.3 it should be noted that all the timing numbers are now 28, and the Y row has 1, 1-1, 0, 28X punched on it. Make flow diagrams of the sequences of orders obeyed by the machine

(i) in reading into one of D.L. 1 - 8;
(ii) in reading into one of D.L. 9 - 12;
(iii) in reading the initial card, in this case assuming that the top row is read in minor cycle m. (m may be odd or even.)

Demonstration

Use of card punch. Colours of cards.

Examples IX

1. Make a list of the binary integers from 0 to 31 and give their binary equivalents with the least significant digit first.
2. Using your solutions to examples VII (1) and VIII (1) make a programme to read in two vectors and punch out their scalar product in binary. The punching of the vectors is as specified in example VIII (1). Check that the vectors are of the same length. Punch up your programme.

Demonstration

Conversion from decimal to binary (integers and fractions) on a Brunsviga.
Use of check detailed coding on DEUCE.

Example X

Prepare a test case for your programme for scalar products. Choose two vectors (fairly short) and calculate their scalar product on a desk machine. Find the binary equivalents of all the terms of the two vectors and of the product. Punch up the vectors in the way specified in example VIII (1).

Demonstration

Use of Hollerith equipment.
Programme testing on DEUCE: use of console; programme display.

Read the description of the console of DEUCE in Computing Note 201 and the hints on programme testing in Computing Note 202. In the latter case the intention is to give you a general idea of methods used rather than a detailed knowledge.

Destination Triggers

The "destination triggers" instructions are:-

0-24	Stim. Mult.	6-24	Alarm off.
1-24	Stim. Div.	7-24	Alarm on.
2-24	TIL Discrim.	8-24	Clear output staticiser.
3-24	TCA on.	9-24	Clear reader and punch.
4-24	TCB off.	10-24	Stim. punch.
5-24	TCB on.	12-24	Stim. reader.

8-24

The output staticiser can be cleared by pressing the key on the console, by this instruction, or by running the punch. In the last case the word on the O.P.S. is punched on the first row. The O.P.S. cannot be cleared by 30-29, since, if two numbers are sent to D.29 when this is the output staticiser, they will be superimposed. The second transfer does not clear the number left by the previous transfer.

6-24

This has exactly the same effect as pressing the key on the console.

3-24

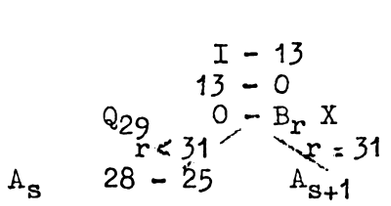
Read paragraph 6.5 of the manual.

DEUCE News

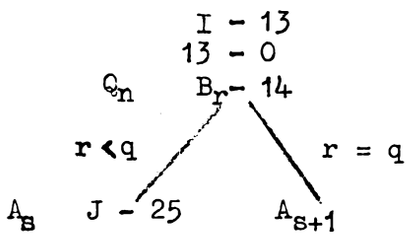
This publication contains news of the present state of the subroutine and programme libraries, work on hand at the various establishments using DEUCEs, and useful programming devices. We do not recommend the use of "Programme Surgery" in the August 1955 issue (see Computing Note 201) for the procedure for testing the continuity of a programme as suggested by N.P.L. in the October 1955 issue. In this case it is much better to use "Check detailed coding."
One of the programming devices is the order 21_2-23_2 (TCB off). This clears the lower half of 21 and rounds off the top half at the same time. Another and more important subject is the use of "spill-over".

Spill-over

The method of spill-over is to modify an instruction in a loop, e.g. by adding P_{17} each time round the loop, so that at some time a spill-over into the timing number occurs. The variable instruction then selects a next instruction one minor cycle later (or earlier) than usual. The following example fills delay line B from minor cycle 0 to m.c.31.



I is A, 0-B, 0, 1, (15), s+1 X, where the (15) refers to the four spare digits in the instruction word between the wait and timing numbers. On the first time round the loop the variable instruction is 0 - B_0 X. For 0 - B_{30} X the instruction is A, 0-B, 0, 31, (15) s+1 X. When another P_{17} is added it becomes A, 0 - B, 0, 0, (0) s+2 X and the next instruction has become A_{s+1}.
It is essential to this example that spill-over occurs when the instruction is 0-B_{31} X. The wait number must have increased to 32 (i.e. 0, with a spill-over of 1 into the spare digits) this time, so the variable instruction must be a quasi-29 instruction. This is the limitation of the method; it can only be used when the programmer knows in advance where the last number will be stored. (Normally this means he must know how many numbers are being stored.) If it is required to fetch numbers from minor cycles p to q of delay line B and q - p $\leq$ 16 another method can be used.



where J is P_{17} + P_{22} and I is A, B-14, 0, p-n-2, (16+p-q), s-n-2. B_r is fetched when the instruction has had P_{17} + P_{22} added (r-p) times and is A, B-14, 0, r-n-2, (16+r-q), s-n-2. So when r = q the instruction is A, B-14, 0, q-n-2, (0) s-n-1 and spill-over has occurred. In this case it has not mattered which minor cycle the variable instruction is obeyed in.
A variant of this method is often used in the magnetic transfer instructions in the programmes described in Computing Note 203. The modifier here is P_5 + P_{22}.

Magnetics

Read paragraphs 3.3 and 6.4. Ignore paragraph 8.2 since this has been superseded by later methods. Computing Note 203 gives details of the magnetics programmes now available.