Canon 163 Calculator

Theory of Operation

bhilpert / madrona.ca
2024 Jan
Rendition: 2024-Nov-16

Contents

Introduction

The Canon 163 is a desktop calculator from the late 1960s. A schematic for the model was produced from reverse engineering, based on two units of the model. A logic simulation was produced from the schematic. This commentary has been produced from analysis of the schematic and observation of the simulation. The units examined contain components with date codes from 1967 to 1969.

[reference]: Text within square brackets is a reference to some page, figure or logic element in the schematic. E.g. [pg 15] [fig A3.M1] [5205.8].

Architectural Overview

As typical for calculators of its era, the Canon 163 is a bit-serial design. At a functional architectural level, data processing involves six 68-bit shift registers, each holding a 17-digit decimal number, and a serial arithmetic unit. These 6 register are in actuality implemented in a single magnetostrictive acoustic delay line operated in torsion mode. Register data is continually rotating through the delay line, for processing, for supplying the multiplexed display during idle, and for maintaining the memory state.

A state machine directs the logic through higher-level procedures to implement number entry and the arithmetic operations.

Physical Implementation

The Canon 163 is implemented using SSI DTL ICs made by Texas Instruments. The ICs are labeled with type numbers of SN39xx and SN45xx. These do not appear in general TI Databooks, though they do have matches for pinout and function in common commercial DTL series. A solitary page of documentation specific to these ICs was received in the mid-1970s from a service department, which included the internal schematics presented on [pg 20].

The ICs contain primarily 2, 3 & 4-input NAND gates. The outputs permit wired-OR operation, this is used extensively with occasional discrete diodes at inputs for isolation [pg A1].

Flip-Flops

Nearly all flip-flops in the implementation are formed from two NAND gates with capacitor triggering. The capacitive storage and RC time constants provide a storage element prior to the active NAND flip-flop, in a sense functioning as a master stage to a NAND slave stage. The capacitive storage allows a zero-to-negative hold time for the flip-flop, allowing synchronous triggerring of flip-flops with reliable transfer of state. At the time, this was more economic than full M-S flip-flops.

The basic form of these flip-flops and several variations commonly used in the design are shown on [pg A1].

Architectural Elements

Timing

A timing diagram for the major clock signals is presented on [pg A2].

- Master clock — Timing begins with a 1 MHz crystal oscillator. An RC oscillator was sufficient for most calculators of the era. Use of an acoustic delay-line memory as in the 163 creates a requirement of greater frequency stability, hence the crystal.
- Bit-slice timing — The master clock triggers a toggling flip-flop to produce the 500KHz, 50/50 duty-cycle signal CPR. The basic bit period of the system - CPA - is sliced in two to interleave bits in the delay line. The reading of these sliced bits from the delay line is timed by CPR.
- Bit clock — CPR triggers another toggling flip-flop to produce CPA. CPA is the primary bit clock for the serial bit streams of the system.
- Bit-of-digit — CPA clocks a 4-state counter to produce a cycle of 4 bit periods which will define a decimal digit. The periods are TB0,1,2,3 though only TB0 and TB3 are needed as actual signals.
- Digit-of-number — The –edge of TB3 represents the end of one digit and the beginning of another. TB3 clocks a 5-flip-flop ripple counter configured for 17 states. The 17 states define the digits of a number, TD0::TD16.
- State cycles — The architecture will process 5 numbers in serial in coordination with the delay-line memory. These 5 numbers are counted out by a 5-state ring counter producing the signals SCT0::4. These signals define in time 5 registers containing numbers of 17 decimal digits each.

  A full cycle of these 5 number-registers constitutes the period of a state cycle for the state machine controlling procedure execution.

Number Registers & Delay Line Organisation

6 registers for storage of numbers are provided. Each register is 17 decimal digits of 4 bits/digit. Two of these registers are temporaries not seen at the user level.

The magnetostrictive acoustic delay line and a few flip-flops provide the storage for the digits of these registers. The decimal-point-positions and the signs of the registers are not stored in the acoustic delay line. Rather, the 5-bit PA,PB,PM flip-flop shift registers are used to store decimal point positions. Similarly, the signs of the registers are stored in a bank of flip-flops.

• Number registers The 6 registers are:

Register	Use	Processing
RB	displayed operand	SCT0,1,2,3,4
RA	previous operand/result	SCT0
RC	temporary (div, root)	SCT1
RD	temporary (mul, div, root)	SCT2
RM1	Memory 1	SCT3
RM2	Memory 2	SCT4

• Decimal encoding The decimal digits are encoded using 1242 bit weighting [fig A4.M5], rather than 1248. The second 2 bit is referred to in this commentary as the W bit (tWo) to distinguish it from the first 2 bit. With 1242 encoding subtraction is a simple matter of binary-complementing the subtrahend and adding an initial 1 via the adder carry input.

• Abstraction [Fig A3.M1] presents the registers as they are functionally seen at the procedure level. Implementing this would present the detriment of requiring two delay lines and keeping them synchronised.

• Practice Rather, this model is implemented in one delay line with two write heads on the line, as presented in [fig A3.M2]. One write head produces a long loop for registers RA,RC,RD,RM1,RM2. The 2nd write head is located ~ 4/5 of the length down the line from the first to produce a short loop for RB. The bit injection is timed such that the bits of RB are interleaved with bits of the other registers as they go by the B write head.

The data contents shown in [fig A3.M2] are that of the delay line and the flip-flops in the register bit paths during the idle state. Each 1-bit storage element is labeled with the data bit contained in that element at bit TB0 of digit TD0 of number cycle SCT0.

• Data requirements The 6 registers present a requirement for 6•17•4=408 bits of storage. The delay line provides for storage of 671 bits but as a consequence of the organisation 271 bits are not utilised (shown in the diagram as empty bit slots), leaving 400 bits utilised. The remaining 8 bits are provided by the AD0 and BD0 flip-flops, the 4 flip-flops of the arithmetic sum correction, and the 4 flip-flops of the BD register. This provides 2 extra bits, accounted for by overlap in time of bits being written and read to the delay line.

• Cycling The B register data cycling on the short loop 1/5 the length of the 5-

register long loop exits the delay line during every number cycle. The other 5 registers having a much longer cycle only show up during one particular number cycle of a state cycle (per table above). Over the course of a state cycle then, the B register is interleaved with all the other registers in the last 5th of the delay line and operations can be performed between the B register and any of the others by choosing the appropriate SCT period during which to execute the operation. Referring to [pg 17], notice the difference in the load resistors for the WA and WB write coils to produce a stronger pulse for the longer delay.

- Timing

  [Fig A4.M3] presents a timing diagram for two data bits injected into the delay line. One bit at the WA input (A1) produces current in the WA write coil (A2) to inject a pulse at the beginning of the delay line. When this pulse has traversed the delay line it produces a read pulse (A3) which sets the M flip-flop (A4) and is then transferred to the AD0 flip-flop (A5). After shifting through the 4 AD register flip-flops (arithmetic sum correction) it is injected back into the delay line (A6 and A7).

  A similar sequence is followed by a bit injected via the intermediate WB write coil (Bx).

- Read window

  As shown in [fig A4.M4], the M flip-flop is reset at the beginning of every bit-slice period as delineated by CPR. A bit pulse exiting the delay line then has a 2μS window to set the M flip-flop during the appropriate bit-slice period. The longer delay is 1341±1μS (671•2 = 1342μS). The shorter delay for the B register is 255±1μS (128•2 = 256μS).

- Shifting down

  The 4-bit BD and AD registers are normally part of their respective storage paths. Removing them from their normal path shortens the delay by one digit, moving subsequent digits to be earlier in the delay, e.g a digit in TD3 is moved to TD2. The number in the register is thus shifted down/right one digit.

- Shifting up

  The 4-bit SD register can be inserted into either the short or long register loop. This lengthens the delay by one digit, moving subsequent digits to be later in the delay, e.g a digit in TD3 is moved to TD4. The number in the register is thus shifted up/left one digit.

Adder

• Serial Adder — The adder proper is a straightforward serial adder comprised of two half-adders (XNOR with appropriate inversions) and a carry flip-flop.

• Decimal Correction — The raw sum coming out of the serial adder must be normalised for the decimal 1242 encoding. A benefit of the 1242 encoding is the carry in the raw sum is correct, thus there are no cascading inter-digit consequences to normalising, and each digit can be treated in isolation.

The basic concept to the normalisation is to store each digit of 4 bits as it emerges from the adder, then at the end of the digit assess it's value for valid encoding and correct any incorrect bits.

As the bits of a digit are shifted into the Sn flip-flops, OK is asserted by TB3=0, and the bits shift in without alteration. At the end of the digit with TB3=1, the AND-NOR gates feeding [8106.11] assess the digit value.

If all is good, OK remains asserted and on the 4th bit-shift the bits remain unaltered.

If the digit value needs correcting, OK is not asserted, and OK~=1 gates the correct bit-values to the flip-flops to be captured on the 4th bit-shift. All the correction for a digit is performed in the single shift, and the corrected digit subsequently shifts out with OK=1 as the next digit shifts in.

Position Counters

A set of four 5-bit shift registers constructed from flip-flops is used to store decimal-point positions and track procedure loop iterations. While shift-registers in implementation, in use they serve as counters.

• Registers — The 4 position counters are:

Register	Use	Processing
PB	DP for register B	SCT0
PM	DP for M1 & M2	SCT1
PL	loop counter for mult.	SCT2
PA	DP for register A	SCT3

• Arrangement — The 4 shift registers are arranged in a 20-bit serial loop with their own serial adder, and gating for injection and routing. The counters are binary, not decimal, so there is no decimal correction for the adder.

• Clocking — The counters are clocked in 4 bursts of 5 pulses of the CPA bit-clock. The 4 bursts occur at the beginning of SCT0,1,2,3; there is no burst during SCT4. The movement of the particular counters is thus synchronised to number cycles and a full loop synchronised to state cycles.

The burst production begins with generation of 4 enabling windows. Flip-flop FF asserts at the beginning of number cycles by the –edge of

TD16 – except at the beginning of SCT4 because (SCT3=>FF.SE~)=1 prior to the edge. FF continues to assert until TD0=0 prior to the –edge of CPA [5206.11], which is after the first bit of TD1 and a net period of 5 bit-clocks. The FF windows are then gated with CPA to produce burst-clock CPC to clock the position counters.

• Idle

The point counters loop does not rotate during idle [5210.6]. During idle, stationary PB data is compared with the digit-timing counter as it cycles for multiplexing the display [3302.12 et al], when they are equal the Nixies decimal point line is enabled [LP], illuminating the appropriate DP.

• Operations

As with the number registers, performing an operation on a position counter is done by making a selection of an SCT number cycle during a state cycle, as listed in the table above.

Note the SCT ordering differs from that of the number registers, e.g. PA is processed during SCT3, but RA is processed during SCT0.

Most of the operations on the counters are a simple increment by adding 1, such as when a numeral is entered after the decimal point during number entry [4205.6], where TP0 (pulse during first bit of number, TD0.TB0) is the addend 1.

Sign Flags

Five flip-flops are present for management of the sign of numbers and control of add vs. subtract.

• Number signs

Four flip-flops correspond directly to an associated register:

SA, SB, SM1, SM2

• A/S control

The fifth flip-flop – SK – controls whether an arithmetic operation through the adder will produce the sum (SK=0) versus the difference of the operands (SK=1). The value of SK is determined through a sequence of toggling based on the requested operation (Add/Subtract) and the signs of the operands.

• Form

Four of the flip-flops – SB, SM1, SM2, SK – have the ability to toggle and can be synchronously reset. SA can only be loaded from SB. SB also has the ability to be loaded from SA.

Control State Machine

The Control State Machine (hereafter: the Machine) sequences most of the procedures of the calculator: number entry, multiplication, etc.

- State register — The current state is held in 4 flip-flops [FCC1,2,4,8]. Of the 16 possible states, 12 are used. States 4,5,6,7 are not used. The binary value in the registers is decoded to distinct signals CC00, CC01 ... CC17. These state numbers are presented in octal, to maintain consistency with labeling on the PCBs. “CC” perhaps meant Control Code.

- Clocking — The state register is clocked on the –edge of SCT4. State transitions are thus synchronous to a full 5-number-cycle loop of registers through the delay line.

- Transitions — The next-state logic determining state transitions is comprised of AND-NOR logic feeding the SE~ and RE~ inputs of the state register flip-flops. To transition from one state to another, the AND-NOR logic of each SE~/RE~ input will determine which bits of the state register will be set, reset or not change. E.g. to transition from state 15 to 16, the AND-NOR=>FCC1.RE~ must assert 0 to reset the 1-bit , and AND-NOR=>FCC2.SE~ must assert 0 to set the 2-bit.

  If none of the AND-NOR gates asserts 0, none of the state-register bits changes and the current state loops for another state cycle.

- Two sequences — Rather than implementing different state sequences for the various procedures, there are fundamentally only two sequences: a short sequence for simpler procedures such as numeral entry, and a longer sequence for the arithmetic procedures. Heavy use of flags and additional gating varies the behaviour within a sequence during a particular procedure.

  The two sequences are distinguished by the 8-bit of the state register (FCC8): the short sequence uses states CC01::03, the long sequence uses states CC11::17.

Procedures

A procedure is a sequence of actions initiated by a key-press: e.g. numeral entry, clearing M1, subtraction, etc.

Most procedures invoke the Machine to perform a sequence of state cycles. Some of the simpler procedures can be completed without invoking the Machine (the Machine stays looping in CC00). Procedures can be classed into 3 groups:

- ST (execute during FST assertion, Machine not invoked)
- Short-sequence
- Long-Sequence

State Graphs for procedures are presented on [pg A5,A6,A7].

Idle

At idle, the Machine loops in state CC00.

• Display — RB is displayed in the Nixies via multiplexing. Bits from RB feed into SD as they exit the delay line, digits are captured in parallel from SD by the 4-bit display latch and decoded to 1-of-10 to drive the Nixie cathodes. Note there is a 1 digit-time stagger for illumination of the Nixies (anode activation), e.g. digit 0 (the LSD or right-most Nixie) is displayed during period TD1, not TD0.

Procedure Start & End

A timing diagram for a sample execution of the Subtract procedure is presented on [pg A8]. The first portion of this exemplifies the actions of starting a procedure execution.

• Initiation — Pressing any momentary-action key asserts KST~.

• Debounce — The leading edge of KST~ assertion triggers keyboard debounce monostable FKD. FKD asserts for ~ 4.5 mS.

• Sync — Flag FST synchronises the FKD assertion to the internal timing. FKD assertion enables FST to set 1 for an integral number of cycles relative to the +edge of SCT4.

The simpler operations which do not need the Machine are performed while FST asserts.

• Machine invokation — The –edge of FST triggers the set-clock input of flag FU. The FU set-enable~ input is 0 for procedures requiring the Machine, enabling FU to set 1 on the –edge of FST. FU resets on the –edge of SCT4, so is only high for the one SCT4 period.

FU being asserted during SCT4 directs the Machine to leave CC00 (idle) for some other state on the –edge of SCT4, the target state being determined by the procedure being invoked.

For procedures not requiring the Machine, the FU set-enable~ input is 1 to suppress assertion of FU.

• End — Three signals are generated to provide indication of the end of procedures.

ENDSP – End of Short-sequence Procedures
ENDLP – End of Long-sequence Procedures
ENDP – End of both Long and Short Procedures

These are used for returning the machine to CC00 Idle, as well as clearing flags.

Clear, Clear Indicator, Clear Memory

The Clear procedures perform their actions during FST. Machine invokation is suppressed [3112.2].

• Clear, Clear Ind.
- Clear B:
  - 0=>RB [7107.13]
  - 0=>PB [4301.6]
  - 0=>SB
- Clear pending multiply or divide. 0=>FMQ, 0=>FDQ
- Clear overflow state. 0=>FOVF
- 0=>FJ1

• Clear
- Clear FMLT, FDIV. 0=>FMLT, 0=>FDIV
- Assert END~ and END2~ [6204.6]:
  - Clear FN, FRT, FME. 0=>FN, 0=>FRT, 0=>FME
  - Machine to CC00. [0=>FCCx.RE~]
  - Clear SK. 0=>SK

• Not everything — Note that Clear does not clear everything. At the user level this is seen with the Memories RM1 and RM2, and the 'last operand/result' in RA which remains retrievable with the RV key after Clear. Internal registers RC and RD also are not cleared.

• Clear Memory 1, 2
- Clear according Memory register:
  - 0=>RMn [7101.6,8]
  - 0=>PM16 [4301.8] (Clear 5th bit of DP of RM1,2)
  - 0=>SMn
- Clear overflow state. 0=>FOVF

Numeral Entry

- Initial — The pressed numeral key is encoded into 1242 signals KW1,2,3,4. K(N+P) is asserted indicating a numeral key has been pressed, and KST~ asserted. After FU is asserted, the Machine heads to state CC01 [6202.8] to execute the short state-sequence.
- CC01 — nil
- CC03 — nil
- CC02 / insert & shift — During SCT0, the RB bit-stream is rerouted to insert SD between BD0 and BDW [7212.11,5213.6].

  Just after the beginning of the SCT0 number cycle, but before the first bit-period has completed, the 1242-encoded numeral of the keypress is parallel-loaded into SD [2303.8]. Note the inclusion of TP0 and CPW on this gating. The inclusion of CPW ensures the load is exclusive in time of the serial bit-shifting.

  The inclusion of SD lengthened RB by one digit at the beginning of the number cycle. The new numeral in SD is thus inserted as the LSD of the number and the pre-existing digits shifted up one digit during the SCT0 number cycle.
- CC02 / new or not? — After an operation the result is in RB for display. If the numeral currently being entered is the first of a new number, the pre-existing number in RB must be cleared. Flag FN is used to distinguish this situation.

  FN=0 indicates RB contains a pre-existing number. During the SCT0 insert-and-shift, the input to SD of the existing number from the delay line (BD0) is suppressed [FN=>5213.6] so that number is not recycled into RB.

  {elaborate: B to A}

  At the end of SCT4, FN is set 1 to indicate RB is now holding a new number, to which more digits may be appended [3113.8].
- CC02 / adjust DP — If the DP key was pressed earlier during entry of the number, then the DP position must be incremented.

  FJ1 was set 1 earlier to record the DP press. During the first bit (TP0) of SCT0 of CC02, a 1 is injected into the Y input of the Point Counters adder [4205.6]. As PB is the Point Counter being processed during SCT0, this serves to add 1 to PB, accomplishing the increment.

DP Entry

• Indication — During number entry, FJ1=1 is used to indicate the DP key has been pressed and any numerals subsequently entered are fractional digits.

• First — If the DP key is the first keypress of entering a new number, then the saving and clearing of RB must be performed, as with entering a numeral first. The short state-sequence is invoked and the same actions of numeral entry performed, in essence entering an initial zero.

FN is set 1 to indicate a new number is being entered [K(N+P+RM)=>3313.9]. At the end of a subsequent number cycle FJ1 is set 1 [5213.3].

• Not first — If the DP key is pressed while already in the midst of entering a number, it is merely necessary to set FJ1 to 1. As FN=1 by this time, this will occur during FST by, again, gate [5213.3]. There is no need to go through the state-sequence, so Machine invokation is suppressed [3313.6].

Undo Entry (Shift Right)

• CC01 — RB is shifted down one digit. BD contains the MSD of RB, input to BD is cutoff [7203.6] but BD is allowed to inject the MSD into the delay line during TDO. After TDO, BD is clear so injects nothing, but BDO injects into the delay line [7105.8] The LSD is thus eliminated and the bistream being 4 bits short, digits are shifted down one.

[7203.6] also enables PB to be decremented by adding 1F, but only if FJ1=1 from the DP key having been pressed earlier [4203.3]. If the DP key was pressed but no numerals entered then PB is still 0 and the decrement must be suppressed: FJ1 is cleared during FU, before CC01 starts [5212.12,5211.6].

• Nothing to shift — If a new number is not being entered, there is nothing to do: Machine invokation is suppressed [5113.11].

• Overflow — Entering too many numerals latches overflow [3110.6]. Undo-entry can clear overflow in this circumstance [3113.3].

Reverse Operands

• CC02 — The swapping of RA and RB is performed in one number cycle. During SCTO, RB is directed to the adder Y input while RA is suppressed from its default path to the X input [4202.2], with the corresponding routing of RA to RB while suppressing the default RB bitstream loop [7107.2].

Also during SCT0, PA is cycled into the data position of PB by suppressing PB1 into PX and enabling PA into PX [4108.3]. Simultaneously, PB data is cycled into the PA data position by suppressing PA into PB and enabling PB1 into PB [enBtoA,4106.11].

The signs SA and SB are swapped [enSAtoSB,enSBtoSA].

Recall Memory

• CC02 — During SCT3 for M1 or SCT4 for M2, RMn data is directed to the BD input while suppressing the default RB bitstream loop [7205.6,7205.12].

During SCT1, PM data is cycled into the PA data position by suppressing PA into PB and enabling PB1 into PB [4208.8].

During SCT0, sign SB is toggled if it differs from that of the source Mn [5201.6,5201.12].

DP Alignment

A DP alignment process is used in several procedures. The procedure execution example of [pg A8] is further examined in the following explanation.

• Objective — The objective of DP alignment is to shift a register to bring it into alignment with the current setting of the DP Location switch. The needed count of digit- shifts must be performed, a decision must be made to shift up versus shift down, and corresponding adjustments to the associated Point Counter made.

• States — The alignment process may occur in 6 states. The state determines the register aligned:

State	Register	Shift	PC	Test	Adjust	
CC11	RM1 & RM2		SCT3,4		PM	@end
SCT1	SCT1					
CC01	“	“	“	“	“	
CC12	RA	SCT0	PA	@end SCT3	SCT3	
CC13	RB	SCT0	PB	@end SCT0	SCT0	
CC03	“	“	“	“	“	
CC14	RA,C,D	SCT?	PB	@end SCT0	SCT0	

• CC13 / align RB — In the example, RB needs to be shifted up twice. Examination will begin with this (CC13.0, CC13.1, CC13.2):

• Looping — CC13 loops shifting RB up one digit until RB is equal to the DP switch setting DL. FC0 is used as a catch-and-hold flag indicating when alignment has been reached. FC0 is cleared at the end of every state cycle [4105.8]. During iterations of CC13, FC0 will be set 1 when DL=PA [4102.2]. The assessment of equality is done at the end of SCT0 [TD16=>FC0.T, 4303.6,DPALIGN=>4102.1] as the data of PB has been rotated into the flip-flops of register PA at this point in time. Once FC0=1 at the end of the state cycle, FCC1 is cleared [FC0~=>6204.9,6212.8], exiting the CC13 loop and taking the Machine to CC12.

• Up vs. down — The FRS flag indicates the direction of shift. As with FC0, PB data in PA is compared with the DP setting at the end of SCT0. If the DP switch setting is lower than the PB data, FRS is set 1 indicating a right-shift (down) is needed [DL<PA=>4102.9].

Once FRS=1, it remains set until alignment is achieved, being cleared in

the same action of clearing FC0 after FC0=1 [4105.8].

- Delayed action — The assessment of whether to shift may not be available in time to perform the shift in the current number or state cycle. To account for this, FJ2 is used to catch-and-hold the FC0 assessment, and the shift suppressed until FJ2=1 at the next appropriate number cycle. FJ2 having been cleared prior, if FC0=0 (shift needed), FJ2 is set 1 at the end of a state cycle via [5206.3,5209.6].

  In consequence, there is an extra first state cycle to the number of shifts needed, as in the example where 2 shifts were needed but 3 state cycles executed.

- Shift — A shift down is accomplished by removing the 4-bits of BD from the RB bit path [7204.8]. A shift up is accomplished by adding the 4-bits of SD to the RB bit path [7204.6,5211.11]. Note the use of FRS versus FRS~ at inputs to two of these gates to select direction. Also note FJ2 is a conditional to these data-routing gates, so no shifting take place until FJ2=1.

- Point counter — During the shift-loop iterations, counter PB is also adjusted. An increment is accomplished by adding 1 [4104.12], the 1 being TP0. A decrement is accomplished by adding 1F [4104.8]. The gate here is enabled for longer than 1F but the point counter clock is only enabled for the 5 bits.

- CC12 / align RA — From CC13, execution continues to CC12. RA is already DP aligned, so FC0 is set 1 during the 1st CC12 iteration, directing the Machine to proceed to CC16 [6205.5,6208.8], with no shifting being performed. Note that this time FCO is set at the end of SCT3, not SCT0, as SCT3 is the test and processing period for PA.

  In this instance RA was already DP aligned, in other situations an earlier result in RA may require realignment if the DP switch has been changed.

- CC11 / align RM1,2 — CC11 is conditionally executed at the beginning of the long state-sequence. If switch settings are such that M registers are involved in the procedure - to wit: if accumulating into RM1 or counting operations into RM2 - then the RM registers must first be brought into alignment with the current setting of the DP Location switch. Both RM1 and RM2 are adjusted.

  These modes are enabled in the example. Looking back at the beginning of execution, (KA~=>6203.1)=0 or (KMN~=>6203.2)=0 suppresses setting of FCC2 and the Machine is directed from CC00 to CC11 rather than to CC13. No alignment is needed in this instance, so FCO is set 1 in the 1st iteration, and the Machine proceeds to CC13 by now setting FCC2 [FC0~=>6204.9,6211.6].

- CC01 / CC03 — See M2 Operations.

M2 Operations

The M2-Add and M2-Subtract keys invoke the short-sequence. For M2-Sum mode (sum a multiplier or dividend into M2), the summing into M2 is performed when the MLT or DIV key is pressed, also in the short-sequence. Both circumstances require performing DP alignment and arithmetic.

- FME indicator — The FME flag is set 1 to enable these actions during the short-sequence. RM1,2 will be aligned during CC01 and RB aligned during CC03. A 2-iteration arithmetic loop will be performed in CC02.

  FME is set 1 during procedure start at the end of the FU state cycle [3202.8], and cleared at the end of the procedure.

- DP alignment — With FME set, the immediate transition from CC01 to CC03 is suppressed [6204.11,6211.6], and must instead wait for FC0 to assert [6204.9] – that is, wait for any needed DP alignment iterations of CC01 to be performed.

  The same logic applies to the transition from CC03 to CC02 [6212.8].

- Arithmetic — FME suppresses transition from CC02 to CC00 until FJ2 is asserted [6207.3] to allow the 2-iteration loop of CC02. FME also enables RB into the adder Y input for the 1st iteration and complementing for the 2nd [7212.6]. SK is also toggled appropriately for add/subtract in the 1st iteration [5205.8].

- BUG — Entering numerals after M2-Add/Sub appends them to the existing number in display rather than starting a new number, even if DP align adjusted RB.

Addition & Subtraction

Add and Subtract use the long state-sequence. Add/Subtract commentary continues with examination of the example of [pg A8] from where it left off in DP Alignment, with transition to CC16.

- 2-iteration loops — The actual arithmetic is performed in CC16 and CC17. Both of these states will execute 2-iteration loops. The 1st iteration performs the intended arithmetic operation, the 2nd iteration will complement the result if it is less than 0.

  FJ2 is used as a counter for the 2 iterations. The state enters with FJ2=0, a 1st iteration performed, FJ2 set 1, a 2nd iteration performed, with FJ2=1 the state exits, and FJ2 reset to 0.

- CC16 / 1st iteration — CC16 will add/subtract RB to RM1 if Accumulation is enabled. The combination of the signs of the operands and the requested operation dictate whether to produce the sum or difference of the operands. The SK flag is set 0 to indicate sum, 1 to indicate difference {elaborate}.

  For difference, the 1s-complement/NOT of the subtrahend operand is gated to the adder [7208.12,7109.9] rather than the TRUE version [7208.6,7109.4], and an initial 1 injected to the carry input [6212.3,6].

FJ1 is used as a catch-and-hold flag to capture the sign of the result at the end of the number cycle [SUM<0~=>FJ1.SE~]. The carry output from the adder indicates the sign: if the function was subtract and ACA=0, the result was negative [7109.3=>SUM<0~].

FJ1 must carry the SUM<0 indication into the 2nd iteration. It is inhibited from being cleared via [5212.5].

- Set sign — At the end of the number cycle, with the result<0, the appropriate sign flag is toggled [5213.8], for CC16 Accumulation this will be SM1.
- Toggle FJ2 — At the end of the state cycle FJ2 is toggled to 1 [5209.6,5210.3].
- CC16 / 2nd iteration — If the result < 0, indicated now by FJ1=1, it must be complemented for presentation with the appropriate sign. The X adder input is switched from the normal TRUE data to NOT data [7209.8], and initial 1 injected to carry. In effect:

  0 - RM1 => RM1

  Note the input of FJ1 to select NOT data and the input of FJ2 to perform the complementing only on the 2nd iteration. If the result was >=0, during the iteration TRUE data cycles as normal.

  With FJ2=1 prior to the end of the state cycle, the Machine heads to CC17 [6209.12].

  At the end of the state cycle FJ2 is toggled to 0. This does not conflict with the state transition to CC17 due to the –hold time of the flip-flops.
- CC17 / add,subtract — CC17 will add/subtract RB to RA. The same principles of operation as described for CC16 apply. At the end of the 2nd iteration, the Machine proceeds to CC10 [6209.8,6209.6].
- CC10 / result & count — CC10 is another 2-iteration loop state moderated by FJ2.

  The result rests in RA. It must be moved to RB for display. RB is also moved to RA as the most recent operand entered. This is performed in the 1st iteration.

  In the 2nd iteration, if M2-Count mode is enabled, M2 is incremented [7202.8].

Queue Multiplication/Division

The 163 uses infix/algebraic entry. This requires storing the press of the MLT and DIV keys until the operation is actually performed after an = key is pressed.

- MQ, DQ — The MQ and DQ flags provide this storage. These flags are set promptly after the MLT or DIV keypress, even before FST assertion. When an = key is later pressed after the entry of the 2nd operand, MQ/DQ state is transferred to the FMLT/FDIV flags [3103.3,6]. MQ/DQ are cleared at the same time [3105.8].
- Short-sequence — While the flags have been set during procedure-start, the short-sequence is nonetheless executed. While it appears unnecessary, the short-sequence provides opportunity for M2-Sum mode actions to be performed, as described in M2 Operations.

Multiplication

To do.

Division

To do.

Square Root

The square root procedure implements a variation of a (once-common) manual method, similar to long division. The root under construction is repeatedly subtracted from the radicand with an increment to the root for each pair of successful subtractions. When an overdraft occurs, the radicand is shifted up two digits, and the root shifted up to make way for one new digit. A simple example for 4 digits of the square root of 2 is presented on [pg A9].

- State sequence — The long state-sequence is used. The state graph is presented on [pg A7].
- Registers & flags — The radicand will be copied from RB to RA, from where it will be shifted up into RC digit by digit for the processing subtractions.

  The root will be constructed in RD and copied to RB for use as the subtrahend from RC.

  FJ2 will be used as a modulo-2 counter tracking subtraction-pairs and overdraft-correction-additions, as well as the double shift-ups of the radicand.

  FJ1 will be used to record indication an overdraft occurred.

  PB will be used as a primary-loop counter to terminate the procedure once the desired number of digits have been resolved.
- Initial — The radicand is in RB. FJ2 has been cleared during CC00.
- CC13 — RC and RD are cleared.
- CC12 — The radicand in RB is copied to RA. RB is cleared.
- CC16 — *Subtract & Increment.*

  During SCT1, root RB is subtracted from radicand RC. The arithmetic carry-flag at the end of this subtraction indicates the sign of the result: if ACA=0, the result is negative, so FJ1 is set 1 to record an overdraft occurred.

  If the result>=0, all is OK, and at the end of SCT4, FJ2 will be toggled, and CC16 will loop for another subtraction. Prior to this however, a decision must be made as to incrementing the root. If FJ2=0, this is the first subtraction of a subtract-pair: root RD is incremented and copied to RB during SCT2. If FJ2=1, this is the 2nd subtraction of a pair: no increment is performed.

  If the result<0, FJ2 is toggled at the end of SCT4 and the Machine proceeds to CC17.
- CC17 — *Overdraft correction.*

  If the overdraft occurred on the 2nd subtraction of a subtract-pair, then both subtractions must be undone. The increment of the root must also be undone, in appropriate sequence as for a full subtract-pair only one of the subtractions included the root increment.

  FJ2 indicates whether a subtraction-pair was completed or not.

If FJ2=0, the pair was completed. During SCT1, root RB is added back to radicand RC, this addition includes the increment. During SCT2, root RD is decremented and copied to RB. At the end of SCT4, FJ2 is toggled to 1, and CC17 loops.

With FJ2=1, either at first entry to CC17 or during a 2nd iteration, root RB is added back to radicand RC. This addition does not include the increment, recalling that during CC16 incrementing of the root was suppressed if the first subtract of a subtract-pair produced the overdraft. At the end of SCT4, FJ2 toggles to 0, and CC17 shall exit. If PB=DL, the digits have all been derived and the Machine proceeds to CC14. Otherwise, the Machine proceeds to CC15 to continue the derivation loop.

Note that at exit from CC17, FJ2 has always been returned to 0, so always back in step with the subtract-pairing.

- CC15 *Shift up.*

  Radicand RA & RC are shifted up one digit as a unit (RA.MSD ==> RC.LSD). With FJ2=0, Root RB/RD is shifted up one digit, PB incremented, FJ2 toggled to 1, and CC15 repeats to shift RA & RC up again. With FJ2=1, FJ2 is toggled to 0, and the Machine proceeds back to CC16, to construct another digit of the root.

  At exit from CC15, FJ2 is again always 0, as two iterations of CC15 were performed.

  The toggling of FJ2 and the decision of next-state based on FJ2 are made on the same –edge of SCT4. The earlier pre-toggle state of FJ2 determines the next-state decision due to the zero-hold-time storage provided by F-F trigger capacitors.

- CC14 DP alignment.

- CC10 Root RD is copied to RB.

  If Accumulating, RB is added to RM1.

Servicing Notes

Poor Soldering

The 163 - along with other Canon models of the same period and construction - can be frustrating to keep functional due to multiple and randomly-occurring failures over time.

Canon seems to have had a period of poor quality control with their soldering. Many of the joints appear cold-soldered. Over time, or with mechanical stressing of the boards, solder joints may fail open-circuit or develop poor conductivity. Sometimes an actual crack can be discerned in the solder around a through-hole stub or an IC pin. Sometimes a resistance of some tens-of-ohms can be measured in the circuit trace. The failure behaviour may be intermittent, and may be sensitive to temperature variation.

Most of these failures are associated with the through-hole stubs connecting traces on the two sides of a board. Canon did not use plated-thru holes. The metal of the stubs did not always tin properly during soldering and the solder adhesion is poor.

Board Insertion

Inserting the boards into the backplane may be difficult if attempting to push the board straight in.

- Damage — The pins of the edge connector sockets on the backplane have a shallow angle off perpendicular at the top of the pin where the board first contacts the pin. If the board edge is not beveled adequately it may butt against the pins and bend or squash the pin(s) while being forced in, rather than spreading the pins to slide down beside them. One unit encountered did not have any beveling on these edges. This resulted in a pin in the backplane socket being bent and damaged.
- Beveling — The boards can be beveled with a file, taking precaution to support the board substrate along the 22-pin length of a connector to keep the board from flexing to avoid compounding solder-joint failures.
- Insertion — Insertion of the boards is assisted by starting them at an angle: apply force to one end just till those pins start to engage, then add force at the other end so the pins engage in sequence rather than all at one time. Then, once all pins are engaged, push the board evenly down for the remainder of the engagement depth.

Forcing State

The DTL ICs used in the 163 are in the class of current-sinking logic. The LOW state of an output is low-impedance to GND, the HIGH state is high-impedance to GND with some pull-up to Vcc. The majority of the gate outputs are resistive pull-up to Vcc, the exceptions being the 4553 ICs with totem-pole outputs.

The outputs of the R-pullup gates can be shorted to ground to force a circuit to the LOW state for testing of downstream logic. The inverse - pulling an output HIGH by connecting to Vcc - is not safe for the ICs and componentry.

- Example — Circuit G-X~ (7110.6 et al) is part of the delay-line long-loop, and active during the idle state. As an assert-LOW circuit, connecting it to GND would result in a stream of data-1s being sent into the long-loop which could then be traced through the loop to assist in locating a persistent-0s fault in the loop. Similarly, 8204.8 (S4) is an assert-HIGH circuit in the long-loop, connecting it to GND would send data-0s into the loop to assist in locating a persistent-1s fault. The same approach may be applied to diagnosing the short-loop (RB) and the point-counters loop.
- Test A — Connect G-X~ [e.g. 7A5,7110.6] to GND to inject 1s into the registers long-loop. The CM1 & CM2 lamps immediately illuminate. Pressing RV results in Overflow and brings all-9s to the display.
- Test B — Connect G-BD~ [e.g. 7B9,7107.8] to GND to inject 1s into the registers short-loop (B loop). The display immediately presents all-9s and Overflow.
- Test P — Connect G-PX~ [e.g. 4D10,4107.6] to GND to inject 1s into the point-counters loop. Pressing CLEAR results in Overflow and the MSD Nixie DP on.

———————

Revision Log

- 2024 Jan — Initial creation.

To Do — - Multiplication, Division, Square Root DP management and details
- Control Flags, Rounding

—— EOD ——