Example Consider a program which requires the use of the following general purpose routines:

(a) A card read subroutine which stores the card image (i.e. successive sets of 6 columns in consecutive words of I.A.S.) under R.R.N.3.

(b) A distribution routine which distributes a card image held under R.R.N.3 into a form suitable for processing, the distributed information being stored under R.R.N.4.

(c) A print routine which prints information held under R.R.N.3 and takes the current sprag number from word 0 block 4.

Confusion can be avoided if the user allocates his own block numbers for the various sets of information. Suppose that the program is written and relativizer control words set at the head of the program assuming the following:

(a) The information from cards is stored as a card image in R.R.N.10.

(b) This information is distributed from R.R.N.10 into R.R.N.11 ready for processing.

(c) The information is processed in R.R.N.11 and is distributed into R.R.N.12 (commencing at word 1) ready for printing.

(d) The current sprag number is placed in word 0 block 12. The print routine prints the results from words 1, 2......block 12 taking the sprag number from word 0 block 12.

In order that the general purpose routines should operate on the correct data it is necessary to include relativizer control words immediately before the cards for these routines.

Thus, the control word

D	F	A	R
R	---	0000	10
		0000	3

should be included in the program pack immediately before the cards for the card read subroutine, the control words

D	F	A	R
R	---	0000	11
		0000	3
R	---	0000	12
		0000	4

should be included in the program pack immediately before the cards for the distribution routine, and the control words

D	F	A	R
R		0001	12
		0001	3
R		0000	12
		0000	4

should be included in the program pack immediately before the cards for the print subroutine.

Relativizer cards included in this way must, of course, be numbered so that they pass the Initial Orders sequence check. Subroutines available from the I.C.T. Library have relativizer control words included at their head for all relativizers used by the routine. These relativizer cards have card numbers already punched but the relativizer setting columns are left blank for the user to include his own settings. The I.A.S. and drum address columns of the block relativizer word(s) should also be punched by the user.

To avoid possible errors, general purpose routines should be included in the program pack after the other program blocks. Hence, in the example given, if block 4 were being used by the main program, relativizer 4 should not be set to the same addresses as relativizer 10 until all main program references to R.R.N.4 have been read and converted to absolute form.

All library routines use R.R.N.1 as a temporary storage area. When using several such routines, therefore, it is sufficient to allow space under R.R.N.1 for the largest temporary storage requirement of the routines being used. Care should be taken, however, that intermediate results are not stored in R.R.N.1 by the main program in words where they will be destroyed by one of the subroutines.

RESTART PROCEDURES 4.8

When a program makes a test and discovers that an error condition has arisen, the computer should be stopped with an identifying number in the address part of the stop instruction which is displayed in CR3. By means of this information the operator can diagnose the type of error condition which has arisen, and take the necessary action to restart the program.

It should normally be possible to continue with the program once the necessary error procedure has been carried out. For example, if an error is found in the data it might be possible, when the computer is restarted, to enter an error routine which prints out a statement that the set of data is not correct. When this has been done the program can continue to read and process the next set of data.

There are some errors which may arise after which it is not possible to continue with the program because information recorded in the machine may be incorrect. Before continuing the processing, therefore, it is necessary to return to an earlier stage in the program and re-create this information. For a small program this would mean completely re-running the program. For programs which run for a long time, however, it is necessary to have restart points to which a return can be made if necessary. Restart points should be included at least once every half hour in the program running time. The program must be written so that when a return is made to the previous restart point any accumulations are set to their original value at that point in the program.

Restarts following I.A.S. and Drum Parity Error Stops 4.8.1

If an I.A.S. parity error occurs, a return should always be made to the previous restart point. When a word is transferred from I.A.S. to Register A an automatic parity check is carried out and indicator 06 is set if an error is detected. Parity bits are then regenerated and the word is stored back in its original location in I.A.S. This means that, if a further attempt is made to effect the transfer from I.A.S., the parity will be correct although the word itself is incorrect. To avoid errors, therefore, a return should be made to the previous restart point.

There are standard routines available for testing drum parity and it is recommended that these should be used. By retaining such routines permanently in I.A.S. and using them for all drum transfers certain programming difficulties can be overcome. The drum transfer is supplied to the routine as a parameter and it is obeyed as part of the subroutine. This makes it possible for a drum transfer instruction to cause itself to be overwritten, the instruction being preserved in the subroutine if it becomes necessary to repeat the transfer.

If a drum parity error occurs then the computer is brought to a stop; restart causes another transfer attempt to be made. When a drum parity error is detected this implies that a transfer error has occurred either in the current transfer or when the information was previously recorded on the drum. In the former case, unless there is a serious mechanical fault, the error should be corrected when another attempt is made. If the error is persistent then, either there is a mechanical fault or the transfer error occurred on writing to the drum. In either case a return should be made to the previous restart point.

PROGRAM TESTING 4.9

When a program has been written and thoroughly checked it is necessary to test it by running it on the computer. In order to test a program it is necessary to create (or obtain) sample data. In addition, particularly when testing a subroutine or a section of a large program, it may be necessary to write a special test program to act as a control routine for the program under test. The control routine would, for example, position the input data correctly and ensure that the programmed indicators were in the required states. The test data and programs should be such that every possible sequence of executing the instructions is tested including the error procedures.

For a large program it is advisable to carry out testing in stages. This enables errors to be more quickly located since they are confined to certain sections of the calculation. The testing of a large program can be achieved by one of the following methods:

(a) Divide the program into sections and test each section individually. When the sections have all been tested combine them and re-test as a complete program.

(b) Divide the program into sections. Test the first section and then combine it with the second section. Test these sections and then combine with the third section. Continue until the complete program has been assembled and tested.

(c) Assemble the whole program initially but start by using data which test only part of the program. Gradually include in the tests sufficient data to cover all conditions.

When a program is ready to be tested it should be punched into program cards. The punched cards should be interpreted so that they can be quickly checked for punching errors. The pack should then be run with the Validity Check subroutine to detect invalid instructions. Data cards should also be punched and checked and placed at the end of the program pack. The assembled pack should be accompanied by an operators' instruction sheet and sent to the machine room. If any print -outs are required of the I.A.S., drum or magnetic-tape storage, these should be specified on the operators' instruction sheet. The program pack will be returned with an indication of whether or not the final stop was reached. If the final stop was not reached then a console log is returned with the pack which contains details of the error and the contents of the registers and states of indicators as displayed on the console when the computer stopped. The states of the Mill, overflow and program indicators and the contents of Registers B and C are not included on the console log since they are given at the head of the I.A.S. or drum print-out.

The information provided on the console log should, together with the print-outs of storage, normally be sufficient for the programmer to locate his error. When one error has been found it is a good practice to examine the print-outs to determine whether, discounting that error, the program has operated correctly. Machine time can be saved and the program proved more quickly if several errors can be corrected for one run on the machine. There are several general purpose routines available which can be used as aids to program testing. These are described in Part 5 under Diagnostic Routines.

During program testing it is very often necessary to alter or re-write sections of the program. It should be noted that if a block is lengthened some relativizer settings may need to be altered. Any alterations should be recorded on the program sheets so that the latter provide an up-to-date copy of the program. Care should be taken during re-writing to ensure that altering the positions of instructions does not lead to further errors. Particular care should be taken where:

(a) There is a jump to a program word from another part of the program.

(b) Reference is made to a word by an instruction in another block; for example where a constant held in one block is used by another.

(c) Where instructions are used whose effect depends on their positions in the first or second half of a word. This is particularly true of a 41 instruction.

TIMING A PROGRAM

It is often necessary to calculate the exact time which a piece of program is going to take when it is run on the computer. This is particularly necessary for a general purpose routine or a piece of program which is to be time-shared with the card reader or card punch. It is evident that such timings cannot be observed on the computer, since the times involved are much too small to be accurately measured. The times must therefore be estimated by a summation of the times involved in the execution of the instructions which constitute the program, together with the time for the change of control between program words.

The execution time for a section of program consisting of a straightforward sequence of instructions can be easily assessed as follows:

(a) Sum the instruction times for the instructions which constitute the program. Allow 12 micro-seconds for an unsuccessful indicator test but add nothing for a successful indicator test.

(b) Add 12n microseconds where n is the number of words constituting the second of program. This in effect allows 12 microseconds for each control change, whether it is programmed or occurs automatically between consecutive program words.

Example

I	D	F	A	R
11		45	0000	3
		3	0000	5
12		37	0000	5
		62	0001	5
13		57	0001	
		62	0002	5
14		42	0000	1
	4	18	0016	B
15		61	0000	1
		42	0000	1

Suppose first that indicator 18 is set. The time can be assessed as follows:

45 instruction (3 words transferred)	78 µs
37 instruction	21 µs
62 instruction	21 µs
57 instruction	17 µs
62 instruction	21 µs
42 instruction	21 µs
Successful indicator test	-
12 x number of words obeyed	48 µs
TOTAL TIME	227 µs

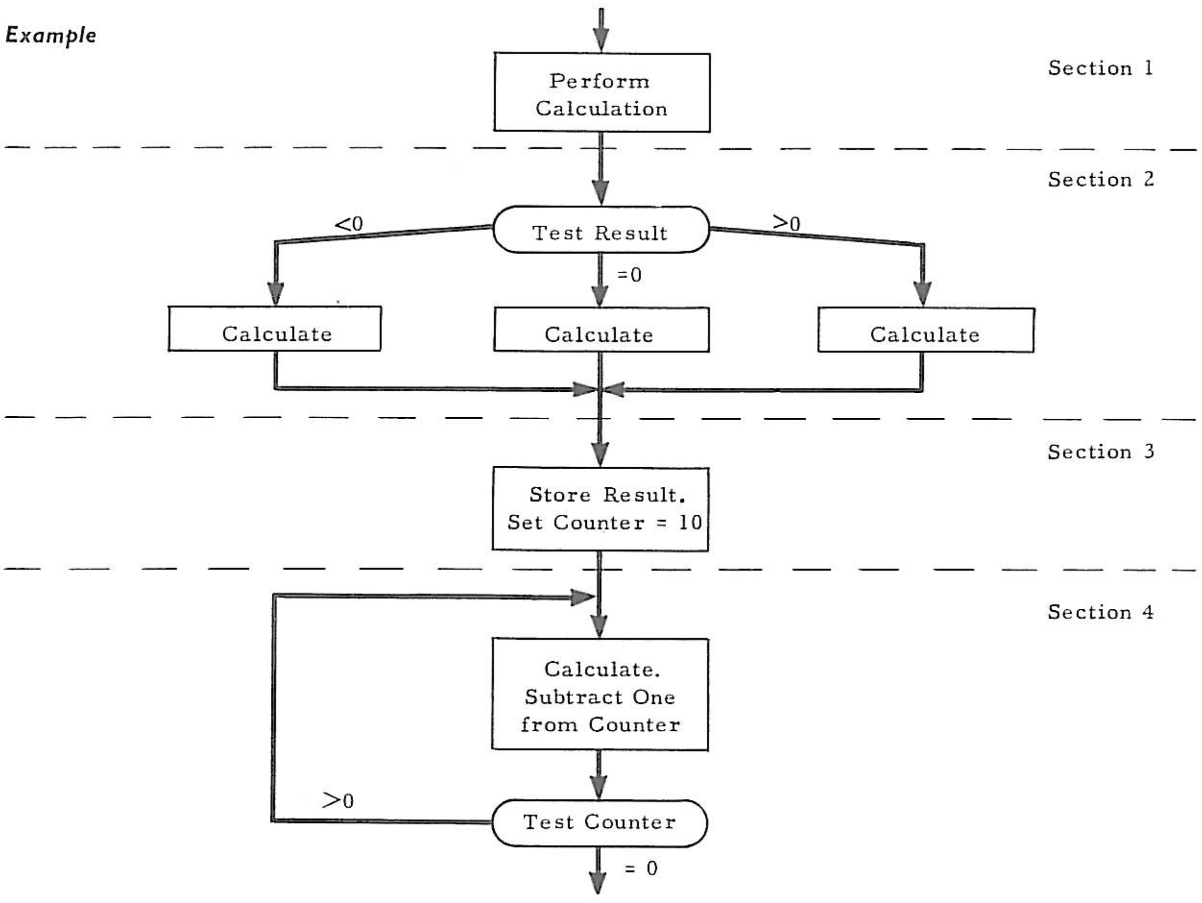
If indicator 18 is unset the times are as follows:

45 instruction (3 words transferred)	78 μs
37 instruction	21 μs
62 instruction	21 μs
57 instruction	17 μs
62 instruction	21 μs
42 instruction	21 μs
Unsuccessful indicator test	12 μs
61 instruction	21 μs
42 instruction	21 μs
12 × number of words obeyed	60 μs
TOTAL TIME	293 μs

It should be noted that these times include the control jump to the next word of program. When timing a subroutine the link should be counted as one of the words constituting the program.

Timing becomes slightly more difficult when the program branches into several alternative paths or when a section of program forms a loop and is obeyed several times. The best method is to use the flowchart as a guide and to divide the flowchart into sections for timing purposes.

Example



Suppose that it is known that the entire calculation is to be performed 18 times and that of these, eight correspond to the case <0, seven to the case = 0 and three to the case >0 in section two. The complete time can then be assessed as follows:

(a) The time for the program corresponding to section 1 can be assessed. This is multiplied by 18.

(b) The time for each of the three branches can be calculated. The 12 microseconds for an unsuccessful test instruction should be included where appropriate. The times for the three branches are multiplied by 8, 7 and 3 respectively.

(c) The time for the program corresponding to section 3 is assessed. This is multiplied by 18.

(d) The time for the program corresponding to section 4 is assessed and is multiplied by 10 because the loop is obeyed 10 times. This time is then multiplied by 18.

(e) All the times are summed.

It is not always possible to give an exact time for a piece of program. The program paths followed may depend on the data values and the calculation may contain instructions such as multiplication or drum transfers which do not take a fixed time. In such cases average and/or maximum times should be estimated. For a general purpose routine the average and maximum timings should be quoted. When a program is to be time-shared with a print or punch program the maximum timings should be calculated since it is necessary that in all cases the program should be within the permitted time available.

For a general purpose routine, it is often helpful to express the timing as a formula when exact times cannot be given. The variables in the formula may be dependent on the data or on parameters which are given to the routine. For example, the timing for a division routine might depend on the number of decimal places required in the result. This might be given to the routine as a parameter. A typical example of timings given by formulae are those for the sorting routines. These are expressed in terms of two variables, the number of words in a record and the number of records to be sorted.

When timing a large program it is not normally necessary to consider the time for each instruction. The operational speed is normally governed by the peripheral units and a time assessment can be made by considering the speed at which these will be working. Alternatively a time can be obtained by timing the program on the computer for a small amount of data.

DOCUMENTATION 4.11

It is essential that, once a program has been written and tested, it is preserved not merely as a pack of cards but as a completely documented piece of work. The documentation should be such that it contains complete instructions for using the routine. Sufficient information should be included for a different programmer to be able to understand the program if it becomes necessary to amend it at a later date. The documentation should include the following details where appropriate:

Specification - A short description giving details of what the routine does and how to use it. In the case of a general purpose subroutine, this should conform to the standard layout.

Description - Any further description not included in the specification.

Flowcharts - Complete flowcharts using standard symbols. For large jobs these should be cross -referenced with the program sheets.

Input and Output Layouts - Planning charts showing the input and output card layouts and the printed output layout.

Storage Charts - Charts showing the I.A.S. and drum storage allocation.

Program Sheets - Program sheets for the entire program complete with narrative. Program sheets for the test program should also be included.

Program Cards - Complete pack of program cards correctly numbered for Initial Orders sequence check.

Test Data and Test Program - Cards for test program and data.

Operating Instructions - Complete instructions including action on error stops.

Sample Print- and Punch-outs - Sample results for the program when run with the test program.

AN EXAMPLE OF A CODED PROGRAM 4.12

Introduction 4.12.1

The accompanying program is an example of a weekly P.A.Y.E. calculation and should be regarded purely as a specimen piece of program and not as the standard method of P.A.Y.E. calculation which will normally be used. For the sake of simplicity, no subroutines or modification techniques have been used; therefore, this program requires considerably more than the minimum possible number of instructions. The program is only intended to demonstrate the use of the functions and indicators described earlier in this manual.

The following is a summary of the calculations involved.

Factors	**Maximum Value**	**4.12.2**
Week No.	52	
Week 1 designation	(1 = Week 1 case; 0 = normal)	
Number of holiday weeks	9	
Week 1 Free Pay	£ 99. 19. 0.	
Gross Wage Brought Forward	£9,999. 19. 11.	
Gross Wage this week	£ 99. 19. 11.	
Tax brought forward	£ 999. 19. 0.	
Total Deductions	£ 99. 19. 11.	

Results	**Maximum Value**	**4.12.3**
Gross Wage carried forward	£9,999. 19. 11	
Tax carried forward	£ 999. 19. 0	
Tax this week	£ 99. 19. 0	
Tax Deduction/Refund designation	(1 = Refund, 0 = Deduction)	
Net Wage	£ 99. 19. 11.	

Calculations 4.12.4

(a) Gross Wage + Gross brought forward = Gross carried forward.

(b) Calculation of Tax carried forward as follows:-

If Week 1 case:- 1 + number of holiday weeks = 'Tax Week Number'.

If not Week 1 case:- Week No. + number of holiday weeks = 'Tax Week No.' Set Indicator 10.

Week 1 Free Pay x 'Tax Week Number' = Tax Free pay to date.

If Week 1 case:- Gross Wage - Tax free pay to date = Net Taxable Income.

If not Week 1 case:- Gross carried forward - Tax free pay to date = Net Taxable Income.

From Net Taxable Income calculate Tax to date carried forward (for explanation of P.A.Y.E. calculation, see section 4.12.5).

(c) Tax carried forward - Tax brought forward = Tax this week (Deducted or Refund).

(d) Gross Wage ± Tax this week - Total Deduction = Net Wage.

P.A.Y.E. Calculation 4.12.5

The method of deriving Tax carried forward following the calculation of the Net Taxable Income is explained in detail below. For the sake of simplicity the 'Tax Week Number' of Section 4.12.4 will be referred to as week number, and in all references to tax it should be noted that this will actually be 'tax this week' in the case of employees being taxed on a Week 1 basis, and 'tax carried forward' in all other cases. Net Taxable Income is abbreviated to N.T.I.

Certain assumptions are made regarding Earned Income Allowance, rates of tax etc., as follows:-

Earned Income Allowance is assumed to be at the rate of $\frac{2}{9}$ths of the N.T.I., and in the specimen program it is assumed that no employee has a N.T.I. of over £4,005 so that it is unnecessary to allow for the reduction of the Earned Income Allowance to $\frac{1}{9}$th of the N.T.I. when the latter exceeds £4,005.

Over the whole year, it is assumed that, after deduction of Earned Income Allowance, the first £60 of the N.T.I. is taxed at 1/9d in the £, the next £150 at 4/3d in the £, the next £150 at 6/3d in the £, and the remainder at 7/9d in the £. At any given week, the £60 and the two £150 ranges will be the proportionate amounts up to and including that week, so that, at any week x, the ranges will be:-

$$\frac{£60x}{52} \quad ; \quad \frac{£150x}{52} \quad ; \quad \frac{£150x}{52}$$

The following is a summary of the sequence of operations involved in the P.A.Y.E. calculation in the specimen program:-

(a) If the N.T.I. is negative or zero, the computer is programmed to ignore the whole of the tax calculation so that the result will be zero.

If the N.T.I. is positive, then:-

(b) Tax payable at 1/9d in the £ is calculated by multiplying the whole of the N.T.I. by 0.06806, which is derived as follows:-

$$\frac{1\frac{3}{4}}{20} \times \frac{7}{9} = 0.06806 \text{ approximately.}$$

Thus the $\frac{2}{9}$ths Earned Income Allowance is automatically taken into account.

(c) The £60 range factor is deducted from the N.T.I. This is calculated as follows: at Week 1 the amount to be deducted will be

$$\frac{£60}{52} \times \frac{9}{7} = £1\text{-}9\text{-}8.044 \text{ approximately.}$$

The reason for increasing the proportionate £60 range, i.e. $\frac{£60}{52}$ by $\frac{9}{7}$ths is the fact that $\frac{2}{9}$ths Earned Income Allowance has not been deducted from the N.T.I. and therefore the factor deducted must be increased proportionately while the tax-rate decimal (e.g. 0.06806 in (b) above) is correspondingly reduced by multiplying by $\frac{7}{9}$ Therefore, at Week x, the factor to be deducted from the N.T.I. is found by multiplying £1-9-8.044 by the Week Number x.

(d) Following subtraction of the appropriate £60 range factor, the computer tests whether or not the result is negative; if it is negative, no tax is due at the higher rates and there is an automatic jump to the end of the tax calculations. If the result is positive (or zero), the remainder of the N.T.I. is taxed at a further 2/6d in the £ by multiplying by .09722, which is derived as follows:-

$$\frac{2\frac{1}{2}}{20} \times \frac{7}{9} = 0.09722 \text{ approximately.}$$

Again the $\frac{2}{9}$ths Earned Income Allowance is automatically allowed for, and the resulting amount of tax is added on to the tax calculated in (b) above.

(e) The first £150 range factor is deducted from the N.T.I. This is calculated in a similar manner to the £60 range factor in (c) above, the amount at Week 1 being:-

$$\frac{£150}{52} \times \frac{9}{7} = £3\text{-}14\text{-}2.11 \text{ approximately.}$$

This will be multiplied by the Week Number to give the appropriate factor to be deducted at the given week.

(f) Following subtraction of the first £150 range factor from the N.T.I. the computer goes into the same routine as that described in (d) at a further 2/- in the £ by multiplying by 0.07778, which is derived as follows:-

$$\frac{2}{20} \times \frac{7}{9} = 0.07778 \text{ approximately.}$$

(g) The second £150 range factor is then deducted from the N.T.I., the amount being the same as in (e) on the previous page.

(h) Following subtraction of the second £150 range factor from the N.T.I. the computer goes into the same routine as in (d) above, except that the remainder of the N.T.I. if any, is now taxed at a further 1/6d in the £ by multiplying by 0.05833, which is derived as follows:-

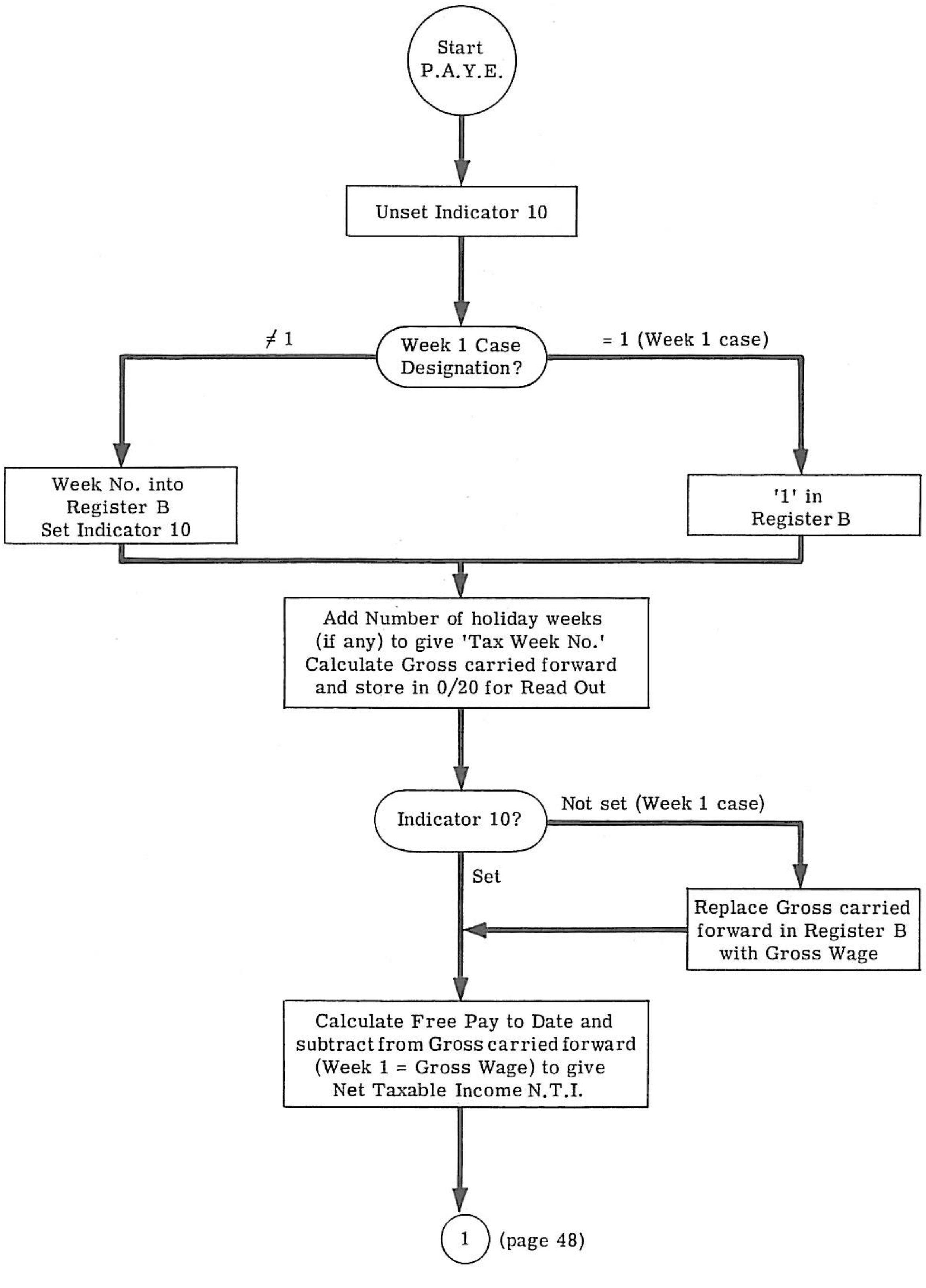
$$\frac{1\frac{1}{2}}{20} \times \frac{7}{9} = 0.05833 \text{ approximately.}$$

In effect, therefore, the remainder of the N.T.I., if any, has been taxed at 7/9d in the £, being taxed at 1/9d in the £ in (b) above; a further 2/6d in the £ in (d); a further 2/- in (f), and a further 1/6d in (h).

(i) The application to the N.T.I. of the various tax rates has now been completed, the resulting amount of tax representing the Tax to date carried forward in normal cases, i.e. those being taxed on an accumulative basis, and Tax this week in the case of employees being taxed on a Week 1 basis. Therefore, the computer now tests whether or not indicator 10 is set; if it is unset, indicating a Week 1 case, Tax brought forward is added to Tax this week to give Tax carried forward for all cases, Week 1 or normal, and Tax this week can then be calculated (see Section 4.12.4).

Timing 4.12.6

The total time required for the program will be very variable, depending on the size of the Net Taxable Income, the minimum being the case where the Net Taxable Income is zero or negative, in which case the computer skips over instructions 11-37 inclusive (see Figure 46), and the time required will be about 1 millisecond. On the other hand, when the Net Taxable Income is sufficiently large for tax to be paid at the highest rate, i.e. 7/9d in the £, the time required will be about 6 milliseconds.



Start P.A.Y.E.
Unset Indicator 10
Week 1 Case Designation?
≠ 1
= 1 (Week 1 case)
Week No. into Register B Set Indicator 10
'1' in Register B
Add Number of holiday weeks (if any) to give 'Tax Week No.' Calculate Gross carried forward and store in 0/20 for Read Out
Indicator 10?
Not set (Week 1 case)
Set
Replace Gross carried forward in Register B with Gross Wage
Calculate Free Pay to Date and subtract from Gross carried forward (Week 1 = Gross Wage) to give Net Taxable Income N.T.I.
1
(page 48)

Figure 45: **FLOWCHART OF P.A.Y.E. PROGRAM**

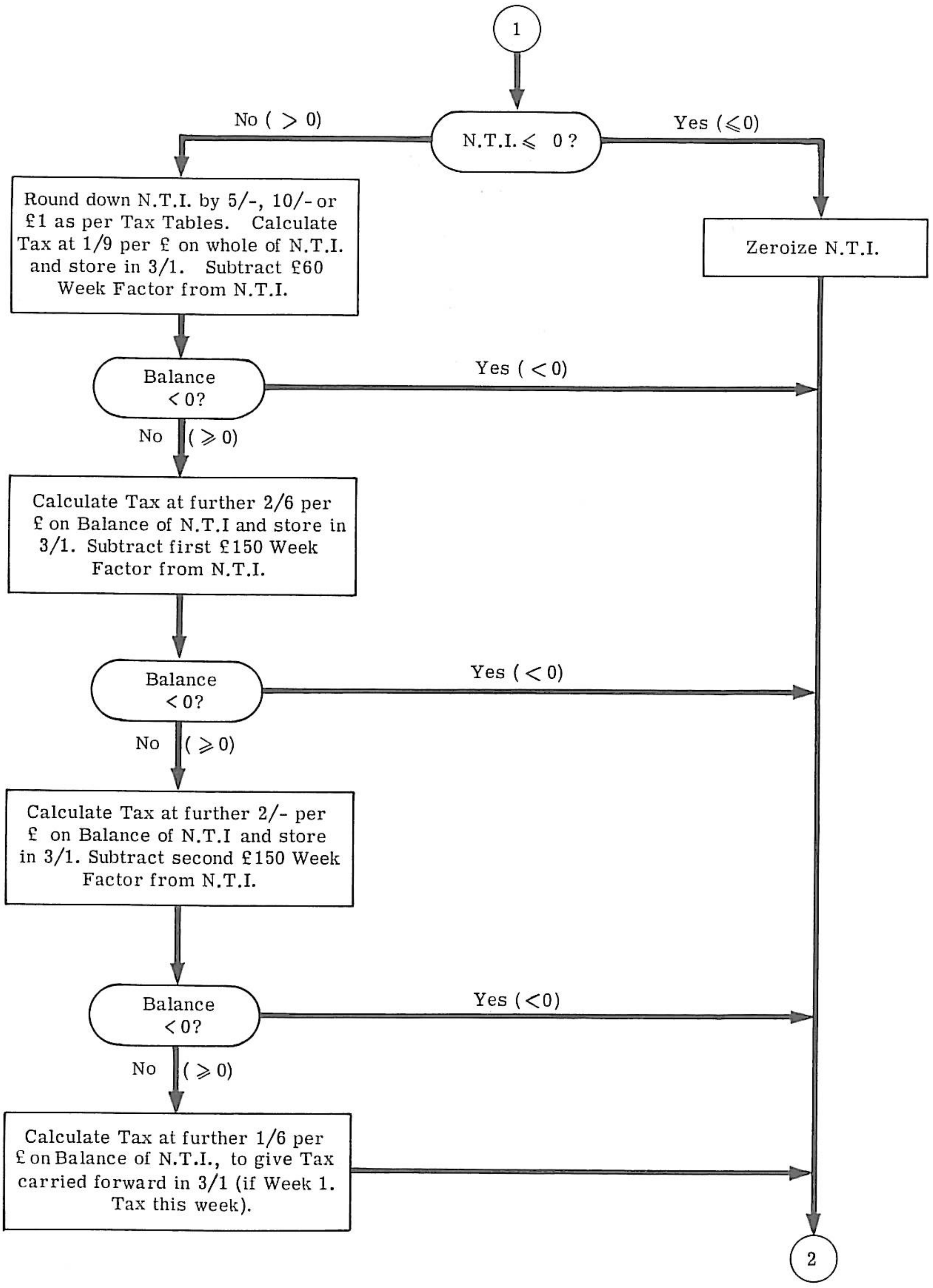

1
N.T.I. ⩽ 0 ?
No (> 0)
Yes (⩽0)
Round down N.T.I. by 5/-, 10/- or £1 as per Tax Tables. Calculate Tax at 1/9 per £ on whole of N.T.I. and store in 3/1. Subtract £60 Week Factor from N.T.I.
Zeroize N.T.I.
Balance < 0?
Yes (< 0)
No (⩾ 0)
Calculate Tax at further 2/6 per £ on Balance of N.T.I and store in 3/1. Subtract first £150 Week Factor from N.T.I.
Balance < 0?
Yes (< 0)
No (⩾ 0)
Calculate Tax at further 2/- per £ on Balance of N.T.I and store in 3/1. Subtract second £150 Week Factor from N.T.I.
Balance < 0?
Yes (< 0)
No (⩾ 0)
Calculate Tax at further 1/6 per £ on Balance of N.T.I., to give Tax carried forward in 3/1 (if Week 1. Tax this week).
2

Figure 45: continued

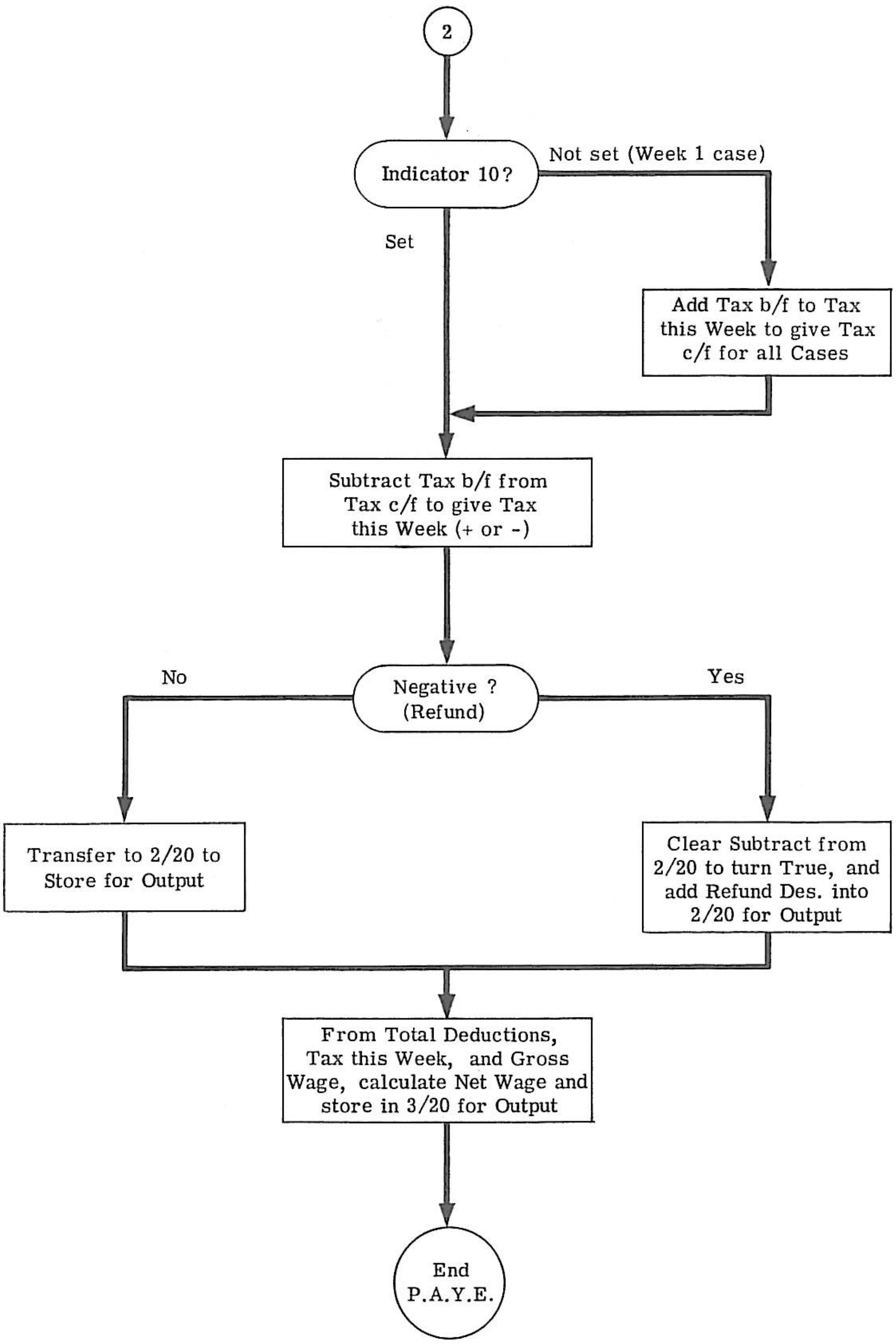

2
Indicator 10?
Not set (Week 1 case)
Set
Add Tax b/f to Tax this Week to give Tax c/f for all Cases
Subtract Tax b/f from Tax c/f to give Tax this Week (+ or -)
Negative ? (Refund)
No
Yes
Transfer to 2/20 to Store for Output
Clear Subtract from 2/20 to turn True, and add Refund Des. into 2/20 for Output
From Total Deductions, Tax this Week, and Gross Wage, calculate Net Wage and store in 3/20 for Output
End P.A.Y.E.

Figure 45: continued

I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET	JOB P.A.Y.E. (Weekly)	BLOCK No. 10
		SHEET No 1/4
	PROGRAMMER:-	/ /

C	I	D	F	A	R		NARRATIVE
1		B					
2	0	9	10	0	—		Ensure that indicator 10 is unset.
			60	1	18		Test if Wk. 1des. (0 or 1 in Register B)
	1	4	02	3	B		If Week 1 case jump to word 3
			60	0	18		If not Wk. 1 case basic Week No. into Reg. B
	2	8	10	0	–		If not Week 1 case set indicator 10
		4	00	3	B		Jump to word 3
3	3		62	2	18		Add number of Holiday Weeks to given 'Tax Week'
			42	0	1		Store on 0/1
	4		21	0	–		Set Decimal Point Register = 0
			22	7	–		Set Sterling Register = 7
	5		79	3	18		Week No. x Week 1 Free Pay = Tax free pay to date
			42	1	1		Store on 1/1
4	6		37	4	18		Gross brought forward into Register B
			72	5	18		Add Gross this week to give Gross c/f
	7		42	0	20		Store on 0/20 for output
		4	10	9	B		If not Week 1 case jump to word 9
	8		37	5	18		If Week 1 case replace Gross c/f by Gross Wage
		4	00	9	B		Jump to word 9
5	9		73	1	1		Subtract Free Pay to Date to give Net Taxable Income (N.T.)
		4	02	11	B		If N.T. > 0 jump to word 11
	10		57	12	–		If N.T. ≤ 0 zeroize
		4	00	38	B		If N.T. ≤ 0 jump to word 38 skipping tax calculations
	11		42	2	1		Store N.T. (unrounded) on 2/1
			35	2	13		Mask to extract unit shillings and Pence
6	12		73	0	13		Subtract 5/- from unit shillings and Pence
			37	2	1		Net taxable into Register B
	13		57	5	–	}	Eliminate unit shillings
			54	5	–		and pence
	14	4	03	15	B		If unit shillings and pence < 5/- jump to word 15
			72	0	13		If unit shillings and pence ≥ 5/- add 5/- to complete 5/- round down

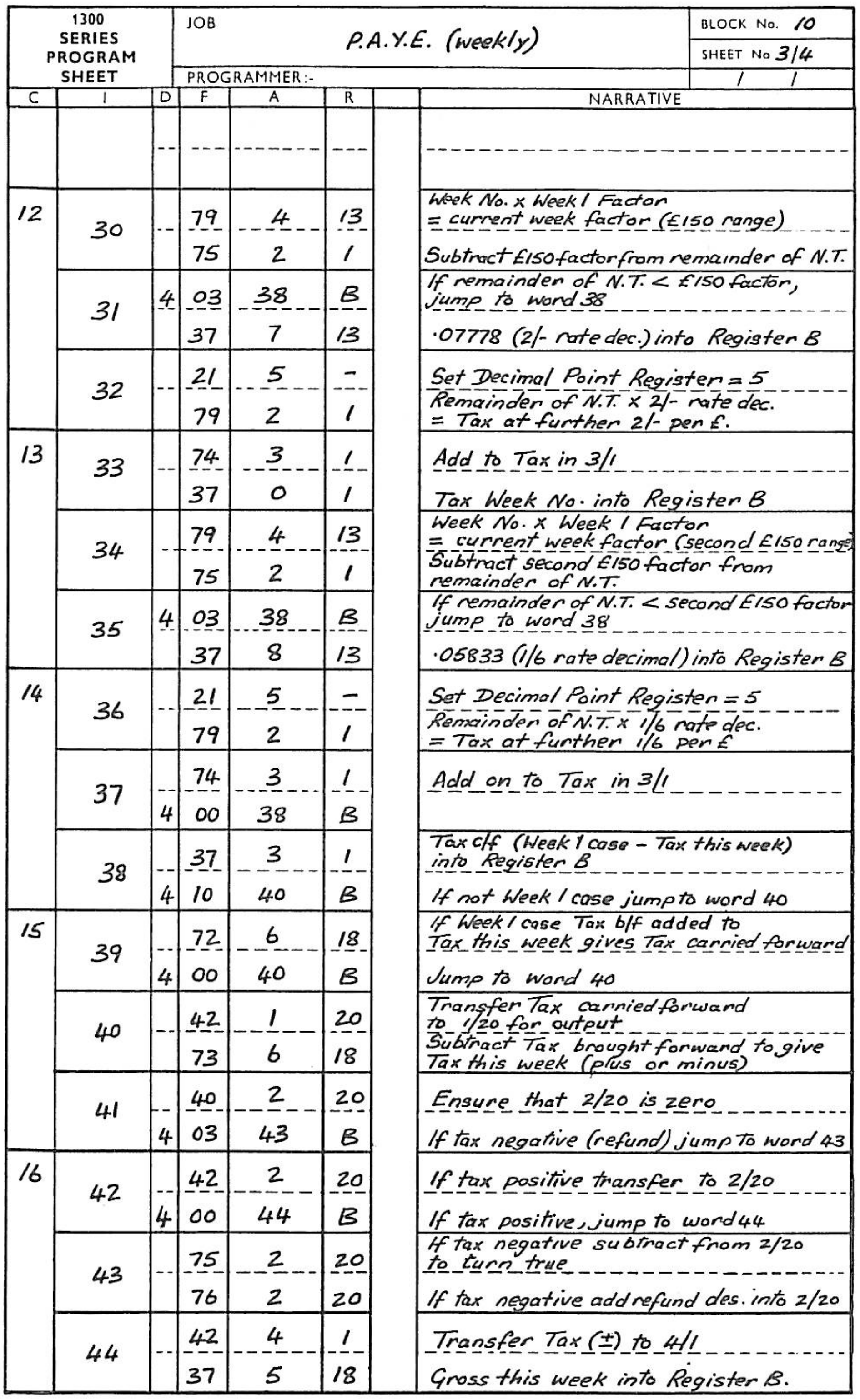
Figure 46: MAIN PROGRAM

1300 SERIES PROGRAM SHEET	JOB P.A.Y.E (weekly)	BLOCK No. 10	SHEET No 2/4
	PROGRAMMER:-	/ /	

C	I	D	F	A	R		NARRATIVE
7	15		42	2	1		N.T. transferred back to 2/1
			37	0	1		'Tax Week No.' into Register B
	16		54	7	–		Shift to 100's and 10's £s position
			73	2	1		Subtract N.T. from 10 x Week Number
	17		37	2	1		N.T. into Register B
		4	02	19	B		If N.T. $<$ 10 x Week No. jump to word 19
8	18		57	5	–	}	If N.T. $\geqslant$ 10 x Week No. eliminate unit shillings and pence to complete 10/- round down
			54	5	–		
	19		42	2	1		N.T. back to 2/1
			37	0	1		'Tax Week No.' into Register B
	20		69	1	13		Week No. x 16 on £s position in Reg. B
			73	2	1		Subtract N.T. from 16 x Week No.
9	21		37	2	1		N.T. into Register B
		4	02	23	B		If N.T. $<$ 16 x Week No. jump to word 23
	22		57	6	–	}	If N.T. $\geqslant$ 16 x Week No. eliminate all
			54	6	–		shillings to complete £1 round down
	23		42	2	1		N.T. (round down) to 2/1
			37	5	13		·06806 (1/9 rate decimal) into Reg. B
10	24		21	5	–		Set Decimal Point Register = 5
			79	2	1		Whole of N.T. x 1/9 rate decimal = Tax @ 1/9 per £
	25		42	3	1		Transfer Tax @ 1/9 rate to 3/1
			37	0	1		'Tax Week No.' into Register B
	26		79	3	13		Week No. x Week 1 Factor = current week factor (£60 range)
			75	2	1		Subtract £60 factor from N.T.
11	27	4	03	38	B		If N.T. $<$ £60 factor jump to word 38
			37	6	13		.09722 (2/6 rate dec.) into Register B
	28		21	5			Set Decimal Point Register = 5 Remainder of N.T. x 2/6 rate dec. = Tax at
			79	2	1		further 2/6 per £
	29		74	3	1		Add on to Tax in 3/1
			37	0	1		'Tax Week No.' into Register B

Figure 46: continued

I.C.T COMPUTERS



1300 SERIES PROGRAM SHEET	JOB	P.A.Y.E. (weekly)		BLOCK No. 10	SHEET No 3/4
	PROGRAMMER:-			/ /	

C	I	D	F	A	R		NARRATIVE
12	30	--	79	4	13		Week No. x Week 1 Factor = current week factor (£150 range)
			75	2	1		Subtract £150 factor from remainder of N.T.
	31	4	03	38	B		If remainder of N.T. < £150 factor, jump to word 38
			37	7	13		·07778 (2/- rate dec.) into Register B
	32	--	21	5	–		Set Decimal Point Register = 5
			79	2	1		Remainder of N.T. x 2/- rate dec. = Tax at further 2/- per £.
13	33	--	74	3	1		Add to Tax in 3/1
			37	0	1		Tax Week No. into Register B
	34	--	79	4	13		Week No. x Week 1 Factor = current week factor (second £150 range)
			75	2	1		Subtract second £150 factor from remainder of N.T.
	35	4	03	38	B		If remainder of N.T. < second £150 factor jump to word 38
			37	8	13		·05833 (1/6 rate decimal) into Register B
14	36	--	21	5	–		Set Decimal Point Register = 5
			79	2	1		Remainder of N.T. x 1/6 rate dec. = Tax at further 1/6 per £
	37	--	74	3	1		Add on to Tax in 3/1
		4	00	38	B		
	38	--	37	3	1		Tax c/f (Week 1 case – Tax this week) into Register B
		4	10	40	B		If not Week 1 case jump to word 40
15	39	--	72	6	18		If Week 1 case Tax b/f added to Tax this week gives Tax carried forward
		4	00	40	B		Jump to word 40
	40	--	42	1	20		Transfer Tax carried forward to 1/20 for output
			73	6	18		Subtract Tax brought forward to give Tax this week (plus or minus)
	41	--	40	2	20		Ensure that 2/20 is zero
		4	03	43	B		If tax negative (refund) jump to word 43
16	42	--	42	2	20		If tax positive transfer to 2/20
		4	00	44	B		If tax positive, jump to word 44
	43	--	75	2	20		If tax negative subtract from 2/20 to turn true
			76	2	20		If tax negative add refund des. into 2/20
	44	--	42	4	1		Transfer Tax (±) to 4/1
			37	5	18		Gross this week into Register B.

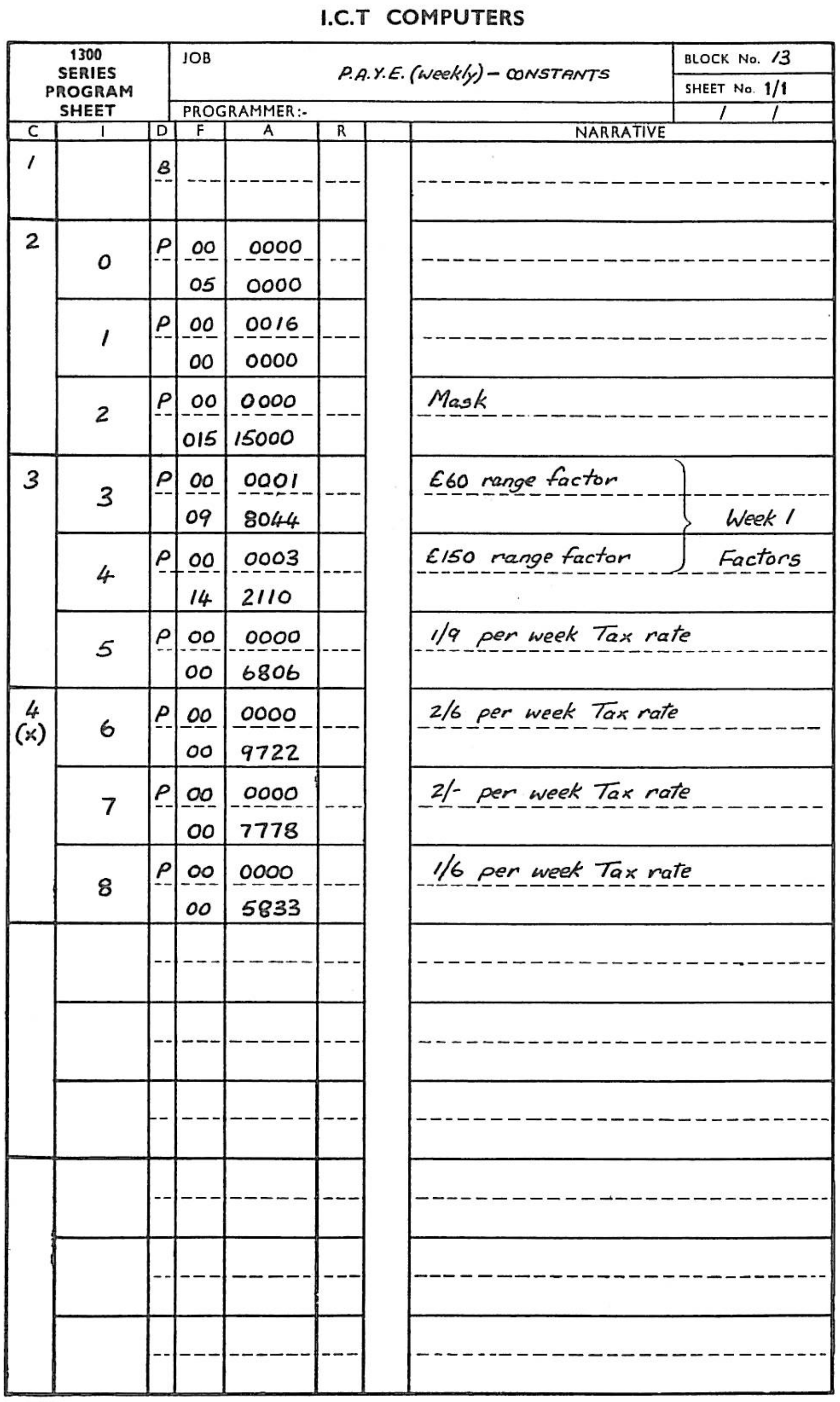
Figure 46: continued

I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET	JOB P.A.Y.E. (weekly)	BLOCK No. 10
		SHEET No. 4/4
	PROGRAMMER:-	/ /

C	I	D	F	A	R		NARRATIVE
17 (x)	45		73	4	1		Subtract tax this week
			73	7	18		Subtract total deductions to give net wage
	46		42	3	20		Net Wage to 3/20 for output

Figure 46: continued

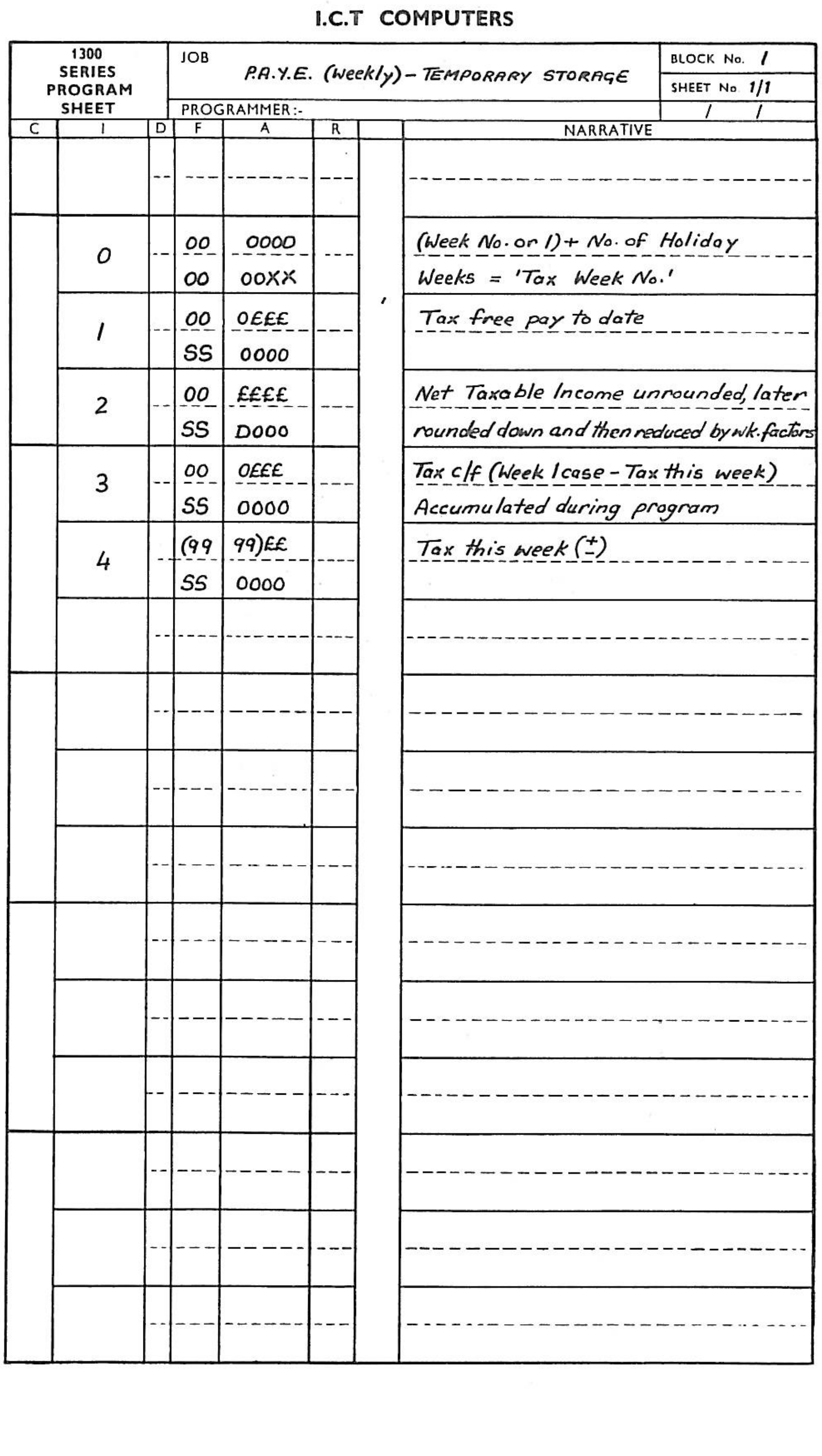


I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET	JOB P.A.Y.E. (weekly) – CONSTANTS	BLOCK No. 13
		SHEET No. 1/1
	PROGRAMMER:-	/ /

C	I	D	F	A	R		NARRATIVE
1		8					
2	0	P	00	0000			
			05	0000			
	1	P	00	0016			
			00	0000			
	2	P	00	0000			Mask
			015	15000			
3	3	P	00	0001			£60 range factor }
			09	8044			Week 1
	4	P	00	0003			£150 range factor } Factors
			14	2110			
	5	P	00	0000			1/9 per week Tax rate
			00	6806			
4 (x)	6	P	00	0000			2/6 per week Tax rate
			00	9722			
	7	P	00	0000			2/- per week Tax rate
			00	7778			
	8	P	00	0000			1/6 per week Tax rate
			00	5833			

Figure 47: CONSTANTS



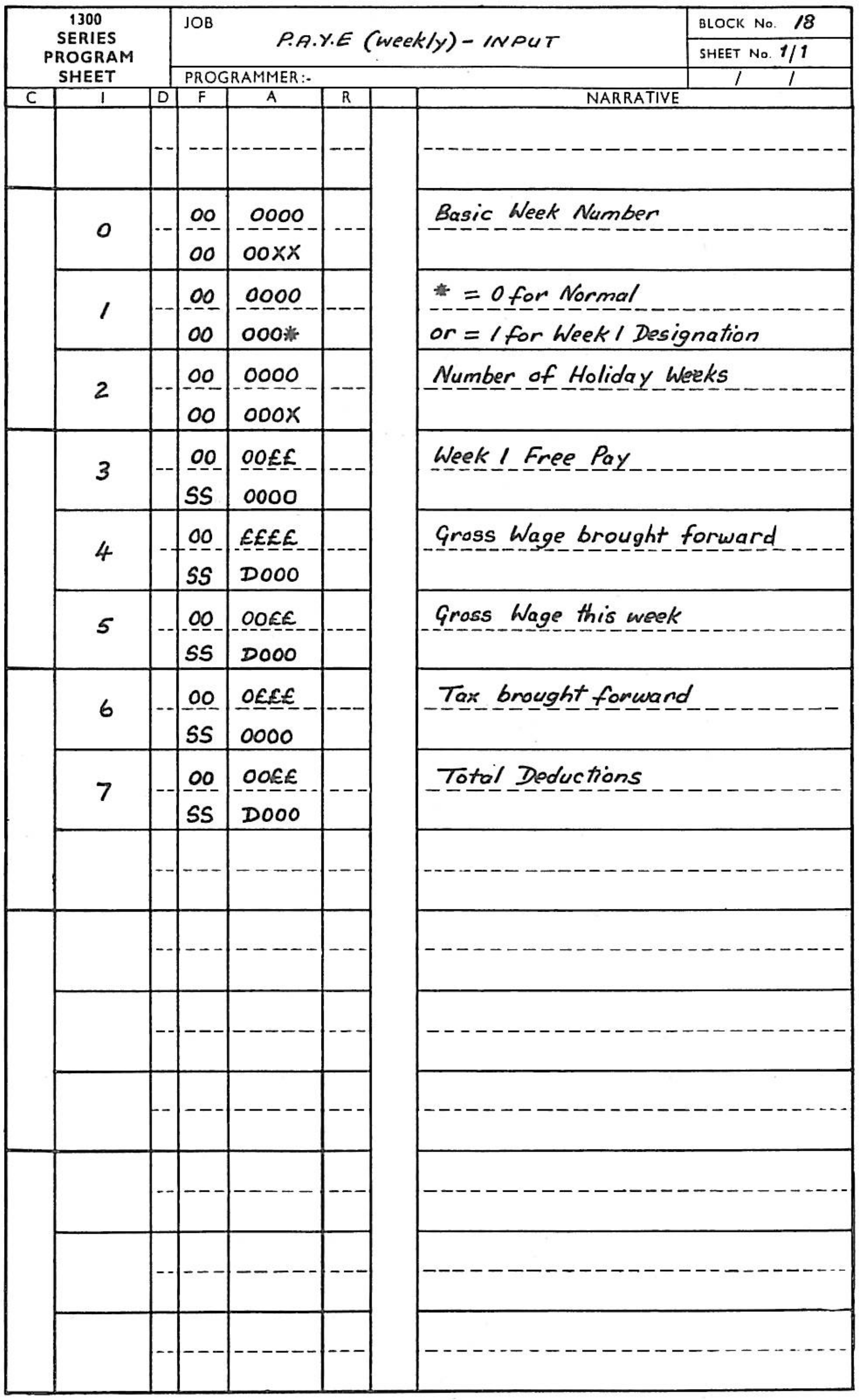
I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET	JOB *P.A.Y.E. (Weekly) – TEMPORARY STORAGE*
BLOCK No. *1*	SHEET No. *1/1*
PROGRAMMER :-	/ /

C	I	D	F	A	R		NARRATIVE
	0		00 00	0000 00XX			(Week No. or 1) + No. of Holiday Weeks = 'Tax Week No.'
	1		00 SS	0£££ 0000		'	Tax free pay to date
	2		00 SS	££££ D000			Net Taxable Income unrounded, later rounded down and then reduced by wk. factors
	3		00 SS	0£££ 0000			Tax c/f (Week 1 case - Tax this week) Accumulated during program
	4		(99 SS	99)££ 0000			Tax this week (±)

Figure 48: **TEMPORARY STORAGE**

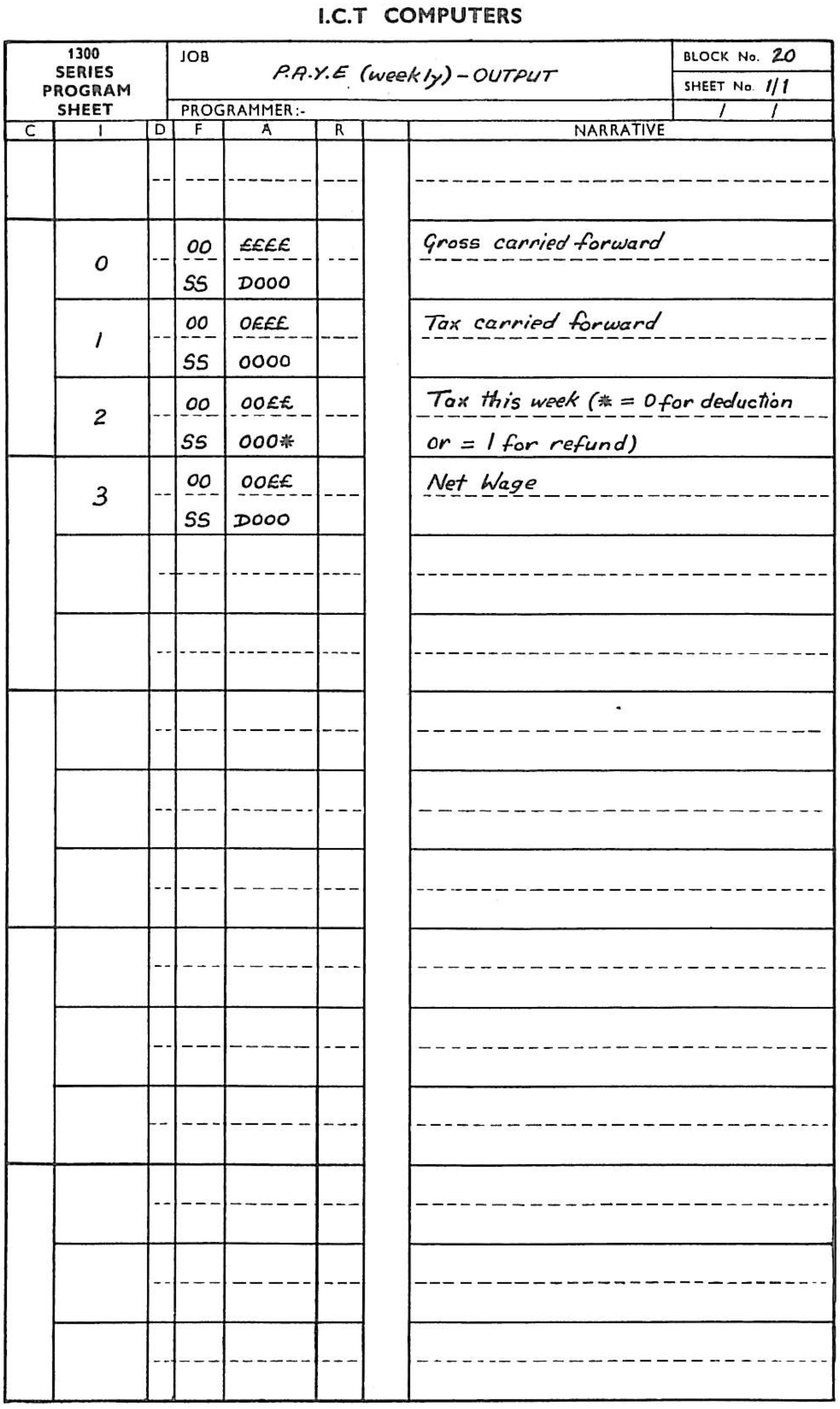
I.C.T COMPUTERS



1300 SERIES PROGRAM SHEET	JOB P.A.Y.E (weekly) - INPUT	BLOCK No. 18
		SHEET No. 1/1
	PROGRAMMER:-	/ /

C	I	D	F	A	R		NARRATIVE
	0		00	0000			Basic Week Number
			00	00XX			
	1		00	0000			✱ = 0 for Normal
			00	000✱			or = 1 for Week 1 Designation
	2		00	0000			Number of Holiday Weeks
			00	000X			
	3		00	00££			Week 1 Free Pay
			SS	0000			
	4		00	££££			Gross Wage brought forward
			SS	D000			
	5		00	00££			Gross Wage this week
			SS	D000			
	6		00	0£££			Tax brought forward
			SS	0000			
	7		00	00££			Total Deductions
			SS	D000			

Figure 49: INPUT INFORMATION



I.C.T COMPUTERS

1300 SERIES PROGRAM SHEET	JOB P.A.Y.E (weekly) - OUTPUT	BLOCK No. 20
		SHEET No. 1/1
	PROGRAMMER:-	/ /

C	I	D	F	A	R		NARRATIVE
	0		00	££££			Gross carried forward
			SS	D000			
	1		00	0£££			Tax carried forward
			SS	0000			
	2		00	00££			Tax this week (✱ = 0 for deduction
			SS	000✱			or = 1 for refund)
	3		00	00££			Net Wage
			SS	D000			

Figure 50: RESULTS OF CALCULATIONS