

Sony 2500 Calculator

Theory of Operation & Summary User Manual

bhilpert / madrona.ca

2024 Oct

Rendition: 2025-Jul-13

Table of Contents

Introduction	3	Procedure Layer	21
• Architectural Overview	4	• PM - Procedure State Machine	21
Implementation of Logic	5	• Main Loop	21
Timing	6	• Zero-Blanking	21
Channel Layer	8	• Rounding	22
Register Layer	13	• CLR-ALL & CLR-ENTRY	23
• Realign-Rotate Process	13	• NML-ENTRY & PNEWN Flag	23
Digits	15	• MEM-CLR & MEM-IN	24
• DX Digit	15	• MEM-OUT	24
• DY Digit	15	• REVERSE	25
• DN Digit	16	• QUEUE-OP	25
• DM Digit	16	• ADD/SUB	25
• DF & DFp Digits	16	• MULTIPLY	26
Counters	17	• DIVIDE	27
• DPC - Decimal Point Counter	17	• SQUARE-ROOT	27
• CJ - Utility Counter	18	Supervisor Layer	31
• CK - Utility Counter	18	• SM - Supervisor State Machine	31
Flags	19	• SPC - Program Counter	31
• μ A Flag: SST, PBRD, PQPA	19	• Execution & the EXEC Flag	32
• μ B Flag: SNML, PNEG	19	• AUTO Mode	33
• μ C Flag: PNEWN2, PPHE	19	• LIST Mode	33
• μ D Flag: RDD>0, PPH2	20	• LEARN Mode	34
• μ S Flag: SPARM, SSKIP	20	• EDIT Mode	35
• μ Y Flag: DY.W, PDPE	20	• IDLE	35
• μ Ø Flag: INTØ, PQRDØ, RNDØ	20	• DIGIT & the SNML Flag	36
• RDNEG Flag	20	• MEM-SET & the SMSET Flag	36
		• JUMP & the SSKIP Flag	36
		• Non-Code Operations	37
		Appendix 1 - Table of PM States	38
		Appendix 2 - List of Operations	42
		Appendix 3 - Summary User Manual	43
		• Registers	43
		• Keys	43
		• Switches	45
		• Indicator Lamps	45
		• Programming Notes	46
		• Program Example 1	47
		• Program Example 2	48
		• Program Example 3	49
		• Program Example 4	50
		• Program Example 4a	51

Introduction

The Sony 2500 is a programmable desktop calculator from 1969. A schematic was produced from a reverse-engineering of a 2500 unit. A logic simulation was produced from the schematic. This commentary is intended as accompaniment to the reverse-engineered schematic, having been produced from analysis of the schematic and exercise of the simulation.

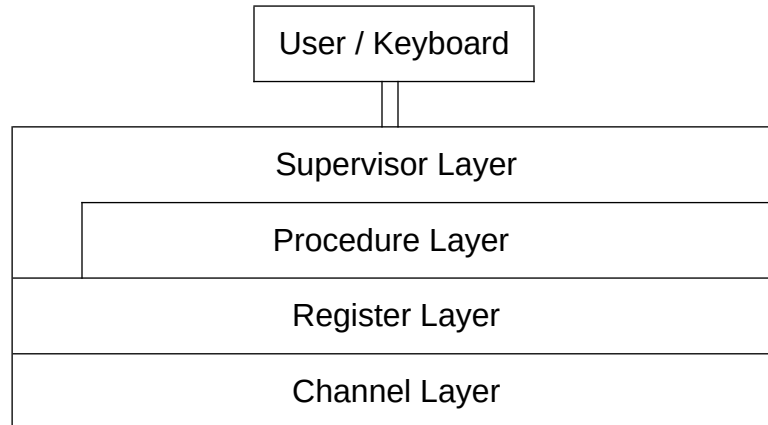
Proviso: There remain areas and aspects of the design not covered in this commentary. Descriptions are not promised to be comprehensive, e.g. the Table of PM States may not specify all actions occurring in some state.

[reference]: Text within square brackets is a reference to some page, figure or component:

[15]	page in this document
[SCH.15]	page in schematic document
[SCX.3]	page in schematic appendix document
[E216.3]	pin 3 of IC E216 in schematic

Architectural Overview

The Sony 2500 is a bit-serial design using a single magnetostrictive acoustic delay line for number and user-program storage. Conceptually, the architecture can be construed into 4 layers:



- Channel Layer At the lowest level, the Channel Layer deals with the details of multiplexing the single delay-line into a set of 15 registers. Each register as presented to the Register Layer is an independant recycling bitstream of 64 bits each.
- Register Layer The Register Layer interprets the bitstreams from the Channel Layer as 16 digits of 4 bits/digit. Some of the bitstreams are used for number storage and some for storage of user-entered programs.

For numbers, this layer performs bit-level intra-register operations such as the serial addition of bits and digit normalisation.

For program storage, this layer delivers an addressed instruction to the upper layers for execution as directed by the Supervisor Layer. A program instruction is a 4-bit function code stored in a register digit.
- Procedure Layer The Procedure Layer sees the number registers as a sequence of digits. It performs the arithmetic functions and sequences actions such as moving operands and results between registers. The state machine PM controls procedure execution.
- Supervisor Layer The Supervisor Layer deals with keyboard activity and execution of the user program. The state machine SM controls these actions. The Supervisor Layer accesses the Register layer for the user-program storage.

Electronic Implementation of Logic

The logic of the 2500 is implemented with a small family of SSI integrated circuits produced by, and proprietary to, Sony. This family is referred to here as the 500 series [SCX.8]. There are 4 types of ICs from this family used in the 2500, though there are understood to be a small number of additional ICs in the family. The 504 flip-flop IC type is further divided into 3 subtypes of -30 -40 -50. This was likely a post-production grading for speed.

The 500 series uses bipolar NPN transistors operating in current-sinking mode. V_{cc} is +4V. Logic 0 is low-impedance to GND, logic 1 is high-impedance to GND.

- Inputs Inputs are frequently left unconnected. Such inputs are seen in logic terms as 1.
- Outputs The gate outputs do not have load or pull-up resistors. The flip-flop outputs do have load resistors of $\sim 1.1K$.

Multiple gate outputs may be connected together to form a wired-AND gate (negative-logic OR).
- Flip-flops FF outputs transition on the 1-edge (positive) of the $\emptyset$ input.

The G (Gate) input controls whether the JK input state will update the outputs or not, when a 1-edge at the $\emptyset$ input occurs. If $G=1$, normal JK operation takes place. If $G=0$ the outputs will not change, i.e. equivalent to $J=0$ & $K=0$.

The flip-flops do not have asynchronous set or reset inputs. They can however, be asynchronously set and reset via collector triggering by asserting 0 at an output, e.g. asserting 0 to the $Q\sim$ output sets the FF to $Q=1$.

Timing

The system timing is tied tightly to the operation of the acoustic delay-line providing storage. The delay-line presents a requirement of precise data timing to maintain synchronism with any drift in latency. In other delay-line implementations this may be provided by a crystal-controlled master clock. The 2500 takes an alternative approach. The master clock is an RC astable flip-flop functioning as the VCO in a phase-locked-loop arrangement with a phase comparator and the delay-line in the loop. This is discussed further in the Channel Layer section. For following commentary refer to the Timing Diagram [SCX.1].

- **Master Clock** The master clock is the astable flip-flop formed from QF13,18. QF19 is a local voltage regulator for the master clock. The output from this astable is buffered to produce the master clock signal ØS.

The frequency of the astable can be adjusted by varying the QF13,18 base bias via Darlington pair QF16,17. This functions then, as the voltage-control input for the PLL VCO.
- **Channel Slicing** ØS sets the raw bit rate for writing-to and reading-from the delay line. The delay line is time-division multiplexed into 16 channels. The channels are delineated in time by the ØS Counter (ØSC) and identified as ØS0::15. ØSC is a straightforward 16-stage 1-of-16 ring counter.
- **ØB** Selected outputs from ØSC feed flip-flop [F408.9] to produce a signal ØB which has a cycle period equal to a full 16-state cycle of ØSC. The duty cycle of ØB is designed in conjunction with the register allocation of the delay line channels (see RS cycling).

ØB is the register-level bit clock for the serial processing of numbers, with the 1-edge delineating these bits. At the register layer and above, all flip-flops are clocked by ØB. The G input to the FFs is then used - if necessary - to control the bit periods at which transitions occur.
- **Bits of Digits** A digit is comprised of 4 bits. The bits of a digit are delineated in time by the ØB Counter (ØBC). ØBC is a 2-stage switch-tail ring counter. The bits of a digit are identified as ØB0,1,2,3 corresponding to bit weights 1,2,4,W, though not all need to be decoded from ØBC to actual signals.
- **Digits of Numbers** A number is comprised of 16 digits. The digits are delineated by the ØD Counter (ØDC). ØDC is an SR16-type counter. By default it produces the full 16-state SR16 sequence. The ØDC state is decoded to signals ØD0::15 with ØD0 corresponding to digit 0 (LSD) of a number.

ØDC provides a simple example of the use of the G inputs of the flip-flops. The G inputs are fed by ØB3, resulting in the JK state updating the flip-flops on the 1-edge of ØB at the end of ØB3. The counter transitions thus occur at the end of digit periods.
- **Extended NC** Assertion of PNCEXT (Number Cycle EXTend) extends the ØDC sequence to 17 states by inserting an additional digit period (ØDEXT) between digits ØD11 & ØD12. This is used when the digit registers DX,DY,DN are inserted into the register bit-loops during processing.

- Numbers A full 16- or 17-digit cycle of ØDC then, constitutes a number cycle. Signal ØEON (End of Number) is a pulse produced during the last bit of a number cycle (ØB3 of ØD15).
- State Transitions State cycles for the execution state machines SM & PM are synonymous with number cycles.

The G inputs of the SM & PM state-register flip-flops (SM.*n*,PM.*n*) are fed by ØEON, with ØB feeding the Ø input. SM & PM state transitions thus take place on the 1-edge of ØB at the end of ØB3 of ØD15.

Channel Layer

The channel layer provides the storage for numbers and user programs.

The channel layer is comprised of the acoustic delay line (ADL), its read and write amplifiers, the clock synchronisation circuitry, the flip-flops DLRCV, DLa, DLap, DLb, and a set of flip-flops and gates for multiplexing and demultiplexing the delay line into bitstreams to present to the register layer. While the ADL provides the vast majority of the storage, the delay-line in full includes storage provided by some of the ancillary flip-flops [SCX.4].

- **Physical** The ADL is a magnetostrictive acoustic delay-line operated in the longitudinal mode. The wire forming the storage medium is ~ 362 cm long, looped into a coil of 10 turns. At each end is a transducer solenoid, one for write and one for read.
- **Channels** As mentioned in the timing commentary, the delay-line is multiplexed into 16 channels. 15 of the channels become a register each. The remaining channel is injected with a fixed pattern to provide a sync signal to the phase comparator of the master clock PLL. The time period for a channel will be referred to as a slice and are identified as ØS0::15.
- **Storage** Each channel provides 64 bits of storage. Interpreted as a register the 64 bits are 16 digits of 4 bits each. For 16 registers, the full delay-line loop needs to be $1024 = 16 \times 64$ bits [SCX.4]. Four of the 1024 bits are provided outside the ADL. DLa with DLap provide one bit of the four, DLc is one, the remaining two are the multiple flip-flops of the demultiplexed registers where a bit resides for 2 slice periods before being written back to the ADL. The ADL then, must store 1020 bits.
- **ADL bit injection** Bits are injected into the ADL on the 0-edge of ØS. This is the middle of a channel-slice period when the state of channel-bits is stable in flip-flop storage.

The 0-edge of ØS triggers the injection monostable flip-flop [QF21,27] to produce a pulse of ~ 500nS. This pulse is gated by the 1-state of a multiplexed channel-bit ADLINJ [F310.6] to drive current in the ADL write solenoid [QF30].
- **ADL bit exit** The degenerated pulses from the ADL read solenoid are first AC amplified [2 in ADL,QF2,4], then re-formed into binary pulses of again ~ 500nS [QF5,7], to appear as signal ADLRD.
- **ADL sync** For a given latency in the ADL, the frequency of the master clock must be such that bits are injected into the delay line at a rate whereby 1020 bits will reside between the write and read solenoids.

Channel 0 of the delay line is used to carry sync marker bits. These markers are injected regularly into the channel [F412.3] from the timing system, they are not recycled per se. Re-formed pulses from the ADL are gated with timing signals to extract the anticipated sync markers. The sync markers are fed to a phase comparator for comparison with a signal from the timing counters. Variations in time between the markers and the timing system signal adjusts a charge pump which maintains the

voltage level into the master-clock frequency-control input.

- Charge pump F306.3, F306.5, Q14,15 and associated components form the charge pump.

Capacitor CF24 is the charge storage. Resistors RF44,45,48 present a determined discharge rate for CF24. CF24 is charged by asserting 1 at either of the two control inputs driven by TNUPC and TNUPF (Tune UP Course & Fine). TNUPC=1 turns on QF14 to charge CF24 quickly through the low-R emitter resistor. TNUPF=1 turns on QF15 but the charge rate is slower due to the much higher R of RF48.

The voltage on CF24 (TNV) feeds the frequency control input to the master clock. The high-gain Darlington-pair QF16,QF17 reduces the load on CF24 so the discharge rate will be in better accordance with the engineered rate set by RF44,45,48.

- Achieving sync In normal operation, the phase comparison is made at slice-period resolution while updated at a per-digit rate. At power-on, the initial master-clock frequency will be some distance from the needed target, requiring a wider capture range to bring the frequency closer to target. The frequency must be resolved to the precision of a particular slice in the full number cycle.

There are two phase comparators that resolve the synchronisation to different levels of precision: a coarse comparator (TNUPC) that will be used to bring sync into number-cycle precision, and a fine comparator (TNUPF) that will be used to then bring sync into channel-slice precision and maintain it there.

In the following commentary, functioning of the TNUPF comparator during normal, post-initialisation operation will be described first, followed by the TNUPC comparator, and then the process of achieving sync during initialisation. Reference also the phase comparator timing diagrams [SCX.2].

- Fine tuning The TNUPF flip-flop [QF9,12] is triggered at a per-digit rate but its duty cycle will vary with the error of the master-clock from the ØS target frequency. The target is not a fixed frequency, rather it is dependant on the delay period of the ADL which may vary.

During ØB3, TNUPF is set 1 at the beginning of ØS10 [F308.6] and cleared 0 by an extracted sync marker from the ADL [F308.5]. The lower the ØS frequency, the longer the TNUPF 1-period, thus a lower ØS frequency charges up CF24 more, raising the ØS frequency.

The sync marker pulses are expected during ØS14 [F308.3]. The sync markers were injected during ØS2 of ØB0 [F412.3]. ØS14 of ØB3 is 4 slices earlier than this to compensate for the 4 bits of the delay line in flip-flops, outside the ADL. The sync markers are isolated further into the 2nd half of ØS [F308.11] – this is significant in the resolution precision of the loop operation.

The loop response is slower than the per-digit update rate. In practice,

the ØS10-to-ØS14 charge period is not long enough to fully compensate for discharge. In observation of unit 620175, the ØS frequency slowly lowers over ~ 11 digit periods till the 2nd half of the ØS14 period slips beyond the sync markers and the markers are in the 1st half. TNUPF is then no longer cleared 0, instead continuing 1 through the digit period, this condition persisting for ~ 8 digit periods when the frequency is high enough to bring the 2nd half of ØS14 back to the markers. This correction cycle repeats with some jitter of ~ ±1 digit [SCX.2].

The net result is the ØS frequency is maintained such that the 0-edge of the re-formed read pulses from the ADL are synchronous with the 0-edge of ØS. The ADL delay from injected pulse to re-formed pulse is:

$$\text{delay(ADL)} = 1020 \cdot \text{period}(\text{ØS}) - 500\text{nS}$$

The 500nS being due to the injected pulse starting on the 0-edge of ØS whereas the re-formed pulse ends on the 0-edge of ØS.

- Coarse tuning

The coarse phase comparator TNUPC is formed from two flip-flops [F305].

F305.9=1 for the period of ØS11 to the end of ØS0 during EON. If a sync marker is received outside of this period, TNUPCp asserts 1 to enable fast charging of CF24 to bring up TNV and raise the frequency of ØS. Eventually the F305.9 1-period coincides with the sync marker and TNUPCp is cleared 0 to stop the fast charging.

- Sync init process

There are two RC power-on-reset circuits: those of CF4 & CF34, with CF34 being the longer. Initialisation proceeds in several stages [SCX.2]. Note TNRDY, TNUPCp and TNUPC are electrically the same signal.

0. Power up: CF34 is not charged; once power and basic circuit operation has stabilised, INIT=1. CF4 is not charged, so TNRDY=0, suppressing coarse tuning of CF24. Charge pump capacitor CF24 is not charged and the ØS frequency starts out being low. Sync markers are injected once every number cycle [F412.6,1].
1. EON capture: As CF4 charges, TNRDY eventually asserts 1, enabling the coarse-tuning phase comparator [F305.3] to assert TNUPC=1 till the EON sub-pulse [F305.9] coincides with the sync markers. Once this is achieved, this phase comparator is essentially out-of-the-picture and the fine-tuning phase comparator takes over.
2. Channel capture: As INIT is still 1, sync marker gating into the fine-tuning phase comparator is enabled to select sync markers during ØS14 of only EON [F307.5,6]. The ØS frequency is now slowly raised further to pull ØS14 to coincide with the sync markers.
3. Normal operation: Once CF34 charges sufficiently, INIT goes 0, and the sync marker injection and gating switches to the per-digit rate of normal operation, as described previously. The

coarse-tuning phase comparator is also formally disabled [F305.2].

- Power-off

When power is turned off, the charge in a 10µF capacitor charged to 100V is dumped in part into CF38 to turn on QF20. This discharges both CF24 and CF34 so a quick re-power-on is ensured to perform the full initialisation cycle. CF4 is discharged by a reverse diode to V+4.

The 10µF@100V charge is also distributed to dump through the ADL delay-medium wire. This is curious but may have something to do with magnetising the wire to enhance or maintain the magnetostrictive characteristic.

- ADL bit recovery

The DLRCV flip-flop is the initial receiving flip-flop of read pulses from the ADL. DLRCV is not an additive bit in the delay loop though, rather, it overlaps in time with the last bit of the ADL. DLRCV is cleared 0 at the beginning of a slice period, the appropriate bit pulse from the ADL then has a window of (less than) one slice period to exit the ADL and set DLRCV=1 before the bit is clocked out of DLRCV to DL_a.

DL_a is gated by DLRCV to toggle when DLRCV=1. DL_{ap} holds the prior state of DL_a and the two feed an XOR gate [F303.3,6;F304.1]. The output of the XOR is 1 if DL_a toggled. The net storage or delay is only one bit after DLRCV – DL_{ap} is not additive to the delay. It is not clear why this toggle-with-transition-detector was used to transform the 1s from DLRCV to a transition and then back to 1s – perhaps to avoid a differential feed from the capacitor-coupled DLRCV flip-flop.

- Channel allocation

The 16 channels of the delay line are allocated per the table following. The channel identification corresponds to the slice period during which bits of the associated register appear in flip-flop DL_b – the last delay line stage prior to demultiplexing. Included are the DM and SP_r values which will select the RS registers.

Delay-Line Channel Allocation								
Channel ØS	Use	DM	SPr		Channel ØS	Use	DM	SPr
0	Sync				8	RM7 RP5	0xD	0x0
1	RA				9	RM5 RP7	0xB	0x2
2	RD				10	RM6 RP6	0xC	0x4
3	RB				11	RP2		0x6
4	RM3	0x3			12	RP3		0x3
5	RM4	0x4			13	RP1		0x5
6	RM1	0x1			14	RP4		0x1
7	RM2	0x2			15	RT		

- Multiplexing

To present the channels as registers they are demultiplexed into the set

of flip-flops RD,A,B,T,S. The recovered channel bits are distributed into these flip-flops as they exit DLb. After two slice periods they are recycled back into the delay-line, e.g. note the two-slice difference between F405.11 and F412.10.

The bits none-the-less persist in the demultiplex flip-flops for longer, through the 1-edge of ØB. This is the transition or processing edge at the register level, thus the bits in these flip-flops are simultaneously available for inter-operation at the register level, such as summing two of them.

- RT logic differs The logic for multiplexing RT differs from the other registers [F404.4,F408.4]. [F404.4] is loaded at the same time as DLb. [F408.4] is collector-set by level from RTw rather than by edge. It is not clear why this was done, perhaps to avoid a clocked transition for RT channel-bits which would otherwise occur on the transition edge for register-bits, or due to the logic being too slow for the half-slice preparation period before the bit is written to the ADL.
- RS registers 11 channels are allocated to the 7 user memories (RMn) and user program storage (RPn). Collectively, they are referred to as the RS registers.

These registers share the bitstream RS. Only one of these registers can be accessed at a time. The selection is determined by the ØSn signal fed to [E314.2]. The bit stream of the currently selected RS register appears as signal RS.
- RS cycling Normally, all the RS registers - including that selected - loop internally in the channel layer through flip-flops RSc & RSd [F407.9,F407.4]. These two flip-flops provide the 2 bits of storage to complete the data-loop length of these registers. A gate in this path is enabled to pass data during the 0 period of ØB [F410.12]. The duty cycle of ØB has been chosen such that it is 0 for the write-periods of the RS register channels and 1 for the write-period of the others [F408.7,8], thus the others are prohibited from cycling via this path.
- RS write To write to an RS register it must be selected, the desired bit stream fed to RSw, and RSwENB asserted 1.

RSwENB=1 enables flip-flops [F409.9,F409.4]. These flip-flops delay the selected slice phase 2 slice-periods. For the delayed slice phase, the normal RS recycle path is disabled [F410.11] and a data bit from RSw enabled for injection to the ADL [F410.10].

Register Layer

The objective of the register layer is to provide to the procedure layer an interface that deals with multi-digit decimal numbers. The task of the register layer then, is to perform the operations internal to registers: arithmetic operations at the bit level, digit normalisation, digit shifting, transferring register contents between registers, etc.

The channel layer provides 15 64-bit registers to the register layer. Four registers - RD,A,B,T - are presented as individual bit-streams. The remaining 11, referred to as the RS registers, share a bit-stream as RS.

- Register RD RD is the displayed number, be it an operand being entered or a result.
- Register RA RA is an operand and results register.
- Register RB Between operations RB contains a prior operand; during operations RB is used as an internal temporary.
- Register RT With tally mode enabled, the tally sum accumulates in RT.
- Registers RS 4 of the RS registers are dedicated to M1,2,3,4; 4 are dedicated to program storage; and 3 may be either memory or program storage. Long programs will abrogate M5,6,7 in descending order.
- Digit encoding The 4-bit digits of a number use a 1242 bit-weight decimal encoding, not standard BCD [SCX.7]. The 2nd '2' bit will be referred to as the 'W' bit to distinguish them.

As with BCD encoding, the binary sum of two 1242 digits requires correction (normalisation) to the 1242 encoding. This is performed in DX.
- Negative numbers For registers RA,B,T,Mn, the stored value is a signed number in 9's-complement format. The 16th digit (ØD15) will be 9 for negative and 0 for positive. Using 9's-complement format results in there being both a positive zero and a negative zero, this is consequential for user programming (see JUMP instruction).

For RD, the stored value remains as a positive magnitude but with flag RDNEG indicating the sign (negative=1). When needed for arithmetic, the RD bitstream is inverted or not, in accordance with the sign, by an XNOR gate [C414.5,C413.5,6] to become bitstream RDCOMP.

Realign-Rotate Process

The register layer is not entirely free of constraints from the channel layer. The bits presented to the register layer are not consistent over the entire ØB period. However, they are guaranteed to be aligned and stable across the 1-edge of ØB when the register layer bit transitions occur.

- Conundrum As mentioned in the channel layer commentary, an operation involving two registers can be performed at the 1-edge of ØB where the register bits are mutually stable.

In storing the result of such an inter-operation however, a problem arises. The current register-bit period has now ended, so injecting a result value into the register channel will place it in a subsequent bit period where it will be out of alignment with the other registers.

- Realignment

Resolving this involves a process:

- The bit-loop for up to 3 registers is extended from 16 to 17 digits by inserting a digit register (DX,DY,DN) into their path.
- The number cycle is extended to 17 digits by assertion of PNCEXT. PNCEXT is a subset of the states of PM.
- Now, during a number cycle, for the extended registers, bits are written to the delay line after the additional digit delay of DX,DY,DN. At the end of the number cycle all is well for these registers: though they were written into a later absolute location in the delay line, due to the extended number cycle the digit timing periods were also delayed, so the data and the timing remain in alignment.
- Unfortunately, the registers which were not extended are now out of alignment with the extended registers and the timing: they have effectively been rotated down one digit. The solution to this is to keep doing it: the register and number cycle extension is maintained for enough number cycles to total 16, at the completion of which the out-of-alignment registers will have rotated into alignment with the extended registers. The extensions can now be removed.

- Tracking with CK

The extended number cycles are tracked by counter CK. Before PM is released to execute a procedure, CK is sent home during S1 [B307.5]. During the procedure execution, for each extended number cycle, CK is incremented [B206.9]. Before a PM procedure ends, state P03 or P04 is entered. This state loops until CK returns to home [D316.10], to complete 16 extended cycles.

- Realign example

An example of this is the MEM-IN procedure [SCX.5]. In P02, RD is added to RS. RS is extended with DX, RD is extended with DN and the number cycle extended, for one number cycle through P02. After P02, P03 is executed in a loop for 15 number cycles with the extensions continued.

Digits

There are six 4-bit shift registers used for intermediate storage of digits. 'Digit' here is 4-bits, but not just digits of a number. Functions are also encoded and stored in 4-bit digits.

The digits are DX, DY, DN, DM, DF & DFp. DX, DY & DN may be inserted in number-register bit-loops for extension to 17 digits but they additionally provide facilities distinct to each.

DX Digit

DX is the arithmetic adder, and is an extension digit, usually for RA. It has two bit-stream inputs for the operands, a bit-stream sum output and a carry output. DX adds the two input bitstreams and performs sum correction for the 124W encoding.

- Adder The adder is formed by the gates between the A.W, B.W and X.4 flip-flops. The operand half-adder (XOR) is formed from [B215.6,B216.3, B212.5]. The carry half-adder is [213.6,212.6,wired-NORN].
- Normalisation In processing the operand bitstreams, the adder performs a binary addition of two 124W-format digits from the DXa and DXb inputs. The sum however, is not in 124W format, so must be corrected - normalised - to 124W.

The scheme for doing this normalisation is to wait till the end of the digit period when the lower 3 bits of the binary sum have been formed and stored in the DX flip-flops (the sum W bit has not yet been formed due to the adder being located between the 3rd and 4th bits). In the next bit-period (ØB0), this state is assessed by combinatorial logic and appropriate corrections applied to the bits of the digit on the shift at the end of ØB0. [B112.1,B212.11,B214.11].

DY Digit

DY can increment a digit, and is an extension digit, usually for RB. The increment facility is used in multiplication for reducing the multiplier digits, and in division and square-root for building quotient digits.

- Increment The flip-flop [C103.4] functions as a carry flag. Incrementing the digit value in DY starts with setting this carry flag for the first bit of the digit [C205.9,11]. The carry is added to the bitstream in the XOR formed with [C203.3,5,etc.]. The carry persists until the operand bit from the bitstream is 0, at which point the carry is cleared [C103.6].
- Normalisation As the DY data value is 124W encoded, normalisation of the sum as in DX is also required here. However, because the sum is following a known sequence - to wit, that of incrementing - a simplified normalisation scheme can be employed: the only situation where normalisation is necessary is when 4 is incremented to 5 - the 0b0101 sum must be corrected to 0b1011.

This correction is accomplished by suppressing the bit shift of the 3 lower bits of DY for the first bit of the subsequent digit period [C202.3]. 1 goes out for the 1-bit, while the 0b101 4,2,1 bit sequence is effectively

shifted up 1 bit to the W,4,2 positions. The net result is 0b1011 is shifted out as the sum.

DN Digit

DN holds an entered numeral, and is an extension digit, usually for RD. It may be loaded in parallel from the keyboard or serially from program storage.

DM Digit

DM stores a 124W-format decimal numeral received either from the keyboard or program store to indicate the currently-selected memory register.

DF & DFp Digits

- DF DF holds a code for the function being performed. Like DN, It may be loaded in parallel from the keyboard or serially from program storage.
- DFp DFp contains the function code of the most recent prior function executed.
- Function Codes Function codes are a 4-bit hex value [SCX.7].

Counters

SR16-Type Counters

There are five 4-bit counters in the system:

- ØDC Timing Digit Counter
- DPC Decimal Point Counter
- CJ Multi-use utility counter
- CK Multi-use utility counter
- SPd Program Counter - digit portion

All these counters are formed as a shift register ring with feedback which - unmodified - produces a full 16-state sequence as shown in the table below. These will be referred to as SR16-type counters. The values in the table derive from bit weights 8,4,2,1 applied first-to-last to the shift register flip-flops. The table shows the count arbitrarily starting with value 0.

SR16-type Counter Sequence

Count:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value:	0	8	4	A	5	2	9	C	6	B	D	E	F	7	3	1

Note these are not common switch-tail ring counters. The point of using the SR16 design was presumably that it is synchronous, provides full state-utilisation of the flip-flops, but requires far less state logic than a 4-bit synchronous binary counter.

- Home state None of these counters have an explicit reset input. Instead, for those necessary, they are gated to clock to some home value and then stopped. This home state may be deemed to be the beginning of the count cycle. The home value is not the same for all the counters - the reason for the lack of consistency being unapparent.

DPC - Decimal Point Counter

DPC maintains the user-selected position of the decimal point in the display.

- Sequence The SR16 sequence is modified to 15 states:
 [3 1 0 8 4 A 5 2 9 C 6 B D E 7]
 0xF is absent from the sequence.
- Home state When DPC=3 the output DP is asserted, activating the DP cathodes of the Nixies.
- Operation DPC is continuously active, incrementing at each digit transition except ØD15 [B206.3]. As a 15-state counter it thus maintains synchronism with the 16-digit number-cycle. For example, if DPC=3 during ØD5, the DP of the 5th LSD Nixie is illuminated. As the Nixies are right-side DP, there are then 4 fractional digits in the display.
- Initial At power-on, DPC is set to value 6 at the beginning of number cycles

(ØB0 of ØD0) by the pulses of INITP. This determines the initial startup position of the DP and corresponds to 4 fractional digits.

- DP shift down To shift the selected DP position down, DPC will receive an extra increment during a count loop. In subsequent loop cycles then, DPC=3 will occur one digit earlier. The additional increment is performed at ØD15 [B205.3].
- DP shift up To shift the selected DP position up, one increment of DPC will be inhibited during a count loop. Subsequently, DPC=3 will occur one digit later. The increment at ØD0 is inhibited [B306.3].
- Thousands marks With thousands input DPCTH=1, DP=1 when DPC=8,5,C,D as well as when DPC=3.
- Units digit An additional output, UNITØD, asserts when DPC=7, which is one digit period prior to DPC=3. This marks the digit period when the units digit is being processed (is output from the register bit-streams).

CJ - Utility Counter

Multi-use utility counter.

- Sequence CJ follows the full SR16 count sequence.
[3 1 0 8 4 A 5 2 9 C 6 B D E F 7]
- Home state In states PM=0bx0xx0, CJ is set to 3: CJ is incremented till stopped by CJ=3 [B301.6].
- Serial load With SM.1=1, CJ is serially loaded from RPF [B403.10]. The counter feedback is suppressed during the load.
- Uses Number-entry, zero-blanking, CHECK mode function display, procedure loop counter.

CK - Utility Counter

Multi-use utility counter.

- Sequence CK follows the full SR16 count sequence:
[F 7 3 1 0 8 4 A 5 2 9 C 6 B D E]
- Home state During S1, CK is set to 0xF.
B209.6=1 when CK≠0xF. With SM.1=1 and B209.6=1, CK is incremented on each digit [B208.6]. When CK=0xF, B209.6 is set 0, disabling the incrementing.
- Output DPCTH DPCTH=1 when CK=0xE or CK=0xF.
- Uses Thousands marks, register realign-rotate.

Flags

There are a dozen flip-flops in the design serving as binary flags.

A number of these flags are as individuals shared for multiple uses where the uses do not overlap in time. These flags are given a base name of μ a along with additional names descriptive of the specific uses.

For flags with a single use, most are not mentioned in this section, rather being described in context in an according section.

μ A Flag: SST, PBRD, PQPA

- SST Supervisor Start. See EXEC Flag [32].
- PBRD Blank Rounding Digits. See Zero-Blanking [21].
- PQPA Square-root Pair-Aligned. See SQUARE-ROOT [27].

μ B Flag: SNML, PNEG

- SNML See DIGIT [36].
- PNEG During arithmetic procedures PNEG tracks the sign of the result. It is manipulated by the following:
 - Operand sign capture: During P0A, RA is assessed and PNEG set 1 if RA is negative [D208.5], as indicated by the MSB.
 - Result sign capture: During P06, RA is assessed for negative [D205.5,6]. See ADD/SUBTRACT [25].
 - Result sign tracking: During P16, PNEG is toggled if RD is negative [D115.6].

μ C Flag: PNEWN2, PPHE

- PNEWN2 Operand Management. For an operation which produces a semantically new number, this flag carries indication over the idle period to then set PNEWN.

PNEWN2 is cleared 0 in P17 at completion of an arithmetic operation [D110.7,9].

PNEWN2 is set 1 at the end of P0A during a recall operation (MEM-OUT, KTREAD, REVERSE) [D109.8,D108.11].

PNEWN2 is set 1 during intermediate binary arithmetic operations (not completed by the '=' key) [D109.3].

At the beginning of P0A, if PNEWN2=1 then PNEWN is set 1 [D111.6,D108.1], that is, PNEWN picks up the PNEWN2=1 status from the previous procedure.
- PPHE Phase End. See SQUARE-ROOT [27].

μD Flag: RDD>0, PPH2

- RDD>0 RD Digit is greater than 0. Set 1 at the end of a digit period if the digit which has just exited RD is not 0. An illuminated zero (0b1000) is recognised as 0.
- PPH2 Phase 2. See SQUARE-ROOT [27].

μS Flag: SPARM, SSKIP

- SPARM Program Parameter. When in LEARN Mode, SPARM=1 indicates a STOP instruction for parameter entry needs to be entered in the program store. See LEARN Mode [34].
- SSKIP Skip Instructions. During program execution (AUTO Mode), SSKIP=1 indicates instructions are to be skipped, as opposed to performed (SSKIP=0). See JUMP [36].

μY Flag: DY.W, PDPE

DY is not needed during number entry, allowing the W-bit flip-flop of DY to be employed in secondary service as a flag:

- PDPE Decimal Point Entered. Indicates the DP key has been pressed and numeral entry is now in the fractional digits. See NML-ENTRY [23].

μØ Flag: INTØ, PQRDØ, (RNDØ)

This flag will assert for some portion of a number cycle. What portion is dependant on the PM procedure.

- INTØ During P09,P08 & P18, asserts 1 for the integral digit periods.
- PQRDØ Asserts 1 during the digit periods of the radicand in RB [B205.10]. See SQUARE-ROOT [27].
- RNDØ During P14, RNDØ asserts 1 for the digit periods from the rounding digit and up. See Rounding [22].

RDNEG Flag

RDNEG is a sign flag for RD. RDNEG=1 indicates RD is to be treated as a negative number, even though it is not in complement form.

- CS key Pressing the CS key toggles RDNEG [D405.7,D207.8].
- Copy from PNEG During P13 and P17, RDNEG is copied from the state of PNEG [D408.6,D405.5].
- Subtract Subtraction operations toggle RDNEG [D104.5].

Procedure Layer

PM - Procedure State Machine

The Procedure State Machine sequences procedure execution. The current state is stored in binary in a register of 5 flip-flops. 25 of the 32 possible states are used. State values are presented as Pxx where xx is a 2-digit hex value.

The decoding of PM states from binary is heavily optimised for necessity, with the consequence of being frequently obscuring of intention. For example, at some point in a procedure the algorithmic intention of a single state may be decoded by only 3 bits, leaving 2 bits of freedom and 4 possible states, but is acceptable because other algorithmic circumstance excludes 3 of the 4 states from occurring at this point or are unused states.

State transition logic determines which state-register bits are to be cleared or set to get from the current state to a target state, though to note, the toggle facility of the JK flop-flops is used at times.

For all commentary on PM, see the Table of PM States [38] and the PM State Graph [SCX.6].

Main Loop

The various procedures are situated in a main execution loop. P09 is a wait state for PM - when inactive PM loops in P09. PM is kicked out of P09-wait under 3 circumstances:

- at the end of S1. As SM leaves S1 to head to S0, it kicks PM to either P08 [E408.6], or P0A [E408.6,E408.3].
- by assertion of KSCS (DP-Shift or Change-Sign). PM heads to P00 [D416.7,D209.2].
- by assertion of KCA or KCE (Clear operations). PM heads to P00 [D416.9,D209.1].

P0A unblanks RD so it is ready for use during a function procedure.

At the end of a function procedure, the realign-rotate process is performed by P03 or P04. If exiting from P04, a dummy pass through P00 is performed. This pass is not necessary and is merely a consequence of minimising the state-transition logic. Execution proceeds to zero-blanking of RD, and PM enters P09 to wait for another kick from SM. DF is cleared of the function code (DF=0) in P01, or P00.

Zero-blanking

Blanking of leading zeroes in the display is accomplished by modifying zero digits in RD for the period spent in wait-and-display state P09. Zeroes which are to be illuminated are modified to value 0b1000 (not a value in the 124W decimal encoding set) while blanked zeroes remain as value 0.

The zero-blanking modification of RD takes place in the main execution loop after the completion of function procedures, prior to going to P09. In consequence, RD must also be unblanked prior to function procedure execution after leaving P09.

The zero-blanking algorithm is based on the equation:

$$16 \text{ digits} = z(\# \text{ of integral digits} = 0) + n(\# \text{ of integral digits} > 0) + f(\# \text{ of fractional digits})$$

The blanking and unblanking involves 3 states:

- P01 RD is assessed in preparation for zero-blanking.
CJ has been sent home in some earlier state (P00,02,04). As RD is scanned during the number cycle, CJ is incremented for integral digits=0 [B401.5]. At the end of P01, CJ is semantically z in the equation above. The count of further increments needed to cycle CJ to home is n+f.
- P11 The zero-digit conversions in RD are performed. The conversion of digits=0 from blanked to unblanked (0b1000,illuminated) will be performed by setting the W bit of the digit as RD rotates through the RD gate [C415.11,C109.5].

As the number cycle begins, the first consideration is blanking of rounding guard digits. Initially, flag PBRD=1, indicating that digits are to be blanked and so suppressing their conversion to 0b1000 [C109.4]. When the end of the guard digits or the units digit is reached, PBRD is cleared 0 [B204.5,B204.3].

Also initially in the number cycle, CJ is incremented for each of the fractional digits (f) [B302.6]. Once in the integral digits however, the increment is inhibited for digits=0 [B403.6], that is, only counts digits>0 (+n).

When CJ is home again (CJHOME=1) blanking begins again: the conversion of digits=0 to 0b1000 is suppressed [C109.3]. At this point, the remaining digits will all be 0, so the incrementing of CJ continues to be inhibited and CJHOME remains asserted.

Example: z=7 n=5 f=4 ø=0b1000
start of P01: RD= 0000 0012 3045 . 0600 CJ=3 (HOME 0)
end of P01: RD= 0000 0012 3045 . 0600 CJ=2 (7)
end of P11: RD= 0000 0012 3ø45 . ø6øø CJ=3 (HOME 16/0)
- P0A RD is unblanked. As RD is scanned during the number cycle, digits=0b1000 are converted to 0 [C107.5].

Rounding

Rounding is performed in P14. The result in RB will be copied to RA with rounding performed on the bitstream moving from RB thru RX to RA. RB is looping through RY.

- RNDØ During P14, RNDØ=1 will be asserted for the period of the number cycle from the rounding-digit KRØD to the end. At entry to P14, flag µØ=0. At KRØD, RNDØ asserts 1 [B305.4]. On the next ØB, µØ is set 1 [B205.6] and thence sustains RNDØ for the remainder of the number cycle [B305.10].
- Truncate The RB bitstream is first truncated of the LSD digits below the rounding digit on the way into DX [B312.6].
- Round up: If the digit prior to the rounding digit is not zero ((DY=0~)=1) the rounding digit is incremented by setting the carry flag [B313.5] by gating

for one bit at the beginning of the digit period. The single bit occurs when RNDØ is 1 but µØ has not yet been set [B312.4,1].

- Round off (5/4): If the prior digit is ≥ 5 (DY.W=1) the current digit is incremented [B312.5]. Round-up incrementing is suppressed [KR54~ to B313.4].
- Round down: The truncation is sufficient for the intention. Other rounding actions are suppressed [B312.3,B313.12].
- P15 P15 is apparently a null state.

CLR-ALL & CLR-ENTRY

The clear operations direct PM out of wait into P00 during which the keystate forces assorted actions.

- CLR-ENTRY PFCE~ is asserted 0. The following actions are performed:
 - EXEC is cleared [E203.2],
 - SM is taken to S0,
 - SSKIP, SMSET are cleared,
 - PM is taken to P00,
 - OVF is cleared,
 - DF is cleared in P00,
 - DN is cleared,
 - RD is cleared,
 - RDNEG is cleared.
- CLR-ALL KCA~=0 pulses KCE, so the actions for CLR-ENTRY are performed, and in addition:
 - RA is cleared by suppressing normal recirculation [C212.1],
 - RB is cleared by suppressing normal recirculation [C214.2],
 - DFp is cleared by enabling shifting so DFp will loaded from the cleared DF [D406.11],
 - SPC is reset to program start [D411.9].

NML-ENTRY & the PNEWN Flag

The NML-ENTRY procedure is invoked by either a numeral keypress or the D instruction.

Elements in use:

- PNEWN flag The PNEWN (NEW Number) flag indicates whether RD contains a new operand in contrast to a previous result.
PNEWN is cleared 0 in P11 prior to P09-wait [D108.6], and set 1 in P18 [D111.10] when the first numeral of a new number is entered.
- PDPE flag Pressing the DP key performs an immediate collector-set 1 of flag PDPE (DP Entered) [C101.10]. No other action is taken.
- CJ counter During the entering of a new number, if DP has been pressed and entry is thus in the fractional digits, CJ will track the digit position into which the next numeral is to be entered.

CJ is used in zero-blanking, and set to home before PM enters P09-wait [B301.6].

In P08 of the first numeral entry, CJ is incremented for the integral digits [B401.6]. CJ can now be said to point to the units digit in that the number of increments needed to bring CJ home is the distance of the units digit from the LSD.

Also in P08, for all numeral entries, if the DP key has been pressed (PDPE=1), CJ is incremented [B201.5]. Thus, at the end of the first P08, CJ points to the first fractional digit.

In P18, CJ is incremented per digit for the number cycle [B302.6]. For fractional digits, when CJ is at home in this cycling the current digit is the target for entry of the numeral.

The process:

- Start When SM releases PM to execute the NML-ENTRY procedure the numeral to be entered is resident in DN, asserting (DN=0~)=1 [E302.6]. This assertion is the indicating condition for PM to execute NML-ENTRY.
- P08 If this numeral entry begins a new number, RD must be cleared before entering the numeral: if PNEWN=0, recycling of RD is suppressed [C209.5].

 If this numeral entry begins a new number and the DP key has been pressed, an illuminated-0 must be entered in the units digit: if PNEWN=0 and PDPE=1, 0b1000 is fed to RD during UNITSØD [B406.5].
- P18 The number cycle starts in the fractional digits (if any) with RD cycling on itself [B405.3]. If CJ reaches home, DN is inserted into the data loop [B405.5,B405.1] for the current digit period, the new numeral thus written to that digit.

 Once in the integral digits, if the DP key has been pressed, RD continues cycling on itself but routed via gate [B406.6]. If the DP key has not been pressed, DN is inserted into the data loop [B405.6]. The new numeral is thus written to the units digits, but the DN insertion is held for the rest of the integral digits, so are all shifted up one digit.

MEM-CLR & MEM-IN

These operations are largely identical, with the distinction of 0 being fed to the DX inputs for MEM-CLR, instead of RD and RS for MEM-IN [see PM State table].

MEM-OUT & KTREAD

These operations are straightforward. In P12, the currently-selected RS register or RT as appropriate, is copied to RA. In P13, RD is saved to RB as a prior operand/result, and RA copied to RD for display.

REVERSE

REVERSE swaps the currently displayed number (RD) with the previously displayed number. The previous displayed however, depends on recent actions. PNEWN is referenced to resolve this. If the current displayed is a newly entered number, the previous must be obtained from RA [C115.1]; if the current displayed is a result, the previous must be obtained from RB [C216.1].

The PNEWN2 flag is also used to carry status over P09-wait for a 2nd REVERSE [D111.6,D109.3].

QUEUE-OP

The infix entry sequencing results in some complexity to deciding when an arithmetic procedure is to be executed. In general, the idea is when an arithmetic key is pressed (recorded in DF), DFp (the queued function) must be observed to select the procedure. At the end of an arithmetic procedure, DF is moved to DFp.

For arithmetic (ASMD) keypresses where there is no prior operation requiring completion (only the first operand for a binary function has been entered), no arithmetic needs to be done, the operand in RD just needs to be copied to the result register RA.

PM performs a pass through P12 during which RD is copied to RA [C110.3]. If RD is negative (RDNEG=1) the RD bitstream is inverted by XNOR RDCOMP as it enters the DX/RA bit loop [C211.5]. Be it positive or negative, RA thus receives the number in signed form.

ADD/SUBTRACT

- Addition For an arithmetic keypress where there is a queued function (the 2nd operand has been entered), PM heads to P06 from P0A. In P06, RD is added to RA [C211.5]. A pass through P07 is then performed but this is an inconsequential dummy state.
- Subtraction Prior to P06, at the end of P0A, RDNEG is flipped if the operation to be performed is subtraction as recorded in DFp [D209.10,D105.5].

To perform subtraction with the 1242 encoding, the complemented subtrahend must be added to the sum. As in QUEUE-OP, the complementing is accomplished by inverting the bitstream of RD via RDCOMP as it enters DX/RA.
- P17 - Finish At the completion of P06, RA holds the result, however this result is in signed form and if negative must be complemented to be displayed.

At the end of P06/7, PNEG is set to the sign of RA [D205.5]. In P17, RA is copied to RD but inverted via XNOR RACOMP if PNEG=1 [C415.6,C402.8]. PNEG is copied to RDNEG [D408.6].

Also in P17 - as this is the completion of an arithmetic procedure - DF is copied to DFp [D109.5].

MULTIPLY

- Loops

There are two loops in the process:

 - A primary loop of P1E, with occasional digressions to P0E, in which each iteration processes a digit of the multiplier and produces a digit of the product.
 - A secondary loop of the single state P0E, in each iteration of which the multiplicand is added to the product.
 - Registers

RB will be the multiplier and will receive completed digits of the product. RB will be shifted down in the primary loop and the removed LSD decremented in the secondary loop to count additions of the multiplicand to the product.

RD is the multiplicand. It is not changed in the process.

The product will accumulate in RA. RA is also shifted down in the primary loop with the completed digits of the product being shifted from the low end of RA into the high end of RB.
 - CJ counter

CJ serves as a counter of iterations of the primary loop. CJ will be decremented in P1E for each iteration.
 - Phases

There are two phases, 1 & 2, to the process. A phase ends when CJ returns home, with a brief but distinct transition period between the phases. The digits of the multiplier are processed in phase 1 to produce the product, in phase 2 some further shifting of the product is performed to account for the fractional digits.
 - PPHE flag

PPHE is set 1 at the end of phase 1.
 - PPH2 flag

PPH2=1 to indicate processing is in the 2nd phase.
- The process:
- P16 - Preparation

At the end of P0A, PNEG is assigned the sign of RA by assessing the MSB of RA. PNEG has been cleared 0 prior to the end of P0A. At EON, (bit 3 of digit 15) the MSB of RA is presented on signal RA. If RA=1, RA is a negative number, and PNEG set 1 [D208.5].

In P16, RA is transferred to RB [C105.6] and RA cleared 0. If RA is negative as now indicated by PNEG=1, RA is inverted during the transfer by the XNOR RACOMP. Both the multiplier RB and multiplicand RD are now positive-magnitude.

At the end of P16, PNEG is inverted if RDNEG=1 [D115.6]. PNEG is now the sign for the product.

CJ is set to home [B301.6].
 - P1E - Digit shift

RA and RB are shifted own one digit in each iteration through P1E. The shift-down is accomplished by the number cycle being extended but *not* extending the bitstream loop, there thus being 17 shifts for 16 digits.

For RA, the shift means the partial products from additions of the

multiplicand RD will accumulate in higher digits of the product. On the 17th shift the LSD of RA is shifted into the MSD of RB [C215.3,6]. The data-looping of RA is suppressed for the 17th shift so the LSD does not rotate into the RA MSD [C212.6].

For RB, on the 17th shift the LSD is fed to DY through an XNOR gate [C104.5..C105.3]. In phase 1 (PPH2=0), DY is thus loaded with the complement of the LSD. Arithmetically this is equivalent to $DY = 9 - RB[0]$. In phase 2 the value is not complemented.

CJ is decremented for each pass through P1E [B301.3]. When 16 digits have been processed, CJ returns home. PPHE is set 1 indicating the end of phase 1. In the next pass through P1E, CJ is incremented to the number of fractional digits. This is accomplished by incrementing CJ per digit [B302.3] from the beginning of the number cycle until phase 2 begins as indicated by flag PPH2 being set 1 when UNITØD=1 [B305.3].

In phase 2, CJ continues to be decremented in each pass through P1E but will now return home after (# fractional digits). When CJHOME=1 in phase 2, the primary loop exits and PM heads to P14 for rounding [B404.3].

P1E loops on itself while $DY = 9$ [C201.3]*, that is, the multiplier digit being processed is 0. If the digit is not 0, PM.10 is toggled to take PM to P0E [D415.11,C414.3].

* (DY.1 has been optimised out of the test).

- P0E - Add

The multiplicand RD is added to the product RA.

DY is incremented [C205.5], and P0E loops until $DY = 9$. As DY was loaded with the complement of the multiplier digit, this is the equivalent of decrementing the multiplier digit until it is 0. When $DY = 9$, PM.10 is toggled again, taking PM back to P1E.

DIVIDE

- P16 - Preparation The same actions as described for multiplication are performed.
- P1C,P0D,P0C The operation of these states is essentially as described in the SQUARE-ROOT commentary.

SQUARE-ROOT

The square root procedure implements a variation of a well-known algorithm once commonly performed manually. The algorithm has similarities to long division, discernible in the PM state graph.

A constructed divisor will be repeatedly subtracted from a reducing dividend (the radicand) starting from the most significant digits, while building a digit of the quotient (the root) from the count of subtractions. The construction of the divisor involves incrementing digits through the odd numbers. The digits of the dividend and divisor will be processed in pairs; the pairing must be aligned to the decimal point.

As with division, there are two loops in the process: a primary loop for shifting digits and a secondary sub-loop for subtractions. There are two phases to the process. In the 1st phase, the 16 digits of the radicand will be dealt with in 8 iterations of the primary loop (16/2 due to the pairing of digits). In the 2nd phase, the primary loop will iterate for the number of fractional digits to further reduce any remainder.

Elements in use:

- Registers

RB starts out containing the radicand. RB will be shifted up with digits of the radicand shifting from the high end of RB into the low end of RA.

New digits of the root will be produced in DY, DY will be shifted into the LSD of RB, so the root will build in the low end of RB as the radicand evacuates from the high end.

RD will contain the constructed divisor.
- CJ counter

CJ will be used as a loop counter for the primary loop (P1C..P0C). Digits of the radicand are processed in pairs. CJ will be decremented twice per iteration. The initial radicand will be thus be processed in 8 iterations.
- PPHE flag

A phase ends when CJ returns home. PPHE is set 1 to briefly carry this state at the end of a phase.
- PPH2 flag

PPH2=1 to indicate processing is in the 2nd phase.
- PQRDΛØD flag

PQRDLØD will assert 1 during the LSD digit of the radicand in RB, that is, PQRDLØD asserting 1 marks the transition point in RB from the root to the radicand [B404.6].

This is accomplished by observing when CJ hits home. CJ is being decremented in the P1C & P1D number cycles by 15 increments. At the beginning of the number cycle, the number of increments to CJ being home corresponds to the number of digits of the root.
- PQRDØ flag

During the number cycle, PQRDØ=1 for the period of the digits in RB containing the radicand: PQRDLØD=1 sets flag PQRDØ to 1, the flag remains asserted until the end of the number cycle.
- PQPA flag

PQPA=1 if the digit pairing is aligned to the DP; if not, an initial extra shift of the radicand must be performed.

The process:

- P12

The radicand in RD is moved to RA and RD cleared.
- P16

The radicand is moved from RA to RB and RA cleared. CJ is sent to home base.
- P1C & P1D

The objective of these two states is:

 - (a) shift a digit pair of the radicand from RB into RA,
 - (b) shift a newly formed digit of the root into RB,
 - (c) decrement the divisor in RD and shift it up one digit,
 - (d) adjust the loop counter.

	Phase 1	Phase 2
1C	$\ll RA \ll RB(\text{rad}), RB(\text{root}) \ll DY$	$\ll RA, \ll RB(\text{root}) \ll DY$
1D	$\ll RA \ll RB(\text{rad}), \ll RD$	$\ll RA, \ll RD$

- P1C

Perform the first shift of a pair.

(a): In both phase 1 & 2, RA and RB are shifted up: RA and RB operate on a 17-digit data loop but the number cycle remains at 16 digits, consequently they will be shifted up one digit. RA is extended with DX, RB with DY [C112.3,C113.3,C215.5,C206.6].

In phase 1, the MSD of RB shifts into the LSB of RA: during ØD15 of the preceding state (be it P0C or P1C), the MSD of RB was shifted into DX [C210.5,C210.6]. As P1C or P1D begins, this digit shifts into RA as the LSD. (On loop entry from P16, DX should{?} be 0).

(b): Prior to P1C, a new digit of the root was built in DY. As P1C begins this is shifted into RB as the LSD.

(d): CJ is decremented by 15 per-digit increments [B301.3].

- Pair alignment

On the first entry into the primary loop, P1C may loop for two state cycles, the extra cycle serving to align the pairing to the DP position.

During P0A, flag PQPA will be cleared at the units digit as [B303.5=>J] does not assert due to UNITØD~=0 while [B303.6=>K] does assert. For the rest of the number cycle, PQPA will toggle as both J & K are asserted. At the end of the number cycle, PQPA=1 if the digit pairing is aligned to the DP (integral and fractional digits are even) and PQPA=0 if not.

At the end of P1C, if PQPA=1 (pair-aligned) PM heads to P1D [D115.3], if PQPA=0 P1C will loop for a 2nd shift. PQPA is set 1 at the same time [B304.3] to indicate pair-alignment has been done, this assertion persists for the rest of the procedure.

- P1D

Perform the 2nd shift of a pair.

(a): In phase 1, while the radicand digits in RB must shift up, the root digits must remain at place. As CJ is being incremented 15 times during the number cycle, CJHOME=1 marks the end of the root and the first digit of the radicand. PQRDØ is set at this point and remains so for the rest of the number cycle, i.e. the digits of the radicand. The shifting is suppressed for the root digits of RB [C404.5,C113.6].

In phase 2, shifting of RB is not necessary as the radicand digits were all shifted out in phase 1 [C404.8,9].

(c): RD is extended with DN and so will shift up one digit. Additionally, the first bit of the LSD is cleared [C401.3]. As this is a constructed digit of the divisor, it is known to be odd at this point, so clearing the LSB in effect decrements the divisor.

(d): CJ is decremented as in P1C.

- Phase 2 prep

At the end of P1D, if CJ is home the phase is at its end [D109.7]: PPHE is set 1. In the next P1C, PPHE being asserted, PPH2 is set 1 [B305.3], and will remain asserted for the remainder of the square-root procedure [B305.6]. Having transferred its state, PPHE is now cleared at the end of P1C [D110.6].

PPH2 however, is not set 1 until UNITØD in the P1C number cycle. In the digit periods from the start of the number cycle till UNITØD, CJ is incremented [B302.3], setting CJ to a count of the number of fractional digits. This then, is the primary-loop iteration-count for phase 2.
- P0F

The constructed divisor in RD is incremented by 1 or 2. In this state DX is the extension digit for RD [C111.3].

On first entry to the P0F-P0D sub-loop (arriving from P1D), an increment is produced by adding 1 via DXb [C115.6].

On subsequent iterations (arriving from P0D), the +1 via DXb is in effect, but additionally, DXCA=1 remains asserted from the subtraction in P0D, providing a $2^{\text{nd}} + 1$. The constructed digit of the divisor thus follows the pattern 1,3,5,7...
- P0D

RD is subtracted from RA, the negation being provided by [C111.5] and [C414.6].

If RA remains positive (DXCA=1), DY is incremented [C205.6]. P0F and P0D loop [B311.6] until RA goes negative (overdraft) [C202.6].
- P0C

The overdraft is corrected by adding RD back to RA. If in phase 2 as indicated by PPH2=1, when loop counter CJ returns home, the primary loop exits [B404.3,D415.2].

Supervisor Layer

SM - Supervisor State Machine

The Supervisor State Machine guides the initial response to keypresses prior to invoking PM to perform an appropriate procedure, and guides the user-program operations such as performing the fetch-execute cycle and managing the program store.

The current state of SM is stored in binary in a register of 3 flip-flops. 6 of the 8 possible states are used (3 & 4 are not). State values are presented as *Sn*.

For all commentary on SM, see the SM State Graph [SCX.5].

SPC - Program Counter

SPC is the program counter for programs input by the user. SPC is 7 bits in total but formed from a 4-bit SR16-type counter (SPd) and another 3-bit shift-register counter (SPr). Together, they address a digit position in program storage. SPr indicates one of the 7 program registers while SPd indicates a digit in that register.

- SPd SPd follows the full SR16 sequence:
 [F E C 8 0 1 2 5 A 4 9 3 6 D B 7]
 SPd may be incremented by a single clock pulse [E402.3] or decremented by 15 clock pulses [E207.3].
- SPr SPr follows a 7-state sequence:
 [5 3 6 4 0 1 2]
 State value 7 is absent from the sequence.
- Reset The initial value of SPC, and hence the start of programs, is SPr=0x5 & SPd=0xF. SPd is set to the initial value by clocking via [E307.3,E207.6] until 0xF is reached. SPr is set to the initial value by the clearing of bit SPr.2=0 at INIT and clocking via [E309.3,E208.5,E208.7] until 5 is reached.
- Addressing During states S5 and S7 a digit in the program store is addressed for instruction fetch or store.

 During the number cycle, SPd is incremented per digit for 15 digits [E207.3]. At some digit SPd is 0xF, asserting SØDATPC. This is the desired addressed digit.

 At the end of the number cycle, SPd has been decremented by the 15 increments. At this time, if SPd has looped to the 'first' count position (SPd=0xF), then a carry is generated to increment SPr to the next program register [E208.9,E309.6].

 Note that in stepping through the program store, while SPC may be said to be incremented to the next program step, SPd is decremented, while SPr will be incremented at carry.

- **Fetch & Store** If fetching (S5), during the addressed digit period the RS bitstream is enabled to RP [E308.3] and directed to DF for a function code (SNML=0) [E402.5] or DN for a numeral (SNML=1) [E307.6].

If storing (S7) [E409.3], during the addressed digit period DF and DN are fed to RSw [E411.3] and RSwENB asserted. Only one of DF or DN is non-0 so only one will supply data to the selected RS register.

Execution & the EXEC Flag

The EXEC flag controls whether SM leaves idle to perform some operation.

- **Initiate** The pressing of a key causes the NORN gate [DD1] to assert 0. This produces a 0-going pulse via a 1 μ F capacitor which is double-inverted [QF24,E203.6] to asynchronously set flag EXEC=1. This results in a final pass through S0 but with SST=1 [B304.6]. SM then heads to a new state dependant on the operation mode.

In AUTO mode, an instruction fetch is performed in S5 with the program digit serially loaded into DF as a function code.
- **SST** When SM leaves idle to start some operation, it always executes a state cycle in S0 with flag SST=1. The only discerned activity in this state is to decrement SPC for EDIT-mode operations. This may have been a modification made late in the design process to accommodate EDIT-mode, and deemed easier to incorporate than modifying the SM state machine logic, even though there are two unused SM state values.
- **Execution** There are three ways execution may be accomplished:
 - A few operations are performed by SM, these are MSET, DIGIT and JUMP.
 - Other operations with a function code will end up in S1. At the end of S1, PM is kicked out of P09-wait and proceeds to execute the procedure indicated by the function code in DF, or NML-ENTRY.
 - Non-function-code operations (Change-Sign & DP-Shift) kick PM directly [KSCS].

With PM busy, SM heads to S0 and loops there until PM re-enters P09-wait.
- **Terminate** EXEC remains asserted for some fixed period while the capacitor charges and for some variable period until synchronously cleared at EON by one of a variety of conditions indicating procedure completion [E203.5].

In MANUAL mode, SM now loops in IDLE, until a keypress asserts EXEC.

In AUTO mode, the clearing of EXEC is suppressed [E304.8,E304.2] and SM continues through S0 to fetch another instruction. The end of a program (DF=0) or a STOP code (DF=8) is detected by [D201.5] to clear EXEC=0.

MANUAL Mode

- Start For the pressing of coded keys, DN or DF is loaded with a non-zero value, taking SM through S0•SST to S1 or S6.
- Execute SM heads to S1 for function codes requiring execution by PM. The S1 state triggers PM to leave wait.
For MEMSET, DIGIT and JUMP, SM proceeds to S6. Appropriate flags are set to direct the next iteration.
- Finish Once PM has finished (returned to P09) or the local operation completed in S6/S2, EXEC is cleared and SM returns to idle.

AUTO Mode

- Start Program execution begins, or is restarted, by a pressing of the S key setting EXEC [KPS~=>QF24].
- S5 - Fetch SM leaves idle for S0•SST and proceeds to S5 for a digit fetch from the program store. By the end of S5, SPC has also been incremented to address the next digit in the program sequence.
- Execute & Next Same as in MANUAL mode, except clearing of EXEC is suppressed so SM proceeds through S0 to fetch a new instruction.

LIST Mode

LIST mode is enabled by (KCHECK & KAUTO). In LIST mode, each press of the S key must fetch a program digit and present it as a visual display.

- Fetch & load Pressing the S key sets EXEC [KPS~=>QF24]. SM leaves idle for S0•SST and proceeds to S5 for digit retrieval from program store. S5 loads the program digit into DF or DN as usual. However, function-code digits are also serially loaded into CJ [B403.10,B203.6], with clocking of CJ enabled at register bitstream rate (ØB) [C403.6].
From S5, SM proceeds to S6. The recycling of data in RD is inhibited, so RD is cleared [C207.11].
- Function display To illuminate the appropriate function lamp, the function-code value loaded into CJ will be translated into a digit position in time (ØDn).
Incrementing of CJ is enabled at the digit rate [B201.10]. As the S6 state cycle proceeds, when CJHOME=1 the current digit of RD is loaded with value 0x9 [C108.3]. This is a special code not in the 124W set.
At the end of S6 then, RD has been cleared except for a 0x9 in one digit - the digit being dependant upon the function-code value. During multiplexing of the display in IDLE, the 0x9 code is detected [B415.3] and turns on one of the function lamps according to the digit period.
- Numeral display For program numeral digits, during S6 the numeral is loaded into the LSD of RD [C209.6].
- Step finish SM proceeds through S2 and back to IDLE.

LEARN Mode

In LEARN mode, execution proceeds as in MANUAL mode with the distinction of passing through S7 to store program digits corresponding to the keypresses.

- S7 - Store SM leaves idle for S0•SST and proceeds to S7 [E306.3,E305.3,E205.5]. In S7, the current program location is addressed and either DF or DN written to the program store, as described in the Program Counter commentary.

By the end of S7, SPC has been incremented to address the next digit in the program sequence. SM proceeds to S1 or S6 to execute the keypress operation.
- Parameter entry If numeral(s) for a program parameter are entered while in LEARN mode (as distinct from entry of numerals for a constant using the DIGIT instruction), a single STOP instruction is to be stored for the contiguous sequence of numeral keypresses.

Flag SPARM is set 1 during P18 of a numeral entry [D216.10,D407.10]. With SPARM=1, when a subsequent operation-keypress is being processed, two passes through S7 will be performed. The first pass stores the STOP instruction (0b1000) [E410.6] while suppressing data from DF [E411.2]. SPARM is cleared 0 at the end of this pass [D408.5], but SM.1 is inhibited from clearing at this point as SPARM has not been cleared for the SM state transition [E206.8]. SM thus performs A 2nd pass through S7, during which the operation function code from DF is stored. At the end of this pass SM.1 is cleared, taking SM to S6.
- End Program The end of a program is indicated by an EOP instruction (function code 0). The EOP is added automatically upon moving from LEARN mode to AUTO mode.

In LEARN mode, the capacitor at the base of QF22 is discharged, QF22 is not conducting, and the capacitor at the collector is also discharged. When the mode switch is flipped from LEARN to AUTO to end program entry, the base capacitor begins charging. After a delay QF22 conducts, and the collector capacitor begins charging. F309.4 is taken 0 for some period while the collector capacitor charges. The net result is SADDEOP is asserted 1 briefly after a small delay from the flipping of the switch.

The SADDEOP pulse initiates execution (assertion of EXEC) and directs SM to perform a LEARN operation [E305.6]. As no function code was loaded into DF to initiate this operation, the function code entered into the program store is 0 (EOP).

EDIT Mode

EDIT mode is enabled by (KCHECK & KLEARN). EDIT mode employs the address-and-store facility of S7 as in LEARN mode, however the program counter must first be adjusted.

- **SST - Decrement** When EDIT mode is entered, either from LIST or LEARN mode, SPC has been post-incremented to the next program digit. To edit the current digit, SPC must be decremented.

The SPC decrement is performed in the S0•SST state as SM leaves idle for the edit keypress entry. For SPd the adjustment is simple, it must merely be incremented once to complete a 16-count cycle and return it to its pre-increment value [E402.3]. Note this is not done until EON.

However SPr may also have been incremented and must be decremented. At the start of S0•SST, if SPd=0xF, during the number cycle PCr is incremented 48 times (16 digits * 3 bits, 3 from ØB3~) [E402.6]. 48 is one less than 49, 49 is an integral multiple of 7, 7 is the cycle modulus of SPr.
- **S7 - Store** SM proceeds to S7 to address program storage and store the program digit, as in LEARN mode.
- **S6, S2 - Display** In contrast to LEARN mode, in EDIT mode the keypress operation is not performed. Rather, the new operation is displayed as in LIST mode. To this end SM proceeds to S6 followed by S2, as described in the LIST mode commentary.

IDLE

Idle occurs when both SM and PM are in their wait state (S0,P09). ØDC operates in the normal 16-state sequence. The purposeful activity during idle is multiplexing the display Nixies.

- **Numeral Display** Registers cycle on a 16-digit loop but digits from RD are also fed to DX. At the end of each digit period, the digit from RD being serially loaded into DX is parallel-loaded into display latch DD.

DD provides a stable numeral state during the next digit period while one Nixie anode is enabled. The LSD Nixie is illuminated during ØD1.
- **Thousands marks** Thousands-markers are produced by turning on the DP cathodes at the appropriate digit positions but with a reduced duty cycle.

With thousands input DPCTH=1, DP=1 when DPC=8,5,C,D as well as when DPC=3.

During idle, CK is incremented at the end of each number cycle [B206.8]. For two sequential states of CK (CK=E or CK=F), DPCTH is asserted 1 [B209.5]. Thus - as CK counts a 16-state loop - the additional DP cathodes at the thousands positions are illuminated for 2 out of every 16 number cycles (1/8 duty cycle).

DIGIT & the SNML Flag

DIGIT is a program-only operation (AUTO mode). The DIGIT instruction emulates a numeral keypress: the next digit from program storage is entered as a numeral into RD.

- **SNML Flag** The SNML flag will be used to hold state into the subsequent SM loop to indicate the program digit will be given special treatment as a numeral. This is a multi-use flag (see PNEG).
- **SM 1st loop** After the DIGIT instruction is received in S5, SM proceeds through states S6 and S2 with the D function code. At the end of S6, SNML=1 is asserted if the function code is D or M [D201.6].
- **SM 2nd loop** SM proceeds through S0 to fetch another digit from program storage in S5. SNML=1 now directs the program digit to DN [E301.6] rather than DF [E402.5].

SM heads from S5 to S1 [E403.5], and with (DN=0~)=1 PM leaves P09-wait to execute the NML-ENTRY procedure.

MEM-SET & the SMSET Flag

MEM-SET selects one of the registers M1:M7 for access. The objective is to load DM with a numeral from the keyboard or program storage. This is handled by SM without involving PM.

- **SMSET Flag** When SM is expecting a numeral entry for DIGIT or MEM-SET operations as indicated by SNML=1, SMSET distinguishes between DIGIT and MEM-SET.
- **SM 1st loop** An M function-code is received, either from keypress or S5 fetch, and proceeds through S6 and S2. The M function code asserts SNML=1 and SMSET=1 to indicate the register number is expected [D201.6,D406.6].
- **SM 2nd loop** In MANUAL mode, SM waits in S0 for a numeral keypress.

In AUTO mode, SM proceeds through S0, fetches the numeral from program storage in S5 and loads it into DN.

Once the numeral is obtained, in S6 the numeral is loaded into DM from DN [E216.3,E215.3]. SNML and SMSET are cleared 0 at the end of S6 [D205.11,E204.8].

JUMP & the SSKIP Flag

The SSKIP flag indicates whether SM is to skip over instructions without performing them, as a result of a JUMP instruction.

- **MANUAL** SSKIP is held 0 in MANUAL mode [DE19].
- **JUMP** The JUMP instruction is recognised in S6 by gate [D104.6]. If not currently skipping (SSKIP=0) and RDNEG=1, SSKIP is toggled to 1. If currently skipping (SSKIP=1), SSKIP is toggled to 0.
- **SSKIP=0** Normal SM execution.

- SSKIP=1 SM skips over instructions by directing SM to S2 instead of S1 from S6 [E403.10].
At the end of a program the clearing of EXEC is inhibited [D201.11]. SPC is none-the-less reset to the program start (0x5F) [D401.6] so program execution loops.
Loading of the M register is inhibited [E216.1].

Non-code Operations

Several operations do not have a function-code, and hence can not be performed by a program. These are:

CLR-ALL
CLR-ENTRY
CHG-SIGN
DP-LSHIFT
DP-RSHIFT
DP-ENTER
KTREAD

- Key-press state As a function code has not been stored in DF (and DN remains 0), these operations rely on the key state persisting for some number of state cycles while the operation is executed.
- CLR ops For CLR-ALL and CLR-ENTRY, SM remains in S0 wait. PM heads to the P00 and loops while the key-state is asserted.
- CH-SIGN & SHIFT In executing these operations, SM goes through S0•SST and promptly returns to S0.
The exit from S0•SST together with the key state triggers PM.
EXEC is also cleared before the operation has been completed.

Appendix 1 - Table of PM States

PM States			
State	NC EXT	Procedures	Description
P00		MAIN	0=>DF
P01		MAIN	0=>DF. Zero-blanking assessment of RD.
P02	√	MCLR,MIN, Tally	RD=>N=>RD RA=>RA RB=>DY=>RB
		MCLR	RT=>RT 0=>Xb=>RS
		MIN	Add RD to RS RT=>RT (RDCOMP=>Xa) + (RS=>Xb) => RS 1->PNEWN2
		Tally	Add RD to RT (RDCOMP=>Xa) + (RT=>Xb) => RT RS=>RS
P03	√	MCLR,MIN, Tally	Realign-rotate. Loop till realigned. RD,B extended. RA normal. RD=>DN=>RD RA=>RA RB=>DY=>RB
		MCLR	RS extended. RT normal. RT=>RT 0=>DXa=>RS
		MIN	RS extended. RT normal. RT=>RT RS=>DXa=>RS
		Tally	RT extended. RS normal. RT=>DXa=>RT RS=>RS
P04	√	MAIN	Realign-rotate. Loop till realigned. RD,A,B extended. RT,S normal. RD=>DN=>RD RA=>DXa=>RA RB=>DY=>RB RT=>RT RS=>RS
P05	√		-

PM States			
State	NC EXT	Procedures	Description
P06	✓	ADD	Add RD to RA RD=>DN=>RD (RDCOMP=>DXb)+(RA=>DXa)=>RA RB=>DY=>RB
P07	✓	ADD	dummy state RD=>DN=>RD RA[~15]=>DXa=>RA RB=>DY=>RB
P08		NML-ENTRY	Start new number and track entry position. PNEWN=0: Clear RD, #integral_digits=>CJ PNEWN=0 & PDPE=1: 0b1000=>RD[UNITSØD] PDPE=1: ++CJ
P09		MAIN	Wait for SM. Display RD. Digits serially-loaded into DX from RD are parallel loaded into display latch DD. RD=>RD RD=>DXb->DD
P0A		MAIN	De-blank RD. Scan through RD replacing digits=0b1000 (illuminated zeroes) with proper zeroes (0b0000). if PNEWN2=1: 1->PNEWN if Recall: 1->PNEWN2
P0B			-
P0C	✓	DIV,SQR	Overdraft correction: add RD back to RA RD=>DN=>RD (RD=>DXb) + (RA=>DXa) =>RA RB=>DY=>RB
P0D	✓	DIV,SQR	Subtract RD from RA, loop while ~borrow RD=>DN=>RD (1=>DXCA) + (~RD=>DXb) + (RA=>DXa) =>RA RB=>DY=>RB ++DY
P0E	✓	MUL	Add multiplicand to product, loop while DY≠9 (RD=>DXb) + (RA=>DXa) =>RA
P0F	✓	SQR	Add 1 to RD (RD=>DXb) + (1=>DXa) =>RD RA=>DN=>RA RB=>DY=>RB
P10			-
P11		MAIN	Zero-blank RD. 0->PNEWN

PM States			
State	NC EXT	Procedures	Description
P12	√	QUEUE-OP	RD=>DN=>RD RDCOMP=>DXb=>RA RB=>DY=>RB
		MOUT	RS=>DXb=>RA
		KTREAD	RT=>DXb=>RA
		REV• PNEWN=0 (result)	RD=>DN=>RD RB=>DXa=>RA RB=>DY=>RB
		REV• PNEWN=1 (new number)	RD=>DN=>RD RA=>DXa=>RA RB=>DY=>RB
		SQR	0=>DN=>RD RDCOMP=>DXb=>RA RB=>DY=>RB
P13	√	MOUT, KTREAD,REV	RA=>DN=>RD RD=>DXb=>RA RD=>DY=>RB
P14	√	MUL,DIV,SQR	RD=>DN=>RD RB=>DXa=>RA RB=>DY=>RB
P15	√	MUL,DIV,SQR	dummy state RD=>DN=>RD RACOMP=>DXa=>RA RB=>DY=>RB
P16	√	MUL,DIV	RD=>DN=>RD 0=>DXab=>RA RACOMP=>DY=>RB if RDNEG: PNEG~=>PNEG 3->CJ
		SQR	0=>DN=>RD 0=>DXab=>RA RACOMP=>DY=>RB 3->CJ

PM States			
State	NC EXT	Procedures	Description
P17	√	MAIN	Copy result RA to RD for display. Save operand RD to RB. Save DF to DFp. RACOMP=>DN=>RD (RA RB?)=>DXa=>RA RDCOMP=>DY=>RB 0->PNEWN2
P18		NML-ENTRY	PDPE=0: Enter integral digit. ØD<INTØD: RD=>RD ØD>=INTØD: RD=>DN=>RD
			PDPE=1: Enter fractional digit. Cycle CJ. CJ≠3: RD=>RD CJ=3: DN=>RD
			1->PNEWN
P19			-
P1A			-
P1B			-
P1C		DIV,SQR	Shift up. Shift RA,B up, RB[M] to RA[L], new DY to RB[L] RD=>RD(=>DN) (RA[~15] or DY[15]) =>DXa=>RA RB[~15]=>DY=>RB -- CJ
P1D		SQR	2 nd shift of pair. RD[~0.0]=>DN=>RD (RA[~15] or DY[15]) =>DXa=>RA (RB or (RB[~15]=>DY)) =>RB -- CJ
P1E	√	MUL	RA>>, RB>>, loop while ?=0 RD=>DN=>RD RA[~15]=>RA RB[~15]=>RB RA[15]=>RB -- CJ
P1F	√		-

Appendix 2 - List of Operations

An operation is some activity performed by SM or PM. A sub-operation is an operation performed within another operation.

PM operation is performed by PM, though SM is involved in initiation.

SM operation is performed by SM.

KB operation can be initiated from the keyboard.

PGM operation can be initiated from a program.

Operation	PM	SM	KB	PGM
CLR-ALL	√		√	
CLR-ENTRY	√		√	
DP-LSHIFT	√		√	
DP-RSHIFT	√		√	
CHG-SGN	√		√	
NML-ENTRY	√		√	
DP-ENTRY			√	
MEM-SET		√	√	√
MEM-CLR	√		√	√
MEM-IN	√		√	√
MEM-OUT	√		√	√
KTREAD	√		√	
REVERSE	√		√	√
QUEUE-OP	√		√	√
ADD/SUB	√		√	√
MULTIPLY	√		√	√
DIVIDE	√		√	√
SQUARE-ROOT	√		√	√
ADD-EOP		√	√	
DIGIT	√	√		√
JUMP		√		√
START		√	√	
STOP		√		√

Sub-operation	PM	SM
ZERO-BLANK	√	
UNBLANK	√	
REALIGN-ROTATE	√	
ROUND	√	
TALLY	√	
PGM-FETCH		√
PGM-STORE		√
PGM-SHOW		√

Appendix 3 - Summary User Manual

Number Registers	
D	Displayed number: an operand being entered, or the result of the last operation.
A	Last result.
B	Prior operand or result, temporary during calculation.
T	Tally sum.
M1..M7	Arbitrary use. M5,6,7 may be overwritten by long programs, starting with M7.

Keys	
CALL	CLEAR ALL. Also resets the program counter to the beginning of the program.
CENT	CLEAR ENTRY. Clear the displayed number.
< >	Shift decimal point left or right. Numbers in registers are not adjusted to follow.
0::9 DP	Number entry keys.
CHG SIGN	Change Sign of displayed number.
+ - * ÷	If an arithmetic operation is queued, perform it. Queue the keypress arithmetic operation.
√	Produce the square root of the displayed number.
=	If an arithmetic operation is queued, perform it. Add the result to the tally register if Tally mode is enabled.
R	Reverse operands.
M	Select the Memory register (M1:M7) which will be accessed by the MIN, MOUT and MCLR keys. Use: Press the M key followed by a numeral key 1::7.
MC	Clear the currently selected Memory register.
MIN	Add Display register to the currently selected Memory register.
MOUT	Copy the currently selected Memory register to the Display register.

Keys - Programming Only													
D	DIGIT instruction. During program execution, the DIGIT instruction simulates the pressing of a numeral key. This allows constants to be entered in a program without dedicating an M register for the purpose. Only integers can be entered, there is no ability to enter a decimal-point in the constant. Use: In LEARN mode, press the D key followed by a single numeral key. Example: The program sequence: D 1 D 2 enters the digits 12 during program execution.												
J	JUMP instruction. The JUMP instruction provides 3 programming constructs in one instruction: conditional skip - providing both jump forward and conditional execution, and the ability to loop in the program. The semantics for J during program execution are: • If D is positive, continue performing instructions. • If D is negative, skip following instructions until a J instruction is encountered. • If skipping and an EOP instruction is encountered, loop back to the beginning and continue skipping till a J instruction is encountered. There are some unusual semantics regarding 0: there is both a +0 and −0. The conditional skip treats them in accordance with their sign, thus a '0' value may or may not induce skipping. The production of +0 versus −0 is as follows: <table> <tbody> <tr> <td>• +0 − +0 = −0</td><td>• +0 + +0 = +0</td></tr> <tr> <td>• −0 − +0 = −0</td><td>• −0 + +0 = −0</td></tr> <tr> <td>• +0 − −0 = +0</td><td>• +0 + −0 = −0</td></tr> <tr> <td>• −0 − −0 = −0</td><td>• −0 + −0 = −0</td></tr> <tr> <td>• +x − +x = −0</td><td>• +x + −x = −0</td></tr> <tr> <td>• −x − −x = −0</td><td>• −x + +x = −0</td></tr> </tbody> </table> Note that a −0 can be converted to +0 by subtracting it from +0.	• +0 − +0 = −0	• +0 + +0 = +0	• −0 − +0 = −0	• −0 + +0 = −0	• +0 − −0 = +0	• +0 + −0 = −0	• −0 − −0 = −0	• −0 + −0 = −0	• +x − +x = −0	• +x + −x = −0	• −x − −x = −0	• −x + +x = −0
• +0 − +0 = −0	• +0 + +0 = +0												
• −0 − +0 = −0	• −0 + +0 = −0												
• +0 − −0 = +0	• +0 + −0 = −0												
• −0 − −0 = −0	• −0 + −0 = −0												
• +x − +x = −0	• +x + −x = −0												
• −x − −x = −0	• −x + +x = −0												
S	START / STOP / STEP • As a keypress in AUTO mode, start or restart execution. • As a keypress in LIST mode, step to the next instruction. • As a program instruction, stop execution, typically to pause for entry of a parameter.												

Switches			
MODE	MANUAL	Normal keypress execution.	
	AUTO	OPER	AUTO Mode: Program execution. Press S key to start.
		CHECK	LIST Mode: Each press of the S key displays the next program step.
	LEARN	OPER	LEARN Mode: Program entry.
		CHECK	EDIT Mode: The current program location can be modified with a new key entry.
TALLY	OFF	Tally mode OFF. The T register is cleared in moving from T to OFF.	
	T	Tally mode ON. Results produced from pressing the = key are added to the T register.	
	READ	Copy the T register to the Display register.	
ROUND DIGITS	0	Select the number of LSD guard digits for rounding. E.g., when set to 4, the 4 LSD digits are calculated but rounded into the 5 th digit.	
	2		
	4		
ROUND MODE	UP	Round up.	
	5/4	Round ≥ 5 up, < 5 down.	
	DOWN	Round down.	

Indicator Lamps	
NEG	Displayed number is negative.
OVF	A calculation or entry is improper or has exceeded the permissible range for a number, or program entry has exceeded the program storage.
P1st	Program Counter is at the beginning of the program.
Pexp	Warning indicator that the program length has expanded into the Memory registers (M7,6,5).
functions	In LIST & EDIT mode, these indicate the function at the current program step.

Programming Notes

- A program is stored as a sequence of digits. A digit is either a function code or a numeral. Function codes are entered by pressing a function key. Numerals must be preceded by an M or D function code.
- Most instructions are simply a function code, thus one digit. Two instructions - DIGIT & MEMSET - are a function code followed by a numeral, thus two digits.
- Program storage is up to $112=7*16$ digits.
 - More than 64 digits will conflict with M7.
 - More than 80 digits will conflict with M7 & M6.
 - More than 96 digits will conflict with M7, M6 & M5.
- A program ends with an EOP instruction. An EOP is added automatically when switching from LEARN to AUTO mode.
- Program execution stops when an EOP instruction is encountered while performing, or an S instruction is encountered. If an EOP is encountered the program counter is reset to the start. Execution continues from the start if an EOP is encountered while skipping. This latter facility is the only way to form a loop.
- Program storage is cleared at power-on. There is no other simple operation to clear program storage. Program storage may of course be overwritten.
- Correcting during entry: While entering a program, a keypress may be corrected by switching to CHECK and pressing the correct key.
- Editing during listing: While stepping through a program for listing, program digits may be altered by switching to LEARN, and entering a new keypress.
- D & M instructions cannot be altered arbitrarily with EDIT mode, though they can be replaced with S or EOP.
- A program can be appended to by stepping to the last instruction in LIST mode, then switching to LEARN mode.

Program Example 1

Calculate $e = \sum(1/n!)$, manual iteration control

Calculate e from the well-known formula. Execution stops after each sum iteration to display the derived e so-far. Press S to restart for another iteration.

Statement #	Statement	Pgm Code	# Pgm Digits
1	n=0	M1 Mclr	3
2	p=1	M2 Mclr D1 Min	6
3	e=1	M3 Mclr Min	4
4	LOOP	J	1
5	n+=1	M1 D1 Min	5
6	p/=n	M2 Mout / M1 Mout = M2 Mclr Min	12
7	e+=p	M3 Min	3
8	display e	Mout	1
9	pause	S	1
10	negative to loop	– D4 =	4
11	start skipping	J	1
12	skip to LOOP	EOP	1
Total digits:			42

Program Example 2

Calculate $e = \sum(1/n!)$, till convergence			
Calculate e from the well-known formula. Execution stops when new e = previous e.			
Statement #	Statement	Pgm Code	# Pgm Digits
1	n=0	M1 Mclr	3
2	p=1	M2 Mclr D1 Min	6
3	e=1 (new)	M3 Mclr Min	4
4	ee=0 (previous)	M4 Mclr	3
5	LOOP:	J	1
6	n+=1	M1 D1 Min	5
7	p/=n	M2 Mout / M1 Mout = M2 Mclr Min	12
8	e+=p	M3 Min	3
9	e-ee	Mout - M4 Mout =	6
10	diff<=0 skip to FIN	J	1
11	ee=e	M3 Mout M4 Mclr Min	7
12	force neg to loop	- D 4 =	4
13	FIN:	J	1
14	display ee	Mout	1
15	TERMINATE / skip to LOOP	EOP	1
Total digits:			58

- The J at Statement #5 is a goto destination only. RD is always + here when performing (not skipping), J is thus a NOP. When skipping, the J terminates the skipping.

- The J at St#10 is a conditional jump. Due to the subtraction semantics whereby $x-x$ produces -0 , RD is + only if new e > previous e (ee), that is, they have not converged. If they have converged, then skipping is enabled.

- The J at St#13 is a forced jump and a destination. If encountered when performing (not converged), skipping is enabled due to RD being forced negative in St#12: execution skips to the end of the program, loops and performing starts again at St#5. If encountered when skipping (convergence achieved), performing starts and proceeds to termination of the program.

- Sim results:

2.7179	(n=8)
2.718277	(n=10)
2.71828182840	(n=15)
2.7182818284583	(n=16)

Program Example 3

Calculate $e^x = \sum (x^n / n!)$			
x = number in display. Execution stops at convergence.			
Statement #	Statement	Pgm Code	# Pgm Digits
1	x=displayed	M1 Mclr Min	4
2	n=0	M2 Mclr	3
3	p=1	M3 Mclr D1 Min	6
4	$e^x=1$ (new)	M4 Mclr Min	4
5	$e^{xp}=0$ (previous)	M5 Mclr	3
6	LOOP:	J	1
7	n+=1	M2 D1 Min	5
8	$p=p \cdot x/n$	M3 Mout • M1 Mout / M2 Mout = M3 Mclr Min	16
9	$e^x+=p$	M4 Min	3
10	$e^x - e^{xp}$	Mout – M5 Mout =	6
11	diff<=0 skip to FIN	J	1
12	$e^{xp}=e^x$	M4 Mout M5 Mclr Min	7
13	force neg to loop	D0 – D1 =	6
14	FIN:	J	1
15	display e^{xp}	Mout	1
16	TERMINATE / skip to LOOP	EOP	1
Total digits:			68

Sim results: e^3 = 20.0852 (n=14) $\sqrt{e^3}=20.0855$
 $e^{4.527}$ = 92.4807023929 (n=27) $\sqrt{e^{4.527}}=92.4807023953$

Program Example 4

Simpler than 4a.

Calculate $\ln(x) = \sum (-(1-x)^n/n)$; for $0 < x \leq 1$			
x = number in display. Execution stops at convergence.			
Statement #	Statement	Pgm Code	# Pgm Digits
1	y=1-x	- D1 R = M1 Mclr Min	9
2	n=0	M2 Mclr	3
3	p=1	M3 Mclr D1 Min	6
4	ln=0 (new)	M4 Mclr	3
5	LOOP:	J	1
6	n+=1	M2 D1 Min	5
7	p=p*y	M1 Mout • M3 Mout = Mclr Min	10
8	a=p/n	/ M2 Mout =	5
9	if a=0 then	- D0 =	4
10	skip to FIN	J	1
11	ln-=a	- D0 R = M4 Min	8
12	force neg to	D0 - D1 =	6
13	skip to LOOP	J	1
14	FIN: display ln	Mout	1
15	TERMINATE / skip to LOOP	EOP	1
Total digits:			64

Calc results: $\ln(0.5)$ = -0.6931471805595 (n=39) (RND=5/4)

Program Example 4a

Works but more complex.

Calculate $\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 - \dots$; for $0 < x \leq 1$			
x = number in display. Execution stops at convergence.			
Statement #	Statement	Pgm Code	# Pgm Digits
1	y=1-x	- D1 R = M1 Mclr Min	9
2	n=0	M2 Mclr	3
3	p=-1	M3 Mclr D0 - D1 = Min	9
4	ln=0 (new)	M4 Mclr	3
5	LOOP:	D0 J	3
6	n+=1	M2 D1 Min	5
7	p=p*y	M1 Mout • M3 Mout = Mclr Min	10
8	a=p/n	/ M2 Mout =	6
9	ln+=a	M4 Min	3
10	if a<=0 ..	- D0 =	4
11	skip to CHKN	J	1
12	force neg to ..	D0 - D1 =	6
13	skip to NOTZ	J	1
14	CHKN: if a=0 ..	- D0 R = - D0 =	9
15	skip to FIN	J	1
16	NOTZ: force neg ..	D0 - D1 =	6
17	skip to NEXT	J	1
18	FIN: display ln	Mout	1
19	TERMINATE / skip to LOOP	EOP	1
Total digits:			82

Calc results: $\ln(0.5) = -0.693147180562$ (n=36) (RND=5/4)

For further analysis:

- Operation of INTØ flag.
- D & M in EDIT mode.
- Production of negative-0.

Document Log

- 2024 Oct Initiated.
- 2025 ? First release.

—— EOD ——