

THE ENGLISH ELECTRIC COMPANY LIMITED

DEPT. DATA PROCESSING AND CONTROL SYSTEMS.

REPORT No. K/GD.y.80.

KIDSGROVE

A.C.E.D. Ref. No.

F.S.D. or ORDER No.

Deptl. Ref.

Date 10.12.62.

Copies

TO: -

- | | |
|---|--|
| 1 | Mr. H.M. Mathews, Director of Engineering, London (Summary Sheet). |
| 1 | Mr. J.M. Ferguson, Chief Engineer, Stafford. |
| 1 | Mr. R. Allmark, Computer Development, Kidsgrove. |
| 1 | Mr. J. Boothroyd, Chief Instructor, DP&CSD, Kidsgrove. |
| 1 | Mr. T.H. Elliott, Chief Product Design Engineer, Kidsgrove. |
| 1 | Mr. D. Farthing, Computer Development, Kidsgrove. |
| 1 | Mr. A.C.D. Haley, Chief Development Engineer, DP&CSD, Kidsgrove. |
| 1 | Mr. R. Lloyd, Computer Development, Kidsgrove. |
| 1 | Mr. J.R. Lucking, Computer Development, Kidsgrove. |
| 1 | Mr. A.W. Nicholson, Computer Development, Kidsgrove. |
| 1 | Mr. J.P. O'Neil, Computer Development, Kidsgrove. |
| 1 | Mr. G.D. Royle, Group Engineer, F.E.S. Kidsgrove. |
| 1 | Chief Engineer's File. |
| 1 | MEL, Library, Whetstone. |

KDF 9: SHIFTING AND SHIFT CONTROL.

Report by

J R. Lucking.
J.P. O'Neil.

C2.

SUMMARY

This report describes the parts of Arithmetic Control and the Arithmetic Unit which perform the shifting operations in the KDF 9. The bulk of the design was performed by Messrs. R. Allmark and D. Farthing.

INDEX.

- | | |
|-----|---|
| 1.0 | INTRODUCTION |
| 2.0 | THE SHIFTING REGISTERS |
| 3.0 | SHIFT PATH SELECTION |
| 4.0 | FLOATING POINT OPERATIONS - NORMALISATION |
| 4.5 | THE INSTRUCTION BITS |
| 5.0 | CHARACTER CONVERSION OPERATIONS |

APPENDIX I - PACKAGE ALLOCATION FOR SHIFT PATHS.

1. INTRODUCTION

1.1 General

To perform shifts, AC sets the number to be shifted into the W registers, transfers the number of shifts required (and other information) to a "slave" unit called shift control (SHC), and supplies a sequence of control pulses to this unit. Also associated with SHC is logic used for the automatic normalisation of floating point numbers, and logic used for character conversion operations (i.e. TOB and FRB).

Shift control can logically combine W1 and B1 in such a way as to effectively make W1 a shifting register. Similarly W2 and B2 may be combined (only W1 is directly available to the programmer for single length shifts). Either of the shifting registers is capable of performing, in 1/μsec, any one of the following shift operations :

Right or left (i.e. $\div 2^n$ or $\times 2^n$) shifts of
0, 1, 2, 3, 4, 5, 6, 7, 8, 16 places

W1 can perform arithmetic and logical shifts whereas W2 only performs arithmetic shifts. The two shifting registers may be combined to form a double-length shifting register for both arithmetic and logical shifts as detailed above.

Shifts of a greater number of places (than 8 or 16) are performed automatically by SHC, as a combination of these "basic" shifts. First, any 16 (place) shifts are executed (if possible), then an 8-shift (if necessary), then any shift of less than 8 (if necessary).

1.1 Arithmetic Logical and Cyclic Shifts.

In all arithmetic right shifts, the sign bit of the shifting register is extended into those bit positions which become free (at the most significant (m.s.) end); i.e. if W1 is shifted 8 places to the right then D0 is copied into D1 to D8.

In all arithmetic left shifts, the overflow register (VR) is SET if either :-

- i) Any of the bits shifted off the ms end differ from the original sign,
- or ii) The sign of the shifted number differs from that of the original.

Also, the freed bit positions at the least significant (ls) end are set to zero. In all double-length shifts, W1 is the ms register; if arithmetic the sign digit, D0, of W2 is ignored i.e. D1 of W2 is treated as adjacent to D47 of W1. (Note that D0 of W2 is cleared during SHAD except when zero shift is specified.)

During logical shifts, setting of VR is inhibited, and if a right shift, the digit positions freed at the ms end are set to zero. In double-length logical shifts D0 of W2 is included, being treated as adjacent to D47 of W1.

Cyclic shifts are executed by copying the contents of W1 into W2 and performing double-length logical shifts. The result is in W1 if a left shift, in W2 if a right shift. Thus the number of cyclic shifts must be less than 48, or an incorrect result will be obtained.

/Continued.....

1.2 Underflow

A two bit register (WUFL) is incorporated at the least significant end of W1 (or W2), and is filled with any bits shifted off the 1s end of this register, i.e. WUFL may be considered as containing D48 and D49 of the shift register. These bits are not accessible to programmers, but are used to ensure correct rounding-off in certain arithmetic operations. (For double-length operations WUFL acts as D48 and D49 of W2).

1.3 Normalisation

Attached to W1 is a set of logic which allows normalisation of contents of this register. This normalisation takes place in two stages: logic, spanning the full length of W1, determines the number of 8-shifts (to the left) required such that the digits in D0 to 8 will be non-identical. These 8-shifts take place, and when completed, a further set of logic operating on D0 to 8 only determines the number of shifts required to make D0 and D1 non-equivalent.

The normalisation can also be made to the D0 : D9 pair; and an additional set of logic for determining shifts of less than 8 exists on D0, D8 to 16. The normalisation can be double-length i.e. operating on the d.l. arithmetic register W1 and W2. Normalisation is organised and executed by shift control.

1.4 BITS

The bits counting operation is done by a series of logical right shifts, each of two places. After each shift, the least significant two digit positions are checked, and the number of bits in these positions (0, 1 or 2) added into a counter. At completion, the contents of the counter are placed in W1. The operation is carried out by shift control.

1.5 TOB and FRB

These operations are also carried out by shift control, and utilise additional special logic on W1. They are too complex for a brief description, so are treated fully in section 5.

2. THE SHIFTING REGISTERS

General

To provide the various 'shifting registers' available, shift control selects and drives suitable paths from a complex of transfer paths interconnecting the registers W1, B1, W2 and B2. The principal paths provide:

- i) RIGHTshifts from the W to the B registers of 0, 1, 2, 5, 8 and -8 (i.e. left 8) binary places
- ii) LEFT shifts from the B to the W registers of 0, 1, 2, 5, 8 and -8 binary places.

These are used in the combinations of a W to B shift, followed by a B to W shift, as shown below, to produce the shift register characteristics required.

No. of Shifts.	Right Shifts.		Left Shifts.	
	W to B	B to W	W to B	B to W
1	R1	0	0	L1
2	R2	0	0	L2
3	R5	L2	R2	L5
4	R5	L1	R1	L5
5	R5	0	0	L5
6	R8	L2	R2	L8
7	R8	L1	R1	L8
8	R8	0	~~0~~	L8
16	R8	R8		

not possible to be done in the greater than 8 shifts.

These transfers occur at $\frac{1}{2}$ usec. intervals giving the shift register a basic operation time of $\frac{1}{2}$ usec.

Simple transfers as specified above make up the majority of the transfer paths; however, most of the complexity of shift transfers is associated with the transfers to and from digit positions near the end of the registers.

For example, consider the transfer out of D46 of W1 when executing a RIGHT shift of 2 binary places. This will go to one of 3 locations depending on the type of shift register being simulated:-

- i) Single length logical or arithmetic to D0 of BUFL
- ii) Double length logical to D0 of B2
- iii) Double length Arithmetic to D1 of B2

2.1 The 'B' Underflow Register (BUFL).

Associated with the least significant end of the B registers is BUFL, a 4 bit register which corresponds to WUFL for the W registers. Examination of the table in 2.0 reveals that, in executing both left and right shifts, an "excess" shift right of up to 2 binary places can exist in the B register/s at an intermediate stage. (No excess left shift ever occurs). The 4 bits of BUFL are necessary to ensure that

- i) the least significant bits of the number are not lost;
- and ii) in right shifts, 2 underflow bits (in WUFL) are preserved at the completion of shift.

As for WUFL, any shifting operations which do not transfer into all digit positions of BUFL clear the unused positions. For W to B transfers, WUFL is treated as additional bits of the appropriate W register; and for B to W transfers, BUFL is treated as additional bits of the B register. (Some of the paths provided in these transfers are redundant).

2.2 The Transfer Paths.

Fig.1 summarises the transfer paths available between the W and B registers. It is not intended to provide detailed information on any of these paths, nor to relate the paths to the operations in which they are used. Table 1 completely defines all the transfer paths available for shifting, most of these paths are composed of a number of individual transfer transformers; however these cannot be logically separated i.e. they are basic actions.

In Table 1, where possible those paths required for a particular shift operation are grouped together e.g. W1 to B1 right 8

/Continued.....

shifts includes the paths required for arithmetic and logical, single and double length shifts of this register pair. There are about 80 shift transfers available; shift control selects and pulses those required for any particular shift operation.

2.3 Packaging of Shift Transformers.

The large number of transfer paths between the B & W registers required special packaging techniques for these transformers, to achieve these paths with a minimum number of packages. The packaging takes advantage of the multiple transfers to (and from) any digit position in the B & W registers. Fig.2 illustrates the shift paths for a section of the B1 and W1 registers. As shown in Appendix 1, these 7 4-bit paths are achieved using only two packages, both specially designed for this requirement.

Appendix 1 details all the special shift packages used, and Table 1 specifies the types of packages used in the various shift paths.

3. SHIFT PATH SELECTION

3.0 General

To perform the various "shifts" shift control selects and drives a suitable sequence of the shift paths detailed in Section 2 (see also the various tables). The selection logic is based upon two registers, the Shift Number Register (SHN) and the Shifting Register, (SHR), and a number of "shift mode" indicators. The logic gates a matrix of transformers, the Shift Path Selection Matrix (SHSM), which drives a combination of the pulse amplifiers shown in Table 1. Arithmetic Control supplies the pulses which energise SHSM. Figure 3 is a block diagram of shift control, many of the details of which are discussed in subsequent sections.

3.1 The Shift Mode Indicators.

There are three indicators in SHC which determine the mode (or type) of shift to be performed. They are:-

SHL - if reset then perform an arithmetic shift;
if set perform a logical shift.

SHD - if reset perform a single length shift;
if set perform a double length shift.

SHW2 - if reset shift W1 (and W2);
if set shift W2 (single length) independently.

These provide apparently 8 possible shift modes: however, if SHW2 is set then both SHL and SHD are reset (W2 is a single length shifting register capable of only arithmetic shifts).

The indicator SHL also inhibits the setting of the overflow register during left shifts and ensures that zeroes are inserted into digit positions at the most significant end of W1/B1 during right shifts.

Gates connected to these 3 indicators provide 6 further levels which if at ONE mean :-

W1/D - Shift W1/B1 independently or with W2/B2 as a double length shift.
 W2/D - Shift W2/B2 independently or with W1/B1 as a double length shift.
 W2S - Shift W2/B2 independently (and thus single length arithmetically).
 W1S - Shift W1/B1 single length.
 DLOG - Shift W1/B1 and W2/B2 logically double length.
 DART - Shift W1/B1 and W2/B2 arithmetically double length.

These gates are shown on Fig.4.

These six levels are used to gate the secondary windings of the individual SHSM transformers and thus determine which paths of each group of paths will be used. E.g. during a double length logical shift of 19 places to the right, one of the steps - the first - will be to shift W1 and W2 to B1 and B2 right 8 places; the paths required to perform this shift are those numbered 1, 3 and 5 (see Table 1) and these will be selected by the levels W1OD, W2OD and DLOG being at ONE (the other levels are at '0').

3.2 The Shift Number Register (SHN).

The shift number register, SHN, which has a capacity of 9 bits, is set by arithmetic control to hold the number of shifts to be performed. This register SHN may thus hold a number of shifts between -256 and +255. Shift control can never be asked to perform a left shift of more than 127 places, as in the instructions SHA $\pm n$, SHACq only 8 bits are allowed to specify the number of shifts. However, during floating add the number of shifts calculated (by AC) by subtracting the exponents may be as large as -255 (and is always a right shift).

Even so, shifts in either direction of more than 48 places, for single length; or 96 places for double length, are meaningless and SHC performs all shifts of more than 96 places merely by shifting between 96 and 104 places. A logical level, the "shift out of range" level (SHOR), is generated to check that the modulus of the number in SHN is less than 96. This level is at ONE for any of the following patterns of D0 to D3 of SHN:

Digit	0	1	2	3	
	0	0	1	1	(As left shifts in excess of 127 places cannot arise it is not necessary to check the range +128 to +255).
	1	0	0	0	
	1	0	0	0	

Fig. 5 shows this logic.

If SHOR is at ONE a minimum of 96 shifts (Maximum is 104, see later) in the appropriate direction will be performed. The transfer from the Shift Buffer does not affect SHN (digit 1); this would lead to error when performing right shifts so SHN (digit 1) is always set to 1 before any shift is performed (and ignored for left shifts). The only exception to this rule is when transferring the exponent difference to SHN, when digit 1 may be cleared if number of right shifts required is 127. Thus, unless the number of shifts is out of range, SHC uses only digits 0, 2, 3, 4, 5, 6, 7 and 8 of SHN. Associated with the SHN register is a counter, used when normalising and in performing BEPS: this counter increments the contents of SHN by adding either 8 or 16.

3.3 The Shifting Register (SHR).

As has been stated in para. 2., shifts of a large number of places are performed in three stages :-

- (i) do as many shifts of 16 places as possible.
- (ii) if necessary perform a shift of 8 places by using the appropriate 8 shift path from W1/W2 to B1/B2 and a straight transfer back.
- (iii) if necessary complete the shift by shifting 1,2,3,4,5,6, or 7 places.

This is achieved by Shift Control in two steps:-

- (a) perform as many shifts of 8 places, (in the sequence W's to B's, B's to W's), until the number of shifts left is less than eight. The data word may now be in the W's (if a multiple of 16 shifts were performed) or in the B's (if an odd number of 8 shifts were performed). If necessary, therefore, the last 8 shift from the B's to the W's is converted into a straight transfer.
- (b) complete the shift by shifting 1,2,3,4,5,6 or 7 places.

Shift Control employs a further register of 12 bits, the Shifting Register (SHR), to control the number of 8 shifts performed in step (a) above. SHR is set up from SHN, or the normalisation logic (see Section 4), such that it contains one bit (from the l.s. end) for each 8 shifts to be performed.

Initially SHR is clear. Logic (see Fig.6) is attached to digits 0.2.3.4. and 5 of SHN. Suppose these digits contain a signed integer n (D0 is the sign digit, D5 regarded as the l.s. digit). If n is positive, the logic sets the n right-most digits of SHR; if n is negative then the (1n1-1) right most digits of SHR are set. However, if SHOR is at '1', all digits of SHR are set.

An 8 shift is performed every $\frac{1}{2}$ /usec, simultaneously the string of ONES in SHR are shifted right one digit position. The 8 shifts continue until SHR is zero, i.e. until digit 11 is zero. If an odd number of 8 shifts has been performed, an additional pulse transfers the data directly from the B registers back to the W registers.

The above description was simplified by assuming that SHR is a perfect shifting register. As Fig.7 shows, this is not so: the '0' outputs of the flip-flops gate the resetting of the next significant flip-flop, odd digit positions can be reset on the B to W 8 shifts (at P1 time), even digit positions on the W to B 8-shifts (at P2 time). This operation of SHR requires the use of two additional indicators, Shifting Register Non Zero (SHRNZ), and Odd number of 8-shifts (SHOD). SHOD is set up from the logic setting SHR, SHRNZ is set from SHR(11); both are set by the P2 shift pulse.

3.4 The Shift-Path Selection Matrix (SHSM).

This is a 'matrix' of transformers, rather like a sequence unit, which selects the appropriate shift transfer paths. It consists of two banks of 7 transformers, one bank supervising the paths from the W to the B registers and therefore being pulsed at P₂ time - the other bank supervises the paths from the B to the W registers and is pulsed at P₁ time. Each bank is divided into two sections, one section is pulsed if 8-shifts are being performed (i.e. SHR is non-zero); the other is pulsed during the last micro-second of the shift when "<8 shifts" are being performed.

/Continued.....

The pulse amplifiers which pulse the sections are called P_2 8-shift, P_2 8-shift, P_1 8 shift and P_2 8 shift. They pulse:-

P_2 8-shift - two transformers which control the shift transfer paths from $W1/W2$ to $B1/B2$ providing left and right shifts of 8 places

P_2 <8-shift - Five transfers which control the shift transfer paths from $W1/W2$ to $B1/B2$ providing right shifts of 0,1,2,5 and 8 places.

P_1 8-shift - three transformers which control the shift transfer paths from $B1/B2$ to $W1/W2$ providing left and right shifts of 8 places, also no shift. The latter is used when an odd number of 8-shifts are performed.

P_1 <8-shift - Five transformers which control the shift transfer paths from $B1/B2$ to $W1/W2$ providing left shifts of 0,1,2, 5 & 8 places.

The primary gating which selects which transformer in a section will be pulsed is based on the sign of SHN (for 8 shifts) and upon digits 0,6,7, 8 of SHN for 8 shift. See table 1 for details of SHSM. Figs. 8 and 9 show details of these primary gating levels.

3.5 Other Indicators

Apart from SHOD and SHRNZ which, together with SHR, supply conditions for terminating the shift there is an indicator SHIFT which is set when SHC starts any shift instruction. Whilst this indicator is set SHC will loop between sequence unit transformers P_2 63 and P_1 38, performing the appropriate shift. SHIFT is reset if either

- (i) The right-most digit of SHR (i.e. SHRD11) is zero - for at this time the final shift of 8 places is being performed.
- or (ii) Digit 9 of SHR is zero and DO, 6,7,8 of SHN are zero. This means that a left shift of an integer multiple of eight places has been specified and the 8-shift now being performed is the last.

Note that right shifts of an integer multiple of 8 places always terminate with a 8 shift of 8 places.

See Fig. 12.5.

4. FLOATING POINT OPERATIONS - NORMALISATION.

4.0 General

KDF 9 performs a range of arithmetic instructions operating on floating point numbers. A floating point number is of the form

$$f \times 2^e$$

where f = a fractional part between $\frac{1}{2}$ and 1 (in modulus) and e = the exponent in modulus 128.

In KDF 9, the fractional part of a floating point number occupies digits 0, 9-7 whilst the characteristic occupies digits 1-8. (The characteristic is merely the exponent plus 128 - this form being adopted so that the number zero has the lowest permitted exponent). A full description of floating point operations is given in report K/GD.y.83, this section merely describes a facility provided by shift control which ensures that the result of a floating point operation is in standard form - i.e. with its fractional part f between $\frac{1}{2}$ and 1 (in modulus).

4.1 The Instruction 'STANDARDISE'

For the instruction STAND, a floating number x is supplied as

/continued

data in W1

$$x = f \times 2^e$$

where $-1 \leq f \leq +1$

The result of the operation is a number x' in W1 such that

(i) $x^1 = x'^1 = f' \times 2^{e'}$
and (ii) either $+\frac{1}{2} \leq f' \leq +1$

or $-1 \leq f' \leq -\frac{1}{2}$

or f' and e' are both zero.

(In all KDF.9 logic and engineering literature the term exponent and its abbreviations are really misnomers for the characteristic as defined above).

To perform the operation STAND, Arithmetic Control :-

- Stores e in the exponent register and then if f is +ve clears D1 - 8 of W1 or if f is -ve sets ones in D1-8 of W1,
- If W1 is now zero, i.e. f was zero, finishes the instruction, or,
- Shifts W1 to the left n places until D8 and D9 of W1 are different.

This process effectively multiplies f by 2^n ,

- Subtracts n from e and places the result in D1-8 of W1.

The new fractional and exponent part of x' are $f' = f \times 2^n$
and $e' = e - n$

$$\begin{aligned} \text{thus } x^1 &= f^1 \times 2^{e^1} \\ &= f \times 2^n \times 2^{e-n} \\ &= f \times 2^e \\ &= x \end{aligned}$$

The shifting operations in step (c) are performed by Shift Control, and the process is known as normalisation. Normalisation shifting is different from all other shifting operations in that the number of shifts required is determined by SHC inspecting W1 and not by inspecting the SHN register.

4.2 Normalisation Facilities.

At the end of nearly all operations on floating point of numbers Arithmetic Control ensures that the resulting floating number is in standard form, consequently SHC control may normalise both single and double length numbers. In these normalisation processes SHC performs those shifts of W1 (and W2) which ensure that -

D0 of W1 $\neq$ D8 of W1

and D0 of W1 $\neq$ D9 of W1 (after the shift)

This process is called 'Normalisation to D9'.

The division algorithm requires that the divisor presented to the multiplier-divider unit be a fraction greater than a half (in modulus). Arithmetic Control ensures this by instructing SHC to shift W1 until D0 $\neq$ D1 - this latter process being normalisation to D1. If normalisation to D1 of W1 is required the indicator NORM 1 will be set before AC uses SHC. If the number to be normalised is double length AC will set SHD. SHC ensures that both the indicators SHL and SHW2 are reset during any normalisation shifting. Note that the facility of normalising a double length number to the D1 position is never used.

/Continued.....

4.2.1. Direction of Shift during Normalisation.

The above discussion has assumed that the fraction to be normalised is always, in modulus, less than one and that any shifts required will be LEFT Shifts. Thus if the fraction were less than $\frac{1}{8}$ but greater than $\frac{1}{16}$ it must be multiplied by 8 (2^3), or shifted LEFT 3 places to restore it to standard form. This assumption is not always true, e.g. the result of adding -

$$\begin{array}{r} .525 \times 2^0 \\ \text{and } .736 \times 2^0 \\ \hline \end{array}$$

is 1.261×2^0

and the resulting "fraction" must be shifted to the right one place to give the answer $.6305 \times 2^1$. There is never any need to shift the "fraction" more than one place to the right and the condition only arises when NORM1 is reset - thus one right shift will be performed when SHC control discovers that :-

NORM1 = 0, and (D0 of W1) $\neq$ (D8 of W1).

4.3 The Normalisation Processes.

Of the two normalisation processes, Normalise to D9 and Normalise to D1; only the first will be described here. Normalisation is performed in two separate steps :-

(i) Inspect the contents of W1 (and WUFL (D0) if single length, or D0 of W2 if double length), and determine :-

- a) If any normalisation is required - vis. is $D0 \equiv D8 \neq D9$ *fake*
and b) If more than 8 left shifts are required.

This latter step consists of examining the digit groups D8-D16, D16-D24, D24-D32, D32-D40, D40-D47 (and either WUFL (D0) or W2(D0)), up to the first group which is not all zeroes or all ones, and inserting a 'one' into SHR for every prior group. (Note that SHC has now checked that $D0 \equiv D8$). Thus the process is stopped as soon as a digit in W1 is discovered that is different from all its predecessors. For example, if D0-D33 of W1 are all ones, but D34 is zero, then D9, 10 and 11 of SHR will be set. Shift control then performs these 8-shifts until SHR is zero, simultaneously counting the number of shifts in to SHC (which has previously been cleared).

If 40 left-shifts were performed the whole step will be repeated, if after repeating W1 is still all zeroes (note it cannot be all ones), the indicator ZERO or EXUF (exponent underflow) will be set and the normalisation process terminated. Fig.11 shows the normalisation 8-shifts logic.

(ii) Inspect digits 0,7,8,9 ... 15 of W1

- a) If $D0 \equiv D7 \neq D8$ then set all ones into SHN and perform a shift of one place to the right.
b) If $D0 \equiv D8 \neq D9$ then no shift is required.
c) Examine, in sequence, logical levels which are ONE if D9 D10, D10 D11, ..., D14 D15. ?
and discover the first ONE.

This determines the number of left ≤ 8 shifts required and this number is set into D6-8 of SHN and the shifts performed. For example, if D0,D8,D9,D10,D11 are all zeroes and D12 a 'one' then 3 left shifts are required and D7 and D8 of SHN will be set. Fig.12 shows the normalisation ≤ 8 shifts logic.

The process is now complete:

W1(and W2) contains the normalised fraction and SHN contains the number of shifts that have been performed. /Continued.....

4.4 The Shift Number Counter.

This counter is capable of adding one or two to digits 1-5 of SHN. When performing the 8-shifts in normalisation it is pulsed every microsecond and adds two unless the indicator SHOD is set and SHR digit 10 is zero when it will add one. Note that when an odd number of eight shifts are being performed SHOD will be set during the last 8-shift (from W to B).

4.5 The Instruction 'BITS'.

The instruction BITS works as follows :-

Start by clearing BTF counter (that is used to count the number of bits).

Set shift mode to single length, logical on W1 by setting SHL and clearing SHD and SHW2.

Set SHN to 1 00000110 and clear SHOD, E NORM - this causes SHC to shift right 2 places every microsecond as the 8-shift decoding logic is not pulsed.

Set the control flip-flops BITM & shift & enter the shift loop.

→ P₂ beat Shift W1 RIGHT 2 places to B1

IF SHN (digits 1 and 2) are both 1 then reset BITM

IF BITM = 0 and SHR (111) = 0 then reset ~~the~~ SHIFT,

P₁ beat Transfer B1 to W1

IF BITM = 1 then add 1 to SHN (digits 1-5)

IF BITM = 1 and either BUFL (digit 0 or 1) is one then count in BTF

IF BITM = 1 and both BUFL (digits 0 and 1) are one then count 2 in BTF.

Is ~~SHC~~ ^{SHIFT} = 1 ?

YES

NO

Then transfer the complement of BTF to SHN, the complement of SHN to B2 (digits 1-8), thence, using a special shift path, into d40-47 of W1, (see action 38).

5.0 CHARACTER CONVERSION OPERATIONS.

The KDF 9 executes two characters conversion instructions, called TOB (to binary) and FRB (from binary). The data operated on by these instructions consists of:-

(i) in W2, a radix word

$R = r_0, r_1, r_2, \dots, r_7$, (each r_n is a 6 bit radix number)

and (ii) in W1, either

(a) for TOB, a word of characters

$C = C_0, C_1, C_2, \dots, C_7$ (each C_n is a 6 bit character).

or (b) for FRB, a positive binary integer I.

TOB for s the positive binary integer

$$I = r_7 (r_6 (r_5 (r_4 (r_3 (r_2 (r_1 \cdot C_0 + C_1) + C_2) + C_3) + C_4) + C_5) + C_6) + C_7$$

leaving the answer in W1.

FRB leaves in W1 the character word.

$C_0, C_1, C_2, \dots, C_7$

Where $C, r_1, C_2, r_2, \dots, C_7, r_7$ and the original integer is equal to

$$I = r_7 (r_6 \dots (r_1 \cdot C_0 + C_1) + C_2) + \dots + C_7$$

i.e. FRB performs the opposite operation to TOB. The action of these instructions is illustrated by an example. In this example, which is used later in the detailed explanation of TOB and FRB, only a 24 bit word is used.

Example :-

Suppose that a 'length' has been input in chains, yards, feet and inches - more specifically the length is

X = 5 chains 20 yards 0 feet and 7 inches.

then the character word is

C = 000101, 010100, 000000, 000111

and the radix word is

R = 001010, 010110, 000011, 001100

i.e., 12 inches to one foot, 3 feet to one yard, 22 yards to one chain, 10 chains to one furlong and all lengths are less than one furlong.

In inches the length is

$$I = 12 (3 (22 \times 5 + 20) + 0) + 7 \\ = 4687 \text{ inches}$$

i.e. $I = 000, 000, 000, 001, 001, 001, 001, 111$ in binary.

The instruction TOB is given the words C and R, and produces the binary integer I - the instruction FRB is given the radix word R and the binary integer I and produce the character word C.

In order to engineer these functions fairly simply the following limitations have been imposed on the values of the radices

- (i) All radices > 0
- (ii) All radices < 32

In TOB all the characters in the character word C must be less than the corresponding radices, thus the length

$X = 3$ chains, 22 yards 4 ft. and 9 inches is an invalid representation, and TOB may convert such a number incorrectly without setting overflow.

5.1 THE INSTRUCTION FRB.

The conversion of an integer into the required character string is most obviously achieved by the following process

- (i) Divide the integer by the least significant radix r_7 to form an integer quotient Q_7 and a remainder R_7 . R_7 is the least significant character C_7 .
- (ii) Divide Q_7 by the second least significant radix r_6 to form an integer quotient Q_6 and a remainder R_6 . R_6 is the second last significant character C_6 .
- (iii) Continue until r_0 has been divided into Q_1 , thus forming the 8 character string $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$, with $Q_0 = 0$.

For example, to convert $x = 4687$ inches into chains, yards, feet and inches :

Firstly, divide 4687 by 12

$$\text{thus } x = 390 \text{ feet and } 7 \text{ inches}$$

Secondly, divide 390 by 3

$$\text{thus } x = 130 \text{ yards } 0 \text{ feet and } 7 \text{ inches}$$

Thirdly, divide 130 by 22

$$\text{thus } x = 5 \text{ chains } 20 \text{ yards } 2 \text{ feet and } 7 \text{ inches}$$

Fourthly, divide 5 by 10 to ensure that the 'length' is not out of range.

The process is a series of divisions, in which the quotient obtained in one division becomes the dividend for the next division. A simple integer division process commences operating on the m.s. portion of the dividend and can be arranged to generate the quotient digit by digit, starting with the m.s. digit. As the quotient digits are generated, the next division process can commence on them, and so on for all the divisions; i.e. one step of each division is performed in sequence, then the next step of each division, and so on until the original integer is exhausted. This is the method used for performing FRB on KDF.9, and the algorithm is shown in the flow chart of Fig.13.

The positive binary integer is first effectively shifted (arithmetically) right 47 places. This ensures it is less than r_7 at the beginning of the process. The divisions can then be commenced, and will not be complete until a total of 47 shifts left have occurred. (The initial shift right is most readily done by making the word containing the integer the l.s. word of a double length operand. This can then be shifted left (automatically until the m.s. ONE digit is in D1 of the l.s. word)

5.1.1. The Division Process

Each step of the division process performs one of two actions:-

- (i) If the integer (in that character position) is greater than or equal to the radix, subtract the radix and add one to the integer in the next character position.
- (ii) If the integer is less than the radix, do nothing.

This is carried out for all character positions in turn starting at the l.s. character. Then the double length operand is shifted left one binary place, and another step of division performed.

Which of step (i) or (ii) is to be used in each character position is decided by a trial. The radix is subtracted from the integer in that character position: if the remainder is + ve, then (i) is used. If the remainder is - ve, then (ii) is used.

In KDF 9 subtraction is achieved by adding the negated subtrahend. The negated form of an n bit number (modulo 2^n) is obtained by adding 1 to the l.s. digit position of the ones complement. Thus, to obtain the negated radices, the radix word is complemented, and a one added into the l.s. digit of each radix position.

e.g.

$R = r_0, (10)$	$r_1, (22)$	$r_2, (3)$	$r_3, (12)$
$= 001010,$	$010110,$	$000011,$	001100
$-(R) = -r_0,$	$-r_1,$	$-r_2,$	$-r_3$
$= 110101,$	$101001,$	$111100,$	110011
$+ 000001$	000001	000001	000001
$= 110110,$	$101010,$	$111101,$	110100
$= -10,$	$-22,$	$-3,$	$-12,$

During the addition, no carry can occur between adjacent character positions unless one of the radices was zero: this is NOT allowed. Because the radices are all less than 32, the m.s. digit of each negated radix will be a sign digit, i.e. 1.

The word of negated radices is stored in register NB, and a copy placed in B2. This is added to the m.s. word of the operand, the sum being placed in W1.

If the contents of a character location of the m.s. word are less than the corresponding radix, the m.s. digit of this character location of the sum will be ONE; i.e. the addition of a +ve 6 bit number to a -ve 6 bit number has produced a negative 6 bit number. However, if the contents of a character position are greater than the corresponding radix, a +ve sum will be produced AND a carry out will occur into the next character position. This carry is precisely what is required to add one to the next m.s. character position; and means that a trial step of division is carried out in all character positions with the one addition.

From the result of this trial, the radices in B2 corresponding to those character positions with a negative remainder are cleared, and the addition repeated to give the new set of remainders in W1.

The d.l. operand (in W1 and W2) is shifted left one place, W1 transferred to B1, NB to B2, and the division process repeated. This continues until a total of 47 left shifts have occurred, leaving the required character string in W1.

A step in the process is illustrated, for 24 bit words.

B1	000000,	010110,	000100,	001010
B2	110110,	101010,	111101,	110100
Add into W1	110111,	000001,	000001,	111110
(B1	000000	010110	000100	001010)
Clear B2	000000	101010	111101	000000 (from result above)
radices				
Add into W1	000001	000001	000001	001010

5.2 TOB

The conversion of a mixed radix number into a binary integer is most obviously achieved by a process of nested multiplication as follows:-

- (i) Multiply the most significant character C_7 by the radix r_6
- (ii) Add the second most significant character (C_6) to the product obtained from (i) and multiply this by the radix r_5 .
- (iii) Continue until the least significant character (C_0) has been added.

For example to convert the length

$x = 5$ chains 20 yards 0 feet and 7 inches

to an integer number of inches.

Firstly, multiply 5 by 22
thus $x = (110 + 20)$ yards 0 feet and 7 inches.

Secondly, multiply 130 by 3
thus $x = (390 + 0)$ feet and 7 inches.

Thirdly, multiply 390 by 12
thus $x = 4680 + 7$ inches.
= 4687 inches.

The process is thus a series of multiplications, the multipliers being the 7 radices $r_1 - r_7$ in turn and the multiplicands being essentially the product of the previous multiplication. A simple integer multiplication process forms the product digit by digit starting at the least significant end. Thus as each separate multiplication produces a digit, the next less significant multiplication commences and so on until the multiplication by the least significant radix has been performed. The next step in the multiplication is then commenced. This process is used in the KDF 9 to perform TOB and the algorithm is shown in the flow chart of Fig.14.

Note that this process differs from the normal multiplication process in the use of the multiplier register. In the KDF 9 multiplier the contents of the multiplier register are shifted one place to the right on each step this creating a hole at the left most end which is used to store the least significant digits of the product as they are generated. In the TOB algorithm the multiplier register (W1), which contains the character word at the start, is not so used for this purpose; it being simpler to use the double length shifting facilities of the register W1, W2, and to form the product (which must be single length anyway) bit by bit in the register 2. Consequently, the condition $W1 = 0$ may be used to terminate the process.

5.2.1 The Multiplication Process

Each step in the multiplication process performs one of two actions:-

- (i) If the character (in specified character position) is odd, then decrease this character by one and add the next less significant radix to the next less significant character

or (ii) If the character is even, do nothing.

One of these actions is carried out for each character position in turn, starting at the most significant character and terminating when the new least significant character has been formed. The new character word is then shifted right one place and the next step of the multiplication process begins. The least significant digit of the least significant character is shifted into W2 to become the product.

If each step of the multiplication process included, as implied above, seven serial character additions this would make the process intolerably slow. Instead, the KDF 9 uses a special block of logic, similar to carry propagation logic, to determine whether any character, which may be modified by the addition of the corresponding radix is odd. The seven additions are then performed in parallel by adding the radix word, with certain radices cleared, to the character word.

A step in the process, for 24 bit words is illustrated. Suppose that, after a certain step, the register W1, W2 contains 000010, 010101, 000000, 000011 (0)10,000, 000, 000, 000, 000, 000, 000, (W1 contains a length - 2 chains, 21 yards and 3 inches).

Register W1 (the multiplier) is compared with the radix word

$$R = \begin{matrix} 001010, & 010110, & 000011, & 001100 \end{matrix} \quad (10, 22, 3, 12)$$

$$\begin{matrix} r_0 & r_1 & r_2 & r_3 \end{matrix}$$

Note that the top radix is never to be added to the character word, thus as the top character is even r_1 must not be added. As the second from the top character is thus still odd then r_2 must be added to the third character; this makes the third character, which was even, odd, and r_3 must be added to the least significant character. Consequently, the 3 serial conditions, required by the multiplication process, may be performed in parallel by adding a modified radix word.

$$R^1 = 000000, 000000, 000011, 001100 \quad \text{in B2}$$

$$\text{Character word} = 000010, 010101, 000000, 000011 \quad \text{in B1}$$

$$\text{To give} \quad 000010, 010101, 000011, 001111 \quad \text{in W1}$$

The contents of W1, W2 may now be shifted to the right one place thus W1/2 contains

000001, 001010, 100001, 100111, (0)11,000,000,000,000,000,000

(Thus W1 contains 1 chain, 10 yards, 1 foot and 7 inches).

The underlined digits in W1, which if one merely denote that before shift the character above was odd may now be cleared, as their presence has already been detected, and this leaves the double length word W1/W2 ready for the next step in the process.

5.2.2 The Special TOB Logic.

In order to form the modified radix word, the complete radix word, stored in NB throughout TOB, is placed in B2. If necessary the relevant character position in B2 are cleared, before the character word is placed in B1 and the adder pulsed. The equipment for clearing the relevant character positions is a set of 8 6-secondary winding transformers whose primaries are gated by a set of DC logic operating on registers W1 (the character word) and NB (the radix word). The conditions generated by the logic are:-

Clear r_0 - this is unconditional

Clear r_1 - if C_0 is even i.e. if $W1(5) = 0$

Clear r_i - if C_{i-1} is even and r_{i-1} is cleared

or r_{i-1} is cleared and the result

of adding r_{i-1} to C_{i-1} is even.

A typical stage in this logic is shown in Fig.15 - note that as subsequent stages required positive levels to differentiate between this radix position being cleared and not being cleared, both these conditions must be generated.

5.3 AN EXAMPLE

The example given below illustrates the action of the FRB algorithm when converting the length 4687 inches into 5 chains 20 yards 0 feet and 7 inches. At the start of this instruction

W1 contains 000,000,000,001,001,001,001,111 (I)

W2 " 001010,010110,000011,001100 (R)

Firstly I is normalised, thus 10 left shifts are performed thus

W1 = 010,010,010,011,110,000,000,000

It is then extended to double length and the division process begins as shown below.

10	000000,000000,000000,000000	(0)10, 010, 010, 011, 110, 000,000,000
11	000000,000000,000000,000001	(0)00, 100, 100, 111, 100, 000,000,000
12	10	(0)
13	100	(0)
14	001001	(0)00, 100, 111, 100, 000, 000,000,000
15	000000,000000,000001,000110	(0)01, 001, 111, 000, 000, 000,000,000
	(Note - 2 x 9 inches =	1 foot 6 inches)
16	000000,000001,000000,000000	(0)10, 011, 110, 000, 000, 000,000,000
	(Note - 2 x 1 ft 6 ins =	1 yard)
17	000000,000010,000000,000001	(0)00, 111, 100, 000, 000, 000,000,000
18	000000,000100,000000,000010	(0)01, 111, 000, 000, 000, 000,000,000
19	000000,001000,000000,000100	(0)11, 110, 000, 000, 000, 000,000,000
20	000000,010000,000000,001001	(0)11, 100, 000, 000, 000, 000,000,000
21	000001,001010,000001,000111	(0)11, 000, 000, 000, 000, 000,000,000
	(Note - 2 x 16 yds. $9\frac{1}{8}$ inches = 1 chain 10 yds. 1 ft. and $7\frac{1}{4}$ inches)	
22	000010,010101,000000,000011	(0)10,000, 000, 000, 000, 000,000,000
	(Note - 2 x 1 chain 10 yds. 1 ft. $7\frac{1}{4}$ ins = 2 chains 21 yds. and $3\frac{1}{4}$ ins)	
23	000101,010100,000000,000111	(0)00, 000, 000, 000, 000,000,000,000
	Note - 2 x 2 chains 21 yds. $3\frac{1}{4}$ ins. - 5 chains 20 yds. and 7 ins.)	

The above example may also be used for TOB which is effectively performed using the same steps in the reverse order.

R.H. Allmon
P.P.

A.C.D. Haley.
Chief Development Engineer.
DP CSD.

J.R. Lucking
J.R. Lucking.
Computer Development.

No.	Loc. of P A	Digits Transferred	Explanation	Gating level.
1	1C22C	(W1(0-39) to B1(8-47) (0's to B1(0-7) (1's to B1(0-7)	RIGHT 8 basic shift Gated by $\overline{\text{SHL}} + \overline{\text{W1}}(0)$ Gated by $\overline{\text{SHL}} \cdot \text{W1}(0)$	W1/D
2	1C23f	W1(40-43) to BUFL		W1S
3	1C23g	W1(40-47) & W2(0) to B2(0-8)	) Double length	DLOG
4	1C23h	0 & W1(40-47) to B2(0-8)	) interconnections	DART
5	1C22d	W2(D1-43) to B2(9-47) & BUFL	RIGHT 8 basic shift	W2/D
6	1C23e	(0's to B2(0-8) (1's to B2(0-8)	Gated by $\overline{\text{W2}}(0)$ Gated by $\overline{\text{W2}}(0)$	W2S
7	1C15c	(W1(0-42) to B1(5-47) (0's to B1(0-4) (1's to B1(0-4)	RIGHT 5 basic shift Gated by $\overline{\text{SHL}} + \overline{\text{W1}}(0)$ Gated by $\overline{\text{SHL}} \cdot \text{W1}(0)$	W1/D
8	1C16d	W1(43-46) to BUFL		W1S
9	1C16g	W1(43-47) & W2(0) to B2(0-5)	) Double length	DLOG
10	1C16h	0 & W1(43-47) to B2(0-5)	) interconnections	DART
11	1C15d	W1(D1-46) to B2(6-47) & BUFL	RIGHT 5 basic shift	W2/D
12	1C16e	(0's to B2(0-5) (1's to B2(0-5)	Gated by $\overline{\text{W2}}(0)$ Gated by $\overline{\text{W2}}(0)$	W2S
13	1C13a	(W1(0-45) to B1(2-47) (0's to B1(0,1) (1's to B1(0,1)	RIGHT 2 basic shift Gated by $\overline{\text{SHL}} + \overline{\text{W1}}(0)$ Gated by $\overline{\text{SHL}} \cdot \text{W1}(0)$	W1/D
14	1C14b	W1(46,47) & WUFL to BUFL		W1S
15	1C14c	W1(46,47) & W2(0) to B2(0-2)	) Double length	DLOG
16	1C14d	0 & W1(46,47) to B2(0-2)	) interconnections	DART
17	1C13b	W2(1-47) & WUFL to B2(3-47) & BUFL	RIGHT 2 basic shift	W2/D
18	1C14a	(0's to B2(0-2) (1's to B2(0-2)	Gated by $\overline{\text{W2}}(0)$ Gated by $\overline{\text{W2}}(0)$	W2S
19	1C13c	(W1(0-46) to B1(1-47) (0 to B1(0) (1 to B1(0)	RIGHT 1 basic shift Gated by $\overline{\text{SHL}} + \overline{\text{W1}}(0)$ Gated by $\overline{\text{SHL}} \cdot \text{W1}(0)$	W1/D
20	1C14f	W1(47) & WUFL & 0 to BUFL		W1S
21	1C14g	W1(47) & W2(0) to B2(0,1)	) Double length	DLOG
22	1C14h	0 & W1(47) to B2(0,1)	) interconnections	DART
23	1C13d	W2(1-47) & WUFL & 0 to B2(2-47) & BUFL	RIGHT 1 basic shift	W2/D
24	1C14e	(0's to B2(0,1) (1's to B2(0,1)	Gated by $\overline{\text{W2}}(0)$ Gated by $\overline{\text{W2}}(0)$	W2S
25	1C21d	W1(0-47) to B1(0-47)	Basic path	W1/D
26	1C16a	WUFL & 0's to BUFL		Earth
27	1C21c	W2(0-47) to B2(0-47)	Basic path	W2/D
28	2E6b	W1(0-47) to B1(0-47)	Used by AC	-
29	1C12e	Set BUFL and ACAR from '50-bit' subtract	Floating subtraction	-
30	2E5c	W2(0-47) to B2(0-47)	Used by AC	-
31	1C15a	(W1(8-47) to B1(0-39) (Set VR if digits 0-8 of B1 are not identical - gated by SHL	LEFT 8, basic shift	W1/D
32	1C16b	WUFL & 0's to B1(40-47)		W1/S
33	1C16c	W2(0-8) to B1(40-47) & B2(0)	) Double length	DLOG
34	1C16d	W2(1-8) & 0 to B1(40-47) & B2(0)	) interconnections	DART
35	1C15b	W2(9-47) & WUFL & 0's to B2(1-47)	LEFT 8 basic shift	W2/D
?	?	0's to BUFL		Earth

B registers to W registers

No.	Loc. of P A	Digits Transferred	Explanation	Gating level.
36	1C22a	(B1(8-47) to W1(0-39) (Set VR digits 0-8 of B1 are not identical - gated by SHL	LEFT 8 basic shift	W1/S
37	1C23b	BUFL & 0's to W1(40-47)&WUFL		W1S
38	1C23c	B2(0-8) to W1(40-47)&W2(0)	} Double length	DLOG
39		B2(1-8) & 0 to W1(40-47)&W2(0)	} interconnections	DART
40	1C22b	B2(9-47)&BUFL&0's to W1(1-47) &WUFL	LEFT 8 basic shift	W2/S
41	1C11c	(B1(5-47) to W1(0-42) (Set VR if digits 0-5 of B1 are not identical - gated by SHL	LEFT 5 Basic shift	W1/S
42	1C12f	BUFL & 0's to W1(43-47)&WUFL		W1S
43	1C12g	B2(0-5) to W1(43-47)&W2(0)	} Double length	DLOG
44	1C12h	B2(1-5) & 0 to W1(43-47)&W2(0)	} interconnections	DART
45	1C11d	B2(6-47)&BUFL&0's to W2(1-47)&WUFL	LEFT 5 Basic shift	W2/S
46	1C9a	(B1(2-47) to W1(0-45) (Set VR if digits 0-2 of B1 are not identical - gated by SHL	LEFT 2 Basic shift	W1/S
47	1C10b	BUFL to W1(46,47)&WUFL		W1S
48	1C10c	B2(0-2) to W1(46,47)&W2(0)	} Double length	DLOG
49	1C10d	B2(1,2) & 0 to W2(0)	} interconnections	DART
50	1C9b	B2(3-47)&BUFL to W2(1-47)&WUFL	LEFT 2 Basic shift	W2/S
51	1C9c	(B1(1-47) to W1(0-46) (Set VR if digits 0,1 of B1 are not identical - gated by SHL	LEFT 1 Basic shift	W1/S
52	1C10f	BUFL(0-2) to W1(47) & WUFL		W1S
53	1C10g	B2(0,1) to W1(47)&W2(0)	} Double length	DLOG
54	1C10h	B2(1) & 0 to W1(47)&W2(0)	} interconnections	DART
55	1C9d	B2(2-47)&(BUFL(0-2)to W2(1-47)&WUFL		W2/D
56	1C21a	B1(0-47) to W1(0-47)	Basic path	W1/S
57	1C23a	BUFL(0,1) to WUFL		Earth
58	1C21b	B2(0-47) to W2(0-47)	Basic path	W2/S
59	1C11a	B1(0-39) to W1(8-47)	RIGHT 8 Basic shift	W1/S
60	2F5b	(0's to W1(0-8) (1's to W1(1-8)	Gated by SHL+W1(0) Gated by SHL.W1(0)	W1/S
61	1C12d	0 & B1(40-47) to W2(0-8)	} Double length	DART
62	1C12c	B1(40-47)&B2(0) to W2(0-8)	} interconnections	DLOG
63	1C12b	B1(40,41) to WUFL		W1S
64	1C11b	B2(1-41) to W2(9-47)&WUFL	RIGHT 8 Basic shift	W2/D
65	1C12a	(0's to W2(1-8) (1's to W2(1-8)	Gated by W2(0) Gated by W2(0)	W2S

K/GD.y.80 Table 1 - Second and last sheet

Each block of pulse amplifiers are connected to a transformer in the SHSM, normally each of these transformers has 6 secondaries. The bottom ends of the secondaries of these SHSM transformers are connected to the gating levels shown.

Package allocation for the Shift Paths

Special transformer packages have been designed in order to minimise the equipment required for the shift paths shown in table 1. This appendix lists the basic essentials of these packages -

K244 - SHIFT 1 (See table II)

There are 48 of these packages used in the Arithmetic Unit Cubicle. They are used in 4 groups of 12 packages providing the shift paths listed below. Each package has 4 transformers, each with 8 secondary windings providing the following shift paths

	W1 or W2 to B1 or B2	B1 or B2 to W1 or W2
L1	Left shift of 8 places	Right shift of 8 places
L2	Right shift of 1 place	Left shift of 1 place
L3	Straight transfer	Straight transfer
L4	Inverted transfer	Inverted transfer

Note that the transfer B W is not used.

K242 - SHIFT 2 (See table II)

There are 40 of these packages in the Arithmetic Unit Cubicle. They are used in 4 groups of 10 packages each group providing the shift paths listed below. Each package has 3 transformers each with 8 secondary windings.

	W1 or W2 to B1 or B2	B1 or B2 to W1 or W2
L1	Right shift of 2 places	Left shift of 2 places
L2	Right shift of 5 places	Left shift of 5 places
L3	Right shift of 8 places	Left shift of 8 places

Note that ^{NO} packages of this marque transfer into the 8 right-most digits of the W registers or into the 8 left-most digits of the B registers. These digits positions are treated specially using :-

K288 - SHIFT 3 (Drwg - S20535. See table II)

There are 4 of these packages in locations:- 2G16, 2G17, 2B18, 2B19 in the A.U. Cubicle.

Each package has 3 transformers, each with 6 secondaries. L1 and L2 are connected in parallel. The transfers provided by these packages are

	W1 or W2 to B1 or B2	B1 or B2 to W1 or W2
L1 & L2	Right shift of 2 places	Left shift of 2 places
L3	Right shift of 5 places	Left shift of 5 places

The above three boards provide all the shift paths except the special end of register effects. The end of register effects required are:-

- fill in signs at left-most end when shifting to the right
- clear unused digit positions at right-most end when shifting to the left
- Transfers involving the underflow registers
- The cross-coupling between the two single length shifting registers required for double length shifts
- Overflow detection.

These are provided on 5 more special transformer packages.

K290 - SHIFT 4 (Drwg. 20559, 20560, 20579)

There are 8 of these packages in locations:- 2D13, 14, 15, 16, 2E13, 14, 15, 16 of the A.U. cubicle.

These packages, each has 7 transformers, are used in two groups of 4. The first group perform the transfers from BUFL into the right most ends of the W registers. In this group the boards are used in pairs, one pair for W1, the other for W2; each board of the pair transfers either the 0's or 1's sides of the flip-flops. The second group copies the sign of W1 (or W2) into the left-most digits of B1 (or B2) - again the boards are used in pairs; each pair transferring either 0's or 1's. The relevant board of this group is selected by gating the primaries with conditions based on W1 (or W2)'s sign digit and the indicator SHL. If SHL is set then the board transferring zeros is selected.

TXFR	No. of Secondaries.	Shift Path	Function
L1	6	B to W 8 places LEFT W to B 8 places RIGHT	BUFL(0-3)+0's to W(D40-47)+WUFL 0's or 1's to B(D0-7)
L2	5	B to W 5 places LEFT W to B 5 places RIGHT	BUFL(0-3)+0's to W(D43-47)+WUFL 0's or 1's to B(D0-4)
L3	4	B to W 2 places LEFT W to B 2 places RIGHT	BUFL(0-3) to W(D46,47)+WUFL 0's or 1's to B(D0,1)
L4	3	B to W 1 place LEFT W to B 1 place RIGHT	BUFL(0-2) to W(D47) +WUFL 0's or 1's to B(D0)
L5	2	B to W 0 places W to B	BUFL(0,1) to WUFL (0,1) Not used
L6	2	B to W 8 places RIGHT W to B	B(D40,41) to WUFL Not used
L7	2	Spare	Used for 0's and 1's in A.U.

K285 - SHIFT 6 (Drwg. 9273)

There are 2 of these packages in locations 1D3, 1D4. in Control I.

These packages, each has 6 transformers, perform the transfers from the right-most ends of the W registers into BUFL. One of these boards performs the transfers from W1, the other from W2.

Transformer	No. of Secondaries	Used when shifting places	What it transfers
L1	8	8 places RIGHT	W(D40-43) to BUFL
L2	8	5 places RIGHT	W(D43-46) to BUFL
L3	4	2 places RIGHT	W(D46,47) to BUFL(0,1)
L4	3	1 place RIGHT	W(D47) to BUFL(0) Clear BUFL(3)
L5	2	8 places LEFT	Clears BUFL
L6	2	0 places	Clears BUFL(2&3) (Not used?)

K286 - SHIFT 7 (Drwg.20535)

There are 2 of these packages in locations 2D12 and 2E12 in AU. These packages, each has 7 transformers though L1 and L2 are connected in parallel, provide some of the cross-coupling between the two single length shifting registers to make a double length shifting register. The cross-coupling is that associated with shifts of 5 and 8 places.

Transformer	No. of Secondaries	Used when shifting double length ? places.	What it does.
L1 & L2	10 (2E12) (2D12)	W → B RIGHT logical 8 places B → W LEFT logical 8 places	W1(D44-47)&W2(D0) B2(D4-8) B2(D0-4) W1(D40-44)
L3	8	W → B RIGHT arith 8 places B → W LEFT arith 8 places	W1(D44-47) B2(D5-8) B2(D1-4) W1(D40-43)
L4	4	W → B RIGHT logical 5 places B → W LEFT logical 5 places	W1(D47)&W2(D0) B2(D4,5) B2(D0,1) W1(D43,44)
L5	2	W → B RIGHT arith 5 places B → W LEFT arith 5 places	W1(D47) B2(D5) B2(D1) W1(D43)
L6	2	W → B LEFT logical 8 places B → W RIGHT logical 8 places	W2(D8) B2(D0) B2(D0) W2(D8)
L7	1	W → B LEFT arith 8 places B → W RIGHT arith 8 places	Clear D0 of B2 Clear D0 of W2

K292 - SHIFT 8 (Drwg.20533)

There are 4 of these packages in locations 2F16, 2F17, 2C18, 2C19. in A.U.

These packages, each has 6 transformers though L1, L2 and L3, L4, are connected in parallel, together with the SHIFT 7 boards, complete the cross-coupling for shifts of 5 and 8 places and provide the cross-coupling for shifts of 1 and 2 places.

TXF	No. of Secondaries	Shift Path	Details
L1 & L2	9	W → B RIGHT 8 places B → W LEFT 8 places	W → B transfers into D0-4 of B2
L3 & L4	9	W → B RIGHT 5 places B → W LEFT 5 places	
L5	7	W → B RIGHT 2 places B → W LEFT 2 places	B → W transfers into D44-47 of W1 and D0 of W2.
L6	5	W → B RIGHT 1 place B → W LEFT 1 place	

K281 - SHIFT 9

There are two of these boards, one in location 1D2 of Control I is used to set overflow as required when shifting LEFT from the B to the W registers. The other board in location 1E10 of Control I is used to set the indicator 'ONE' on right shifts of W1 and B1.