Division 6 - Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

SUBJECT: STATUS OF XD-1 INTERNAL LOGICAL DESIGN

To: N.H. Taylor, R.A. Nelson, and Group 62 Section Leaders

From: R.P. Mayer

Date: August 17, 1953

Abstract: A detailed logical design exists for most of the XD-1 internal computer, but some additional designing and some re-designing must be done before firm decisions can be made to use the design.

Note: These notes summarize discussions of joint IBM-MIT study of the problem over an extended period of time and represent the best engineering information currently available. Comments from all parties concerned will be appreciated. Decisions in final form will be given in writing by Lincoln Laboratory, Division 6, acting on behalf of the Air Force.

Important Note: Sections 1, 2, and 3, and the attached drawings illustrate a rather consistent system for the purpose of simplifying the discussion and tabulation of questionable items (Sections 4, 5, and Appendix B) and must not be considered to represent the current proposal.

Table of Contents

Introduction

The status of the XD-1 internal logical design as of about June 23 is shown on two drawings: "Second Rough Draft: Comprehensive Logical Diagram" (SD-54846-4) and "Second Rough Draft: Traffic Diagrams" (SD-37625-4), both of which are attached. These drawings show suggested designs as well as firm decisions, but do not indicate which decisions are firm.

This E-note explains the drawings and tabulates firm decisions, incompleted areas, etc. Recent proposals are discussed in an appendix rather than in the body of the note in order to avoid delaying its appearance any longer. These recent proposals are not yet shown on the attached drawings. The drawings will be brought up to date from time to time, but whether they will be made "official" or kept unofficial has not been decided. The official drawings for the system (the block schematics) are being standardized and are not discussed here.

Supplements to this note will bring the descriptions up to date from time to time.

"Internal" logical design here refers to the part of the system which does not include terminal equipment nor the parts of the drum system which communicate with the terminal equipment.

It is assumed that the reader has been introduced to the system by literature such as the IBM and MIT Quarterly Reports.

Section 1. Brief description of system as shown on the drawings*

The attached traffic diagram (see Introduction above) includes a "System Block Outline", which shows the registers of the system and the paths of information flow and selection. The vertical layout of the block outline matches the vertical layout of the traffic diagrams for the instructions. The breakdown of instructions into classes and variations, and the assignment of digits in a word, is shown beside the block outline.

The instructions shown are described in IBM Report IM21, "Preliminary Operator's Reference Manual", as revised June 3 and June 23, except that some minor differences are assumed (such as the inclusion of 3 se instructions instead of one). A proposed instruction, bor-branch on overflow, is parenthetically included.

---

* See Important Note on Page 1.

The traffic diagram shows the activity of the system block outline --(horizontally compressed to a single vertical line) at any instant of time. Different instants of time are shown running horizontally across the page. All instructions are shown, but in many cases different variations of a given class are shown occurring at the same time, the distinction between variations being indicated by balloons and notes. The numbers appearing randomly sprinkled on the drawing indicate which control line of the comprehensive Logical Diagram is responsible for the activity shown near the number on the Traffic Diagram. Each vertical wavy line reminds the observer than an input-output break can occur at this time, as illustrated after the ADD class and after bi. The computer timing of the break will start at a random time with respect to the timing of the drum circuits, so that the break must be designed to work regardless of the phase relationship between the drum and the computer. The diagonal lines shown during the breaks indicate these different phase relationships.

The attached comprehensive diagram (see Introduction above) shows, in compact form, all the logically necessary connections in the computer (except for the areas of incompleteness discussed in Section 4). In nearly all cases, the only difference between this compact form and a fully expanded form is that repetitive units are drawn as a single unit with "multiple cable" connections to it. A "single-line" connection to a repetitive unit indicates that this line is repeated in all such units (unless noted otherwise), which in some cases means all units are connected to the one line, and in other cases means that the line passes via a gate in each repetitive unit. The symbology is not fully standardized yet and may lead to temporary confusion in some cases.

A dot in a box represents either a gate tube or a diode "&" circuit. An open triangle in a matrix represents an "&" function, which will often be accomplished by an "&" diode at this point and another "&" diode at the preceding point, as shown in the upper right-hand panel of the drawing. (Note that there are 8 A-size panels on the drawing.)

The upper three panels of the drawing show, primarily, the generation of all commands. In general, each vertical line controls one command (numbered to correspond with the numbers on the traffic diagram): a class line selects the command pulse output ("CPO") numbers to be used; a variation line may also have to be selected; each CPO unit (gate tube) which is thus turned on will be pulsed by the time pulse number appearing inside the box representing that CPO unit, and the command pulse will result. The upper left panel also shows the Op Reg with its class and variation matrixes, Memory cycle control (for determining which memory-cycle of an operation should be performed, and including the Break synchronizer and Break flip-flop for providing "in-out break memory cycles"), Time Pulse Distributor, Clock, Clock Pulse Control (for determining the use, or suppression, of clock pulses, and including the IO Interlock), and some manual controls.

The lower left panel shows the parts of the input-output system which are concerned with the internal logic of the machine. Note that this includes drum position counters and comparers, equipment for marking drum position at the start point of a search for information, and "Ask Address" circuitry for suspending a drum process if the computer should manually be stopped momentarily.

The lower central panel shows memory and addressing equipment. The memory (in the upper half of the panel) includes a delay-line control for memory (for proper operation under manual control of time pulses), parity check circuits, and Test Memory (Toggle switches, flip-flops, and separate memory address register). The addressing equipment (in the lower half of the panel) includes, among other things, circuits which allow the Adr Reg to be used as a Step Counter for multiply, etc., as a selection switch for the new operation "so" (skip or operate, whose address specifies what equipment to sense or to operate), as a start-control for starting program timing ("PT") when the step counter is nearly finished, and as a divide pulse distributor for controlling the divide process.

The lower right panel shows the arithmetic element. Note that only one of the two arithmetic elements is shown (the other one is identical except where indicated otherwise; for example, the LA Reg can be cleared separately from the clearing of the RA Reg, so two lines are shown clearing the A Reg). Note that circuits are provided for sign control (including temporary overflow storage), overflow storage, end-around-carry control (including control of shift-left after end-around-carry and control of divide additions, including the final corrective step), and for all types of shifts. "Acc X" holds the number shifted off the right end of the accumulator during add. During each "add" or "subtract" step of divide, the resulting sign is complemented and placed in Acc S, to be cycled into B Reg 15 as the quotient digit.

Section 2. Illustrative explanation of one instruction*

Instructions are performed in units, twelve time-pulses each, called memory cycles. (Some instructions contain three memory cycles.) This makes the explanation of an instruction fairly simple even in the following example, which has been chosen in order to illustrate as many techniques as possible in one unified example. The example is the instruction tdv, twin and divide, with Ix Reg 2 specified. To illustrate an in-out break-out memory cycle, it is assumed that instructions previously performed have started the drum system in the writing mode, and that the drum system calls for a break-out cycle just as the divide process is about to start.

---

* See Important Note on Page 1.

Section 2.1*

Briefly, what happens is this (refer to the traffic diagram, upper left corner): A program-timing memory cycle obtains the instruction from the memory (Time Pulse 6) and adds Ix Reg 2 to the address (TP 9). Then an operation-timing memory cycle obtains the number from memory (TP 6), "twins" it by sending the left half to both halves of the A Reg (TP 7), and starts the divide process (TP 9). This process is carried out independently of the memory cycles, and is composed of 16 steps (each one similar to that shown in the balloon between TP 9 and TP 0). When the operation-timing cycle is completed, all time pulses in the computer stop to wait for the divide process. As the last step is being completed, a new program-timing memory cycle is started. It controls an extra, corrective, step of the divide (TP 1 to TP 5), and finishes the divide operation (TP 6) while it is obtaining the next instruction from memory (TP 6). But Before this Pr Tmg cycle begins, while the divide process is going on, and just when the time pulses were stopped to wait for the divide process, it is discover d that the drum has asked for a memory cycle. Consequently the time pulses are permitted to continue until the break-out is completed. This does not affect the divide process in any way. (The traffic diagram for the break-out is shown in the center of the upper right panel.)

Section 2.2*

Before explaining the instruction in more detail it is necessary to explain a few things about Memory Cycle Control and Clock Pulse Control (upper left corner of Comprehensive diagram). At the start of program timing in our example, the break FF (in Mem Cycle Control) contains zero, allowing the selected class matrix line to be "on". The "PT" FF contains zero, allowing the bottom matrix line, "Prog tmg", to be "on" and making sure that all matrix lines marked "OT" (operation timing) are "off". The "AB" FF contains zero, but this will not affect the tdy instruction. The result is that the program timing line will be "on", and one class line will be "on". The class line will control the final steps of the previous operation, which will not affect program timing and will not be described. The break synchronizer contains zero because the drum has not yet asked for a break. The IO Intlk (in Clock Pulse Control) contains "one" because the drum is hunting for an address. The pause, AE, semi-stop, and stop flip-flops all contain zero, allowing only the second gate (in the vertical set of four) to pass 2 mc pulses. These will operate the time pulse distributor, which will produce commands from the command pulse output (CPO) units which happen to be "on". (For example, the first three CPO units are always "on", and so time pulses 6, 10, and 11 will always emerge from them. See later sections for what these pulses do.)

---

* See Important Note on Page 1.

Section 2.3*

The following detailed description of the program-timing part of the instruction can be followed on both the traffic diagram and the comprehensive diagram.

On time pulse (TP) 0 nothing happens (i.e., no CPO unit produces a command). This allows the matrix to finish setting up (the PT FF was complemented only 0.5 μsec earlier).

The traffic diagram shows TP 1 reading the Pr Ctr to the M Adr Reg (CPO 28) and clearing M Buf (CPO 33), and shows, for instance, that the clearing of M Buf is completed by TP 2. These can be found on the comprehensive diagram as follows: immediately above each CPO unit is an abbreviation which suggests the instructions which turn on the CPO unit. Immediately above that is the CPO number. CPOs 28 and 33 can be found in the center panel. The diagram shows that CPO 28 is turned on only during program timing and is pulsed by TP 1. Its output disappears into a multiple cable and can be found emerging at the Pr Ctr and pulsing a set of readout gates which read the Pr Ctr to the M Adr Reg. This line is not labelled at the Pr Ctr, but the title at the CPO unit suggests which line it should be and where it can be found. (The M Adr Reg was cleared on the previous memory cycle.) CPO 33 is always "on", and actually consists of two separate gate tubes. We are here concerned with the one pulsed by TP 1. Its output disappears into a multiple cable and can be found emerging at the M Buf and clearing all FF's, including the parity FF.

The traffic diagram shows nothing happening on TP 2 (to allow M Adr Reg, and associated matrix, time to set up), and shows TP 3 starting the read process in the M Regs (CPO 22). On the comprehensive diagram, CPO 22 is shown to be always "on", and pulsed by TP 3. Its output can be directly traced to M Adr Reg. If the address specifies magnetic memory (the "test memory" FF contains zero; it is digit R 1 of the address, as shown in the upper right corner of the lower right panel of the traffic diagram), the pulse clears the Test Memory Adr Reg (thus disabling test memory if register zero permanently contains zero) and proceeds to the memory delay-line control, turning the read control FF on. (See section 3.13 for further discussion of Test Memory.) If the time pulse distributor were manually turned off at this time the magnetic memory timing would not be affected, because the delay-line control is not disturbed.

---

* See Important Note on Page 1.

Assuming the time pulses are not stopped, the traffic diagram shows that the next event is on TP 5, CPO 27, the adding of "one" to the Pr Ctr, and that the carry propagation may not be done until TP 6 and that the last FF will be stabilized by TP 7. The CPO 27 and the Pr Ctr count circuit can be found on the comprehensive diagram.

The traffic diagram shows CPO 14 and 23 active on TP 6. CPO 14 clears the Instruction Register (i.e., Op Reg plus Adr Reg). CPO 23 reads the M. Regs to the M Buf.

(The comprehensive diagram shows CPO 23 connected not only to the Prog Tmg line of the matrix, but also to the break out and to all "OT" lines except one of the STORE class lines. It is pulsed on TP 3 and its output delayed 1.5 μsec. This is to insure proper operation of the memory if the time pulses are stopped on TP 3.)

The traffic diagram next shows the read pulse terminated on TP 6.5 and, on TP 7, the M Buf read to M Regs and a parity count started in the M Buf. These events are controlled by the memory delay-line control, as shown on the comprehensive diagram. The traffic diagram shows TP 7 reading the M Buf to the Instr Reg, by CPO 15, which can also be found on the comprehensive diagram. On TP 7.5 the memory delay-line control starts the write process.

Nothing happens on TP 8 because the control matrix is being set up, to tdv for this example. The lower right panel of the traffic diagram indicates that tdv is in the multiply class and that variation digits 9 and 10 should both contain "1". On the comprehensive diagram, the class matrix will be setting up to the multiply class. This will turn on the top line of the control matrix (which will stay on even during an in-out break, to allow the dv process to continue) and the "PT" half of the multiply class line (because the "PT" FF suppresses the "OT" half). The bottom line of the class portion of the control matrix ("Prog Tmg") is not disturbed, and remains on. In the variation matrix, assuming digits 7 and 8 both contain zero, the top line ("0") will be on, and the four lines representing digit 7 = "0", 8 = "0", 9 = "1", and 10 = "1" will be on. The Op Reg (upper left corner of comprehensive diagram) also includes the Ix Ic Matrix, which will be set up to "2" in this example. The output of this matrix can be traced down to the Ix Regs.

The traffic diagram next shows CPO 40, on TP 9, allowing the Op Reg to select which index register to add to Adr Reg. This can be traced out on the comprehensive diagram, which shows CPO 40 connected to the "PT" line of the individual classes. (It will not be "on" for classes 4-6.) The command goes to each Ix Reg, where a single gate connected to the Ix Ic matrix determines whether that Ix Reg should be added to Adr Reg. The 0.5 μsec carry delay line in each digit column of Adr Reg will originate a carry, if necessary, on TP 10. The carry takes about 0.55 μsec to travel the length of the register, so the final FF may not be settled until 0.05 μsec after TP 0.

The traffic diagram shows that the parity FF in the M Buf is stabilized by TP 10, and CPO 33 causes a parity check on TP 11 while the memory delay-line control is causing the write process to stop.

The traffic diagram also shows that, on TP 11, CPO 33 clears M Adr Reg and CPO "0" asks if IO wants a break. It implies, but does not show, that TP 11 also changes from Pr Tmg to operation timing. On the comprehensive diagram, CPO "0" sends a pulse to sense the IO Break Synchronizer. Since the drum has not yet asked for a break, the synchronizer contains "0" and so the break FF will continue to contain zero. CPO 2 is shown being "on" during the multiply class, and so TP 11 emerges from there and complements the "PT" flip-flop. This turns off the Prog Tmg line and the multiply "PT" line, and turns on the multiply "OT" line for operation timing. TP 0 does nothing (wait for matrix to settle down), except that the memory delay-line control turns the disturb pulse "on" on TP0 and "off"on TP 2.

This completes the program-timing memory cycle. Note that the contents of M Buf and M Adr Reg are no longer required at TP 0, so that an IO Break at this time would not disturb anything. (The break is described in a later section.)

Section 2.4*

The following detailed description of the operation-timing part of the tdv instruction assumes that the reader is now familiar enough with the drawings to work his own way at least as far as TP 7. Note that CPO numbers are not shown, on the traffic diagram, for commands which occurred in Pr Tmgs; the required connections ARE shown on the comprehensive diagram.

It is also assumed that the reader is now familiar enough with the drawings to be able to follow most of the discussion on both drawings without being told which one to look at, or how to trace the lines.

CPO units 72, 73, and 74 control the transfer from M Buf to A Reg, the "&" diodes (open triangles) on the variation lines on digit 10 selecting either CPO 73 or 74 depending on whether a "twin" variation was specified. Note that this same circuit works regardless of whether the instruction is in class 0,1 or 3. Note, also, that this circuit specifies the only difference between tdv and dv.

CPO 31A sets the Adr Reg for counting the proper number of steps in a divide process (multiply class, divide variation). There are 16 steps, and each step requires four counts (since the Adr Reg is distributing all these counts, as explained later). The setting is accomplished by setting CR clearing each FF of the Adr Reg.

---

* See Important Note on Page 1.

CPO 31A also performs an "Acc and B Reg sign". This pulse goes to both arithmetic elements, and its action in each is illustrated by its action in one of them: if the Acc is negative, the Acc, B Reg, and Sign Control FF are all complemented. This makes the 32-bit dividend (in Acc and B Reg) positive, and indicates that the quotient must later be made negative.

CPO 92 performs an "A Reg sign", which is just like the "Acc and B Reg sign" except that only the A Reg is involved, plus the sign control FF. If both Acc and A Reg were negative, the sign control FF now indicates the quotient should be left positive.

CPO 92 also clears B Reg 15 (of both B Reg's). This is because the "first subtraction" of the divide is omitted, since the quotient should be less than one. Clearing B Reg 15 at this time effectively puts a zero in the 2^0 position of the quotient.

CPO 21B performs a "start AE". This sets the AE (arithmetic element) FF of Clock Pulse Control to "1", allowing 2 mc pulses to flow (unless stopped by "semi-stop" or "stop") to the CPO units. They are connected to all CPO units labelled by a "2 mc". The command occurs on TP 8, but the first 2 mc pulse occurs on TP 9. The action of these pulses on the Adr Reg and arithmetic element is explained in section 2.5. This action is independent of the rest of the memory cycle, which the reader can trace to TP 11, at which time CPO 4 causes the "Pause" FF to turn off the Time Pulse Distributor and turn on a gate tube which repeatedly senses the break synchronizer. This allows the IO system to obtain a break memory cycle while the divide process is continuing. In this example, it is assumed that the drum asked for a break on TP 8, so that CPO "0", on TP 11, would discover this request and set the break FF to 1. The break FF would suppress the repeated sensing of the break synchronizer and supply pulses to the time pulse distributor (which would not actually stop, in this particular case, until the end of the break cycle). The time pulse distributor would produce a break cycle, and not interfere with the divide process, because the break FF has suppressed all class matrix lines except the two at the top of the control matrix, and has turned "on" one of the break lines of the control matrix. The break is described in section 2.7. CPO 2, on TP 11 of the operation-timing cycle, will complement the PT FF, setting it to the "PT" condition in readiness for the completion of the divide process.

Section 2.5*

This section describes the action of the 2 mc pulses supplied to the CPO units during the divide process. Only CPO 32 and 32A will pass 2 mc pulses during divide. CPO 32A senses when the process is done, and will be discussed in section 2.6.

---

* See Important Note on Page 1.

The CPO 32 subtracts one from the Adr Reg (being used as a step counter) every half microsecond. The first two digits, R 14 and R 15, are connected to distribute divide pulses in the following ways When they both contain "1" (11 = "3"), there is no carry (subtracting count), and so a divide pulse emerges on only the "3" line. When they contain "10" ("2") a carry pulse enters digit R 14 and also emerges on divide pulse line "2". When they contain "01" ("one"), no carry occurs and no divide pulse will emerge. When they contain "00" ("zero") a carry pulse enters digit R 13 and also emerges on divide pulse line "0". These three pulses emerge in sequence from the Adr Reg whenever CPO 32 in running, but CPO 47 allows them to do things only during divide.

The traffic diagram shows only one sequence of three divide pulses (four counts), indicating the action in the arithmetic element on each count of the Adr Reg (3, 2, 0). This represents one of the 15 steps of the divide process. Since the procedure is the same on each step it is sufficient to discuss just the first step, which assumes that the "first subtraction" has been performed, and corrected, and that a zero digit exists in the 2^0 position of the quotient. (Also, the discussion refers to only one arithmetic element.)

Divide pulse 3 calls for "dv set EAC" and "dv sl". The "divide set end around carry" discovers whether the previously-calculated remainder is positive or negative (Acc S). If it is positive, a subtraction should occur next, and so the A Reg is made negative (complemented if it is positive) and the end around carry control FF is set to "1" (for the reason discussed in the next paragraph). If the remainder is negative, the reverse is true. The "divide shift left" part of this divide pulse shifts left the remainder (in Acc and B Reg) to position it for the following subtract or add, and shifts left the quotient (in B Reg) to make room for the next quotient digit. This can be found on the comprehensive diagram as follows: the divide-pulse-3 output of CPO 47 connects to some of the outputs of CPO units 86 and 87. The titles of these outputs can be found near the CPO units, and also near the B Reg. The titles show what shift gates are pulsed although the drawing shows these lines merging (but not mixing in an "or" circuit) into a single line running to all shift-left gates. Acc.1 is shifted into the sign digit because it must be included in the following addition. All additions in the machine treat the sign digit like any other digit except for overflow control (which is gated out in divide), and EAC control (which is ignored, and handled separately in divide, as explained later).

Divide pulse 2 next calls for "EAC" and for "EAC 0". These pulses sense the end around carry control FF: if it contains "1", the "carry 1" input to the accumulator adder circuits is pulsed; if it contains "0", the "carry 0" input is pulsed. ("EAC 0" sets the EAC control

FF for a purpose discussed in section 2.6) The carry input to the adder initiates the addition of the A Reg to the Acc. If the A Reg is negative, a subtraction results. The A Reg was made negative by taking the "1's complement", but the EAC control FF was made to specify a "carry 1" input to the adder in this case. The result is to effectively add the "2's complement" whenever a subtraction is required. The remainder will be in true form (if positive) or in 2's complement form (if negative). At least three advantages result from using the 2's complement form (which is used only in the internal automatic divide process; the final answer, if negative, being in "1's complement" form): an end around carry is never necessary, and it is not necessary to add "1" to every bit of the divident which is in the B Reg when a subtraction is performed; and a zero remainder will be positive (resulting in a "1" quotient digit at that point and "0" digits thereafter). Note that 0/0 = 1/2.

The action of the adder circuits can be traced out by the reader (they are described in other literature, such as the IBM Quarterly Report dated April 27, 1953). A carry ("0" or "1") is propagated from digit to digit, requiring about 0.55 μsec to travel the length of the register. (The EAC control FF will remain set to "1".) The sum is placed in the Acc digits, displaced one place to the right (for speed in multiplication), the result of column 15 being placed in Acc X and not in BR S. Acc S is left "empty", and so is used temporarily for the quotient digit: "1" if the sign of the remainder is positive ("0") and vice versa (this circuit is actuated only for divide because Acc S must remain "0" during multiply). If the Acc S, 1, and 2 FF's have a resolution time of .45 μsec, then all FF's are stabilized one microsecond after divide pulse 2.

Divide pulse 0 occurs at this time. It performs an "add shift left", and places the quotient digit (from Acc S) in BR 15. The "add shift left" corrects for the displacement caused by the adder circuits. Acc X is shifted to Acc 15, Acc 1 is shifted to Acc S, and Acc 2-15 are also shifted (to Acc 1-14). All digits are shifted simultaneously because such a connection (compared with a "ripple shift") makes the cycle left instruction more convenient.

This completes one step of the divide process. Note that the result, at the end of each step, is that the remainder (positive or negative) is in digits Acc S-15 (plus BR FF's containing unused dividend digits), the quotient digit resulting from this step is in BR 15, and the EAC control FF contains "1". Since 16 quotient digits (not including the sign digit introduced by CPO 92) are desired, 16 steps must be performed. The step counter end carry (from Adr Reg) clears the AE FF of Clock Pulse Control, which stops the 2 mc pulses to the CPO units, and thus stops the divide process. (The exact method of generating the step counter end carry depends on the design of the shift instruction. This detail is not worked out yet. The end carry is shown occurring when the step counter contains "0", but it is assumed to occur when the step counter contains "1".) The divide process is completed by the next program-timing memory cycle, described below.

Section 2.6*

This section describes the final commands required by the divide process and which occur during the beginning of the program-timing memory cycle, which obtains the next instruction from the program.

CPO 32A passes 2 mc pulses during the performance of the steps of the divide process. Each of these pulses senses the Adr Reg (step counter) to see if the process is nearly done. The timing of the divide process is such that if the program-timing cycle is told to start when the step counter reads "1" there will be no conflict between the commands issued to the arithmetic element from the divide pulse distributor and those issued by the time pulse distributor. The "start PT" pulse from Adr Reg (sc = 1) will clear the "pause" FF of clock pulse control, which will stop the continuous sensing of the break synchronizer and will start the time pulse distributor (or will allow it to continue after the break if an IO break cycle is then in progress).

The PT FF has already been set to "PT" by TP 11 of the previous operation timing cycle, and so the Prog Tmg line of the matrix is on. The action of this line was explained in section 2.3. The clearing of the PT FF has also turned off the multiply-class "OT" line and turned on the multiply "PT" line. This line will remain on until TP 6 clears the operation register. Only CPO units 56 and 61 will be pulsed before that time, and they will finish the divide process as follows:

CPO 56, on TP 1, pulses the same line as divide pulse 0. This is because it is assumed that divide pulse 1 (which doesn't actually appear on a line) is coincident with the step counter end carry, which stops the divide process before the final step on divide pulse 0 can occur.

The rest of the acitivity on CPO 56 is concerned with correcting the remainder if it is negative (i.e., if the final subtraction is "too much", so that the A Reg must be added back in again).

CPO 56, on TP 1, performs "negative set end around carry". If the Acc is positive, nothing happens and the EAC control FF remains in a "1" (as at the end of every divide step). If Acc is negative, an addition is required, and so A Reg is made positive and the EAC Control FF is cleared to request an addition.

CPO 56, on TP 2, performs "EAC 0 or clear". If the EAC Control FF is on "0" the carry 0 line is pulsed and an addition occurs. Otherwise no addition nor subtraction is performed. In either case the EAC Control

---

* See Important Note on Page 1.

FF is complemented to request the next step (i.e., if an addition occurs the FF will be complemented to "1", so a carry 1 output from the sign digit to EAC control is not important.)

CPO 56, on TP 5, request EAC shift left. If no addition occurred, the EAC control FF will now contain "0" and so nothing will happen. But if an addition did occur, the FF will now contain "1" and so an "add SL" will occur, correcting for the shift that occurred during the addition and leaving the resultant remainder in correct positive form in Acc S-15. Note that if the division was exact (at any step), each succeeding step will result in a "0" quotient digit and "minus A Reg" as a remainder. The corrective step performed by CPO 56 will correct this remainder to plus zero.

CPO 61 corrects the sign of the result on TP 6. The sign control FF was previously set up to remember the desired quotient sign. If it contains "1" the "product sign" obtains the "1's" complement of the remainder and quotient in Acc and B Reg respectively, otherwise Acc and B Reg remain positive. The B Reg contains 10 quotient digits and no sign digits; the sign of the remainder after TP 6 is the proper quotient sign, and a shift left of 1 will provide the correct quotient.

This completes the tdv instruction, except that the example was stated as including an IO Break-out memory cycle, which will now be explained.

Section 2.7*

This section describes the in-out break-out memory cycle. The traffic diagram shows this cycle in the center of the upper right panel. The traffic diagram of the break is a little different from that of other cycles because it includes a "phase diagram" whose diagonal lines illustrate all phases of the drum timing with respect to the computer timing.

The phase diagram will be briefly described, and then a specific example will be explained in more detail.

All vertical lines on the phase diagram represent events which occur at a specific time with respect to the computer, regardless of the drum timing. All events controlled by drum timing are shown along the horizontal line just above the diagonals. The diagonals represent these drum-timed events as they would occur at various different computer times. Any horizontal line drawn across the phase diagram illustrates the timing of the drum and computer for the corresponding phase relationship between drum and computer. These phases can be identified by the computer-time at which the drum "asynchronous break" pulse occurs. Thus the phase diagram shows phases 10(at the top) through the second 11

---

* See Important Note on Page 1.

(at the bottom). It extends upward to 10 and downward to 11 because these regions are indeterminate. The upper phase 10 will presumably set up the "break sync." FF of "memory cycle control" in time for TP 11 to set up a break, but a phase later than 10 might not quite make it. Certainly a phase later than 11 (the upper 11) will not get a break until the next TP 11, as shown. These same comments apply to the indeterminate region between the bottom 10 and 11, except that here it is assumed that a phase later than 10 (but earlier than 11) might just barely get an immediate break. The shaded areas indicate when questionable things actually happen, and this will be explained during the following detailed example.

In the detailed example it will be assumed that the drum asked for a break on TP 8 of the tdv operation-timing cycle. Thus, the horizontal line for phase 8 of the phase diagram should be studied. This line happens to coincide with the line labelled "IO Sel" in the right margin. The fact that the traffic diagram shows a bi instead of tdv preceding the break is of no consequence because nothing happens until TP 11, and then the break begins and doesn't care what operation is being performed.

To help follow the action of the drum, the activity initiated by one drum timing marker ("drum sync") is shown in the center of the lower left panel of the traffic diagrams. In the following discussion it is assumed that the reader will refer to all three drawings: (1) the comprehensive logical diagram (the IO system is in the lower left panel), (2) the phase diagram (upper right panel of traffic diagram, the phase 8 line), and (3) the drum timing (lower left panel of traffic diagram, the upper half representing the write timing).

A pulse from the timing marker channel on the drum adds one to the drum position counter or reads the drum identification to the identification register and then senses the comparison circuits. (The "selected Drum sync" pulse will be described later.) An "Address Found" pulse emerges from the selected drum when a comparison checks. The "Ask Adr FF" will be on "1" at this time unless the computer has been stopped. The "address found" will thus clear the "Ask Adr FF" to prevent looking for another register unless the computer resets it to "1" (described later). (The reason for delaying the clear 2.5 μsec as described later). The pulse also adds one to the Drum Adr Reg and the IO Adr Ctr (but not the IO Wd Ctr because a csw instruction might be in progress), and (since wr is selected) immediately emerges as an "asynchronous Break," which happens to arrive at the break sync FF at computer TP 8 in our example.

Nothing happens until TP 11 senses the break synchronizer, sets the break FF to "1", and sends an "asynchronous Ask Address" to the Ask Address synchronizer. (If TP 11 occurred during the rise time of the break-synchronizer FF it must surely ask for another address immediately, if it is strong enough to set the break FF, because the

"ask address" is necessary for the break, as described later.) If an "IO End Carry" ("disconnect") has occurred, the IO interlock will not allow another address to be asked for.

On TP 3 the "address Found" pulse is emerging from the 3.5 μsec delay line and setting the write FF to write the word that was previously placed in the IO Reg. This writing stops on TP 7, and the IO Reg is cleared.

Meanwhile, the Break FF has turned off all class lines (except the upper ones which control the divide, multiply, and shift processes in the arithmetic element) and has turned on one of the break lines (the break-out line, since the read-write selector is on wr). The Break FF has also suppressed any continuous sensing of the break sync FF, and has demanded time pulses. Note that CPO 35 A clears the break sync FF on TP 1 so that the next async break can function on TP 2 (if phase 10 exists).

On TP 7, a "try M Buf to IO Reg" cannot get past the gate tube on the "Ask Adr FF" because it was cleared on TP 1.

On TP 8, the next "Selected Drum Sync" pulse finds the Ask Adr synchronizer has been set and so sets the Ask Adr FF. (It must try again in 1 μsec because in phases before phase 0 the synchronizer will not have set up yet. It must try at this early time because otherwise phases later than 9 would miss the last "try M Buf to IO Reg" pulse. In phases 11 through 0 it is not clear whether the Ask Adr FF will be set the first time or the second, but no pulses are hitting the gate tubes at this time.)

On TP 0, a "try M Buf to IO Reg" can get past the gate tube and will load the IO Reg for the writing that begins on TP 7 if the next drum position is the one desired. (TP 11, delayed, is used because TP 0 occurs while the control matrix is being reselected. The pulse must be as late as this because otherwise the transfer could not occur during phases later than lower 10. The TP 7 trial must occur because otherwise the IO Reg would not be set up soon enough during phases earlier than 2. During phases 6 through 7 it is not clear whether only the late, or both, trials will succeed, but it does not matter because "1" digits are being read to IO Reg and nobody is looking at IO Reg at this time. The Ask Adr FF must not be cleared earlier than shown because then no trials would succeed for phases earlier than 11.)

This completes the discussion of the in-out break. Note that, in our example (phase 8) the next async. break occurs on TP 0, and so TP 11 would find the break sync FF cleared and would clear the break FF. Since the divide process is still continuing, the clearing of the break FF would initiate the repeated sensing of the break sync FF.

The second of these pulses would find the sync FF set and would initiate another break, which would appear as phase 10 (lower). Since phase 10 (lower) requests the next break on TP 2 (after the break memory cycle), all breaks will be phase 10 until the divide process stops. If this happens during a break, then when the next TP 11 clears the break FF the repeated sensing of the sync FF will not occur, but instead the time pulse distributor will continue and will produce the program timing cycle which finishes the divide, and this will then be followed by a phase 2 break.

This completes the explanation of the tdv instructions, with an IO break-out during the divide process.

Section 3. Discussion of obscure features of other instructions.*

It is now assumed that the reader is sufficiently familiar with the drawings to work his way through all the instructions. The tdv instruction and IO break-out illustrate nearly all the techniques used in the other instructions. However, certain details may be confusing or entirely missed. An attempt will now be made to describe all such details.

It is assumed that the reader will work his way, time-pulse-by-time pulse across the traffic diagram, following the action on the comprehensive diagram. Details will now be discussed in the sequence in which the reader is likely to encounter them. The tdv and dv instructions should present no problems.

Section 3.1* (csw)

All commands for csw occur during program timing, so no operation-timing cycle is started (the next cycle is another program-timing one). This is accomplished by not complementing the PT FF (by CPO #2, 3, or 5) on the PT cycle of class 3 (unless et variation is selected).

Section 3.2* (Classes 0 and 3)

CPO 81 clears A Reg on TP 1 of both the OT and PT cycles. It is also needed on certain other instructions, but no attempt is made to suppress it on instructions which will not be damaged by it.

CPO 69 finds the positive magnitude of both Acc and A Reg for instructions cm and dm.

All negatives are in 1's complement form throughout.

---

* See important note on page 1.

The "CO in" clears the EAC Control FF, and a "carry 1" from the sign digit sets it to request an end around carry and corrective SL by way of CPO 48 and 50.

(The IO break-in is discussed in section 3.3)

The sign control FF is not used, as such, on this class. It is cleared by CPO 63 for use as a temporary overflow accumulator. An overflow is indicated by a carry into, but not out of, the sign digit (or vice versa) regardless of whether this condition occurs on the add or the end around carry. But no overflow occurs if there is both a carry 1 "out" on add and a carry 1 "in" to the sign digit during an end around carry (1.111 plus 1.000), and the sign control FF will recognise this non-overflow condition. The overflow, if any, is recorded in the overflow FF for future reference. (In the shift class, slo, the sign control FF cannot be used for overflow, but it is not needed because only one overflow pulse can occur.)

Section 3.3* (IO break in)

The magnetic memory strobe is omitted, and so the selected register is cleared without performing a parity check. The word to be stored is available before the memory cycle is started, so a stopping of the time pulse distributor at any time will not disturb the transfer.

Note that the "asynchronous break" for "read" occurs 4.5 μsec later (in the drum timing) than it does for "write".

Section 3.4* (Class 4, not including slo)

Note that the index register is not added in (on TP 9 of PT), and so the address in the Adr Reg can be used immediately for shifting. CPO 21A starts the arithmetic element on TP 9 (the earliest time at which the control matrix knows what instruction to perform), but the first shift pulse cannot occur until TP 10.

The shift-right pulses going to Acc do not go to all digit columns simultaneously. Instead, the pulse which goes to Acc 14 ("0" and "1" shift-right gate tubes) initiates a "ripple shift" of digits 14 through S (shifting them to 15 through 1). The "ripple shift" operates by letting the output of the shift gate tubes in each column be used as the command pulse for the next column to the left. (Acc 15 is pulsed alone and does not pulse Acc 14.) The "ripple shift" is used to speed up the multiplication process, as explained later, and is not used in B Reg nor in shifting left.

---

* See Important Note on Page 1.

The condition "Adr Reg (or Stp Ctr) = 5" is used to start PT. (CPO 32A also senses Stp Ctr, but Stp Ctr = 5 will occur before stp Ctr = 1.) It could have started 1/2 to 1 1/2 μsec earlier (the only requirement is that the Adr Reg be stabilized before clearing it, or cleared before reading into it) but the condition "SC = 5" is needed, anyway, for multiply.

(The clearing of A Reg on TP 1 is needed only for slo, but does not harm the other shift instructions.)

Sign control is not needed because the sign digit itself is copied in the proper place on each shift. Note however, that peculiar results occur when shifting a negative Acc left while B Reg is full of zeros.

Section 3.5* (aia)

The addition is already performed on TP 9 of PT, but CPO 41 (instead of 40) is used because the addition is not desired on other instructions in this class. (The timing of the commands to A Reg is made so late simply to match the timing of bi.)

Section 3.6* (Class 5, except bi)

Instruction blm, on TP 9, senses L Acc. If it is positive a "no branch" pulse appears. Similarly with brm and R Acc. Instruction bm is a combination of these two, so that if either (or both) Acc is positive a "no branch" appears. The suggested "bov" (branch on overflow) produces a "no branch" if both Acc Overflow FF's contain "0".

In any of these instructions, the absence of the "no branch" pulse allows the branch to continue as shown on the traffic diagram. But the occurrence of the pulse causes the command matrix to switch (from TP 9 to TP 11) to a condition which allows no more commands to occur except PT. A special "blank" OT line for the branch class could be provided for this, but the comprehensive diagram shows that the "no branch" changes the matrix to "store, cycle B" (described later), which has only program timing after TP 9. (The variation FF's are cleared on "no branch".)

The so instruction always provides a "no branch" because its only function is to provide a "sense" or "operate" pulse on TP 9. The function of this pulse is determined by the "so matrix" on the Adr Reg, and if a "sensed" condition is "on" the Pr Ctr is made to count once, thus skipping the next instruction of the program.

If a "no branch" occurs, the clearing of Adr Reg and R A Reg on TP 9 does no harm.

---

* See Important Note on Page 1.

In any case, TP 10 does nothing, to allow the matrix time to change.

The commands to the Adr Reg are for bi, and do not affect the other instructions in this class.

The commands to the Pr Ctr cannot occur earlier than TP 11 because the control matrix is not stabilized until then. The new contents of Pr Ctr are needed on TP 1. (TP 0 is not used because the matrix might be switching for an IO Break.)

Section 3.7* (bi)

The bi instruction produces a "no branch" if the selected Ix Reg is negative.

The commands to the Adr Reg are for the purpose of subtracting Ix Iv (the Index Interval) from the selected index Reg (via Adr Reg). The Adr Reg must be cleared on TP 9 because it was not known on TP 7 that the address from M Buf is not desired.

The complement of the Ix Iv from M Buf is read into the Adr Reg on TP 11. The Ix Iv does not cover the full length of the Adr Reg, so the command must inject "1" digits (negative zeros) into the remaining FF's of Adr Reg.

Section 3.8* (Class 7)

All instructions in this class sense the IO Interlock, and will not proceed until the previous in-out process produces a disconnect. This is accomplished by making CPO 4 call for a "pause" (as in dw), which continually senses the IO Bk Sync and allows IO break cycles to proceed, while the time-pulse-distributor is otherwise stopped to wait. Meanwhile, CPO 7 has requested a "start arithmetic element", but the resulting "2 mc" pulses emerge only from CPO 32 D and continually sense the IO Interlock FF to determine when to end the pause and resume regular operation. There is no need for turning off the "AE" FF of clock pulse control quickly, and CPO 6 eventually does this.

The instructions se, sea, sei, and lac then load either the IO selection register or the IO Adr Ctr on TP 1 and 2.

---

* See Important Note on Page 1.

The instructions rd and wr load the Wd Ctr (different CPO units are used because the wr changes to OT, while rd remains in PT), set up the read-write FF of IO control (which also selects between in-breaks and out-breaks), start IO (set the interlock and start counting one drum revolution), and ask the drum to find an address.

The write instruction must essentially provide an initial break-out to provide a word for the drum to write. This it does by going to an OT cycle. Since it is the only instruction in this class with an OT cycle the variation digits need not be sensed during the OT cycle. The essential difference between this OT cycle and a regular break out is that the IO Reg is cleared and read into on TP 6 and 7. This must be done because no previous drum operation will clear IO Reg nor interfere with the reading into IOR, and because a drum sync pulse might not occur until too late to accept a "try M Buf to IOR" on TP 0.

Section 3.9* (mu, tmu)

Each 2 mc pulse during multiply will sense BR 15 to determine whether to shift or add. If an addition is specified, the carry 0 input of the adder is pulsed. The adder is designed to add, shift, and carry in one step. The Acc 15 FF is pulsed immediately (by the Acc 14 adder) and is stabile and ready for the next activity in 0.5 μsec. The carry pulse takes time to propagate from digit to digit, and arrives at Acc S 0.55 μsec after Acc 15. The Acc S FF is stabile 0.5 μsec after that. The B Reg is shifted (all digits at once) at the same time that Acc 15 is add-shifted into Acc X and into B Reg S.

Half a microsecond after one "BR Sense" it is possible to have another one because the B Reg and Acc 15 are stabile by then and because the carry pulse will not arrive at other digits of Acc until they are stabile (i.e., the carry ripples down Acc at the same rate each time). This is also true even if the "BR Sense" produces a "multiply shift right", because the Acc shift-right circuit is connected to "ripple" at the same rate as the carry circuit.

Section 3.10* (Class 2)

Each store-class instruction, except st and ep, will leave one or the other half of the word in memory undisturbed. It is quite expensive to provide separate drivers for the left and right halves of

---

* See Important Note on Page 1.

memory, so both halves will always be handled. It is desirable to perform a parity check on the half that is to be rewritten, otherwise an error in that half would not be found. It is also necessary to perform a parity count for the new half that is to be written. Since there is only one parity bit and only one parity count circuit it is not possible to parity-count on half-words. Thus the above mentioned parity check and parity count must be performed by two independent full-word parity counts. There is not time to make these two counts between strobe time (TP 6) and write time (TP 7) of one memory cycle, so two memory cycles are required. Similar considerations apply to the ec instruction, so all class 2 instructions except st will have 2 operation-timing memory cycles. They will be called cycle A and cycle B.

These two cycles are selected by the AB FF in Memory Cycle control. The st instruction complements this FF so that it changes from A (the normal condition) to B at the end of program timing (CPO 35 c), thus skipping the A cycle and proceeding with B. The end of the A cycle switches to B and the end of B switches back to A but also switches out to program timing (CPO 2).

An in-out break must be allowed to occur between cycle A and B, and so the half-word to be re-written cannot be left in M Buf through TP 0. The full word is therefore transferred to A Reg (TP 7A) and the half to be re-written is read back on TP 2B. The R A Reg must not be disturbed on sta because the purpose of this instruction is to store the original contents of R A Reg.

There is not time, in rao, to perform an end around carry in the same way as in the ADD Class. The EAC control FF requests an EAC in the same way, but the request is granted simply by inserting a "1" in Acc 15. This is because the only way an EAC can occur in rao is when the original number is 1.11111 and the result should be 0.00001.

An overflow can occur if rao is performed on 0.11111. This will be recorded directly in the overflow FF because a second (corrective) indication cannot occur on end around carry.

Section 3.11* (slo)

Program timing is not started until Stp Ctr = 1 because extra time is needed, during program timing, for roundoff.

Sign control is used to simplify the roundoff process (i.e., so BR S is added to Acc 15 regardless of the sign of the original number).

Section 3.12* (ria and rir)

In ria the number to be read into the selected Ix Reg is placed in Adr Reg by program timing.

---

* See Important Note on Page 1.

Section 3.13* (other features)

This completes the discussion of obscure features which are likely to be encountered in working through the traffic diagrams. There remain a few obscure details on the comprehensive diagram. These will now be discussed.

Test storage is shown immediately below the M Adr Reg. A "Test Mem. Adr Reg" of 4 FF's is provided in order to keep test storage more isolated from magnetic memory. Each of the 16 outputs of this register can be switched to the corresponding one of 16 toggle-switch registers or to a single set of control gates on a single FF register. A single FF of M Adr Reg determines whether or not to use test storage. If it contains a zero, the "start mem" pulse will start magnetic memory and clear the Test Mem. Adr Reg, selecting test register zero. If register zero contains zero, then reading out will not disturb M Buf, and if register zero is toggle switches, reading in cannot disturb it. If the selection FF contains "1", magnetic memory will not be started, and the selected test register will remain selected. The "start mem" pulse will sense digits 2,3,4 of M Adr Reg (see digit layout on traffic diagram) to determine which of 3 reset lines to pulse. Each reset line to each FF of FF storage can set the FF to 1 or 0 or leave it undisturbed. The test storage and magnetic memory are strobed at the same time (but only one of them will have information), and test storage is read into at the same time magnetic memory is (see CPO 34 and 35).

The In-Out system shown in the lower left panel has not been fully explained. This system is now being designed by the in-out group. The details shown on this drawing are rather obsolete and will not be discussed further at this time.

The Memory Pause control (the first two CPO units) synchronizes the computer with the memory timing. If the strobe pulse in memory occurs earlier than TP 6, then the computer timing is not affected. But if the strobe occurs later than TP 6, the computer is stopped. It is started again after a delay which allows the memory to catch up with the computer. Similarly, if the M Adr Reg cannot be cleared on TP 11, TP 10 stops the computer until memory catches up.

The manual controls* shown under the clock pulse control are only suggestive of the controls required. The "start", "stop", and "semi-stop" are synchronized before being sent to the FF's. The "stop" stops everything on the next 2 mc pulse. The "semi-stop" stops everything (except in-out) at the end of a memory cycle; in-out breaks are

---

* See Important Note on Page 1.

allowed to continue until the in-out unit disconnects. The "alarm" can either do nothing, stop the computer completely, or stop it and later start it again (pausing long enough to photograph the indicator lights). The pushbuttons to the right of the "stop" button will start the computer and stop it immediately on the next pulse indicated. For instance, the rightmost button is called "instruction" and will cause the computer to stop after performing one instruction: TP 7 of program timing stops the computer just after inserting the next instruction in the instruction register and before the Ix Reg is added in and before any steps of this next instruction are performed. Any of these buttons can be connected to provide "semi-stop" instead of "stop". The only difference, essentially, is that if "semi-stop" is selected and a button is pushed which carries the computer through the in-out start pulse of wr or rd, the memory cycle will be completed, no more program memory cycles will be initiated, but all in-out breaks will be completed before the computer stops.

This completes the description of the attached drawings.

Section 4. Discussion of Details requiring further study and approval.

It is now assumed that the reader is sufficiently familiar with the drawings to follow a discussion of the alternatives that should be considered before final approval is obtained.

Except for the firm decisions tabulated in Section 5, all details are subject to change for the following reasons: (1) They have not been thoroughly checked to make certain they will work correctly, (2) they have not been studied to determine whether they are the best way of doing the job, (3) they have not been checked for electronic requirements, and (4) they have not been checked to make certain that all desired checking, operational, programming, and other requirements are met. IBM is now in the process of doing this checking and the necessary redesign, after which the drawings will be brought up to date. Firm decisions and approvals on the details of the system can then be attempted.

The following paragraphs discuss the items of Section 5, which indicate the principal areas of known difficulty or omission, and which point out some areas on which no discussion has taken place between MIT and IBM. (Recent results of IBM's work are discussed in Appendix B, and are not included in the discussion below.)

Items 1, 10, 14, 22 of Section 5: Although the general specifications of the arithmetic element have been approved there are numerous details which must be approved. The major ones are: the inclusion of a set of lines for transferring B Reg to A Reg (which might be necessary in some foreseeable new instructions), the location and exact function of a FF to the right of Acc 15 to catch the digit temporarily shifted off during regular addition, and the exact function of the end around carry control FF.

Items 2 and 3 require no comment.

Items 4 and 26: Separate lines are to be used for each transfer, but whether a read-out or read-in gate will be used in each instance has not been approved.

Item 5 requires no comment.

Items 6, 17, 23, 25: (For additional discussion of item 6, see discussions of items 8, 15, and 27.) Although it has been decided to use a class-and-variation selection scheme it has not been decided whether a matrix should be attached to the variation FF's nor how large this matrix should be. The operation code has also not been decided: the position of the bits in the instruction word and the code for the class and variation bits should both be decided upon.

Items 7, 9, 11, 13, 36, 40, 43: The general group of instructions have been approved, but specific approval of the function of specific instructions should also be obtained. The items 11 and 13 need no further comment. The exact form of the proposed overflow, and check, instructions have not been decided: The check instruction might be a "wave of ones" ("woo") or a "wave of zeros" ("woz") inserting ones or zeros in all FF's, or it might be a much simpler instruction. The partial-store instruction is proposed in M-2419, and allows easy modification of any selected digit positions of a word in memory.

Items 8, 16, 30, 39, 42 (and 6): It has been decided to use 0.1 μsec pulses from a time pulse distributor, and it is generally assumed (but not yet approved) that the distributor will provide 12 time pulses at a 2 mc rate. However, the clock frequency might not be exactly 2 mc, and the time pulses might be interrupted occasionally (see item 32). The circuits for the clock and distributor have not even been tentatively decided: the clock might be an oscillator or a delay line, and the distributor might be delay line, or counter, or stepping register. It has been proposed that the control matrix could be simplified by using a delay-line control (rather than time pulses) for the in-out breaks because the pulses to the distributor are more easily shut off than the whole class matrix.

Items 9, 10, 11: see 7, 1, 7 respectively.

Items 12, 19, 32: It is generally agreed that all memory cycles will be uniform in order to simplify the design of the IO break system. A delay-line control appears desirable, to allow stopping the computer without disturbing the memory process. Memory cycles longer than 6 μsec could be matched to the computer speed either by slowing down the computer clock or by interrupting the time pulses when necessary.

Items 13, 14: see 7 and 1 respectively.

Items 15, 18, 20, 21 (and 6): The general sequences of pulses to be used in most instructions have been generally agreed on, but nearly all the details have not been discussed. It seems desirable to provide a full microsecond for carry pulses to ripple down a set of gate tubes (note that the multiply scheme does not violate this principle. See Section 3.9. See item 38 for divide.) It also seems desirable to allow one full microsecond for the memory switch and the control matrix to stabilize whenever they are changed. The details most open to question are in the branch class.

Items 16, 17, 18, 19, 20, 21, 22, 23: see
8, 6, 15, 12, 15, 15, 1, 6 respectively.

Items 24, 34, 38: The use of the last two Adr Reg FF's for the divide pulse distributor is open to question: three FF's are probably required because it seems desirable to use a 5-pulse cycle (to allow carry time), and these FF's may be at the left of the step-counter part of the Adr Reg. The circuit is not yet designed for proper sensing of the end carry, the start of program timing, or the suppression of shift pulses if a shift of zero is programmed. The IO Wd Ctr is also not designed in these respects.

Items 25, 26: see 6 and 4 respectively.

Items 27, 28, 29 (and 6): Although it has been decided that the class and variation conditions will be combined to control "command pulse output units", the required interconnections are not yet decided. The interconnections shown will work in most cases, but a revision is necessary for electronic reasons, and because certain connections restrict the flexibility of adding new instructions. (A single CPO unit should not do more than one thing.) Unambiguous titles should be decided on for commands, other pulses, special FF's (Acc X or Acc 16?), etc. In particular, a command pulse sensing a gate, and the resulting command pulse to a FF, should have different titles. (They should also be differentiated by names such as "pre-command" and "post-command". Commands generated by sub-controls, such as divide control, should be called something such as "sub-commands" and have special titles.) A set of official CPO unit numbers should be assigned.

Item 30: see 8.

Item 31: The reset lines on test storage might be undesirable, or provided in a different way. The restriction on register zero, and the need for clearing the test storage switch, could be avoided by placing three gate tubes on the MM-TM FF.

Item 32: see 12.

Items 33, 41: The complete list of manual controls, and the details for mechanizing them, have not been worked out.

Item 34: see 24.

Item 35: Nearly all the design for in-out is subject to drastic revision, based on the designs developed by the in-out group. (See also item 34.)

Item 36: see 7.

Item 37: The parity digit should probably be "1" for even-parity numbers. If this is not true, then a failure of memory to read out at all would go undetected.

Items 38, 39, 40, 41, 42, 43: see
24, 8, 7, 33, 8, 7 respectively.

Section 5. Tabulation of items approved, suggested, or incomplete.

Refer to Section 4 for a discussion of the scope of this tabulation and for a discussion of each item. (Recent results of IBM's work are discussed in Appendix B, and are not tabulated below.)

Item	Definitely approved by letter from Mr. Forrester to Mr. Solomon, dated June 15, 1953. (following is an abstract):
1.	Provide arithmetic element with 32-bit registers, split into two 16-bit halves operated simultaneously with, and independent of, each other.
2.	Use a 3-input adder with combined add and shift right, for multiply.
3.	Omit check register, but provide space for it and associated read-out gates in other registers.
4.	Use separate lines for each transfer (i.e., no central bus).
5.	Provide two Ix Regs for programmers only. Allow space for 2 more.
6.	Allow operations to be selected by 3-bit class switch and 4-bit variation. Combine these to control "command pulse output" gates pulsed by 0.1 μsec pulses from time pulse distributor.
7.	Provide operations substantially as listed on pages 10 and 11 of IBM's report of April 27.

Item	Fairly firmly established in general conferences:
8.	Use 2 mc time pulse distributor, capable of starting and stopping at any times.
9.	Include an instruction dealing with overflow.
10.	Include gate tubes and circuit for "B Reg to A Reg" transfer.
11.	Omit "full" instructions and branches on zero.
12.	Use uniform memory cycles: 6 μsec minimum; provide for longer.
13.	Use new in-out instructions, substantially as in change #2 of operator's manual, IM 21.
(Note:)	See item 36.

Item	Fairly firmly established in small discussions between IBM and MIT:
14.	Place an extra FF to the right of each Acc 15.
15.	Provide sequences of pulses to be used in dv, and other operations substantially as shown on attached drawings.
16.	Use a 12-time pulse distributor.
17.	Use a small matrix to decode some of the variation FF's.

Item	Probably acceptable, but not necessarily discussed between IBM and MIT
18.	Provide sequences of pulses exactly as shown in many cases.
19.	Use Delay-line control for memory.
20.	Allow one μsec for carry ripple.
21.	Allow one μsec for each change in memory switch or control switch.
22.	Provide end around carry FF in each Acc substantially as shown.
23.	Locate bits of operation code as shown.
(Note:	See items 37-38)

Item	Probably not acceptable
24.	Use rightmost two Adr Reg FF's for divide pulse distributor.
25.	Use operation code (for classes and variations) as shown.
Item	Not yet decided (and probably not acceptable in most cases):
26.	Use read-out or read-in gates as shown.
27.	Use indicated interconnections between control matrix and commands.
28.	Use indicated titles of commands, pulses and special FF's.
29.	Use indicated CPO numbers.
30.	Make the clock run at exactly 2 mc.
31.	Make test storage as shown.
32.	Provide for longer memory cycle by "pause" as shown.
33.	Sync and interlock the manual controls as shown.
34.	Provide indicated control for end carry from Adr Reg and IO Wd. Ctr.
35.	Provide In-Out details as shown.
(Note:	See items 39-43)
Item	Not shown, but farily well established in general conferences:
36.	Provide some kind of check instruction.
Item	Not shown, but probably desirable:
37.	Make parity digit "1" for even-parity numbers.
38.	Make each divide step 5 pulses long (2.5 μsec).

Item	Not shown, and not yet decided (and probably not acceptable):
39.	Use delay-line clock and counting distributor.
40.	Use "W00" and "W02" check instructions.
41.	Provide only the manual controls shown.
42.	Control I-O breaks by delay-line rather than time pulses shown.
43.	Provide a partial-store instruction.

Signed R. P. Mayer
R. P. Mayer

RPM/cs

Drawing Nos.

SC-37625-4
SD-54846-4

Approved J. F. Jacobs
J. F. Jacobs

Approved N. H. Taylor
N. H. Taylor

APPENDIX A

(XD-J)Abbreviations and Titles (Official)

Registers

Note: Abbreviation given is for either half of a 32-bit register.
If a final "s" is included, the abbreviation refers to both 16-bit halves.
If an initial "L" is included, the abbreviation refers to the left half.
If an initial "R" is included, the abbreviation refers to the right half.

Acc	Accumulator
B Reg	B Register
A Reg	A Register
M Buf	Memory Buffer
Adr Reg	Address Register (16 bits only)
Op Reg	Operation Register (16 bits only)
Instr Reg	Instruction Register (Op Reg + Adr Reg)
Pr Ctr	Program Counter (16 bits only)
M Adr Reg	Memory Address Register (16 bits only)
Ix Reg	Index Register (16 bits only) (2 are available, both on R side)
IO Reg	Input-Output Register
IO Adr Ctr	" " Address Counter (16 bits only)
IO Wd Ctr	" " Word Counter (16 bits only)
IO Ctrl Ctr	" " Control Counter (IO Adr Ctr + IO Wd Ctr)
Ident Reg	Identification Register (16 bits only)
Dr Adr Ctr	Drum Address Counter (16 bits only)
Dr Adr Reg	" " Register (16 bits only)
IO Intlk	Input-Output Interlock
Ix Ic	Index Indicator (2 bits of Op Reg)
Ix Iv	Index Interval (7 bits of Op Reg)

APPENDIX A (Cont.)

Instructions

ca	Clear and Add
cs	Clear and Subtract
cm	Clear and Add Magnitude
ad	Add
su	Subtract
dm	Difference Magnitudes
mu	Multiply
dv	Divide
et	Extract
tad	Twin and Add
tsu	" " Subtract
tmu	" " Multiply
tdv	" " Divide
st	Store
lst	Left Store
rst	Right Store
rao	Add One
sta	Store Address
ec	Exchange
csw	Clear and Subtract Word Counter
sr	Shift Right
lsr	Left (Element) Shift Right
rsr	Right (Element) Shift Right
sl	Shift Left
cl	Cycle Left
slo	Shift Left and Round
asr	Shift Accumulators Right
asl	" " Left
acl	Cycle " "
bm	Branch on Minus
blm	" " Left Minus
brm	" " right "
bi	Branch and Index
ria	Reset Index Register
rir	" Ix Reg from R Acc
aia	Add Index Register
se	Select
sea	Select by Address
sei	Select by Identification
rd	Read
wr	Write
lac	Load Address Counter
so	Skip or Operate

APPENDIX B

Some Recent Proposals and Decisions

No official approval of anything has been issued since the body of this report was prepared. Some Project Grind meetings have resulted in a few fairly firm proposals, an H-note on memory pause has been issued, and a few preliminary block schematics have been drawn by IBM. Those developments will now be discussed in that sequence. The developments have not yet been incorporated into the attached drawings in most cases.

The Project Grind meetings have resulted in the following fairly firm proposals:

Item	
44	Project Grind 5th day: Modification of item 41: Provide the following manual controls: Start from present condition, start from register 8192, load standard operating program from drum, load from card reader, clear all FF's, clear memory, stop on next instruction (semi-stop), perform one instruction, one memory cycle, sequence DC off, remove DC suddenly, turn power on, off, marginal check (controls not specified), lock all controls except emergency, make alarm do (unspecified actions), operate program cyclicly, complement all FF's at a given frequency. Any control which starts the computer will clear out all alarm indications. Note that details of these controls are not yet worked out.
45	Project Grind 3rd day: New item: Provide a parity digit on all drums which are written-on from the computer, carry this digit along in all external transfers, and check the parity of all drums which read into the computer (if they have parity digits). This check should be performed in M Buf. Note that details of this check are not yet worked out, including the problem of what to do when an alarm occurs.
46	Project Grind 2nd day: New item: Provide two complete magnetic memories in such a way that the second one can be built later and just plugged in. Note that the details of this have not yet been worked out.

Note H-7 proposes a pause system for memory, which appears to be fairly firmly established:

Item	
47	Modification of items 19 and 32: Allow memory to develop its own timing beginning on TP 3 (implying delay-line control). Make computer stop on TP 6 and 10 if magnetic memory was selected. H-7 says the equivalent restart pulses will be provided by memory, but it has not been decided whether they should be generated by central control instead. Details have not been worked out.

APPENDIX B (Cont.)

About 35 preliminary sketches for block schematics have been drawn by IBM. These have been summarized on drawing SD-47010 (two pages). These are not attached because the additional bulk seemed unjustified. The block schematics (and summary), being preliminary sketches, are understandably incomplete and inconsistent in some places. Comments on these sketches have been forwarded to IBM. No attempt will be made here to review those parts of the drawing which seem questionable. A subsequent supplement to this note, and a revision of the attached drawings, will present the design after IBM has considered the problem further.

The following items are proposals, shown on (or implied by) the IBM sketches, which appear fairly firmly established at this time and which are different from the attached drawings:

Item	
48	Modification of item 13: Make operation "so" two instructions: "sense" and "operate."
49	Modification of item 9: Make overflow-sense an address of "sense" instruction.
50	New item: Obtain Ix Iv from Op Reg instead of M Buf. (Note that Op Reg should now be 16 digits long.)
51	Modification of item 23: Place sense and operate matrix on Op Reg, using Ix Iv bits.
52	Modification of item 17: Use a 16-position matrix on the variation FF's, and do not use the FF outputs directly.
53	Modification of item 27: Use OR and AND gates to combine outputs of class and variation matrixes instead of using variation FF's directly. (Note: To a large extent the only cost of the decode-recode circuit of items 52 and 53, as opposed to a direct use of the FF outputs, is less than 200 diodes. The advantage is that some future instructions will require some of these circuits anyway, and it is desirable to use a straightforward, consistent system of interconnections at least in the early models of the computer.)
54	Modification of item 29: Make each CPO unit do only one thing.
55	Modification of item 28: Call the extra Acc FF "Acc 16" instead of "Acc X".
56	New item: Place parity bit on left end of M Buf.

APPENDIX B (Cont'd)

Item	
57	New item: Use a separate overflow accumulator FF, instead of time-sharing the sign control FF, in addition to the overflow FF.
58	New item: Use FF's, not levels from matrix, for controlling "Acc 15 to B Reg S" on multiply, and quotient on divide. (See item 63).
59	New item: Omit subtracting count circuit in Adr Reg. Always count up; complement Adr Reg before starting shift count.
60	Modification of items 24, 38: count 5 pulses for each step of divide, using Adr Reg digits 8-10 for pulse distributor.
61	Modification of item 34: Use diode AND and OR circuits on Adr Reg to determine if shifting should start, if time pulses should stop, and when to start time pulses and to stop shifting.
62	Modification of item 22: Sense A Reg sign, not End Around carry FF, to determine carry inputs during divide. Sense adder matrix, not separate gate tubes, to complement A Reg during div. (Note: This simplifies divide control by making the end around carry FF unnecessary during divide except for the shift left on PT TP 5.)
63	New item (see item 58): Obtain quotient from carry out of Acc S, and run it directly to B Reg 15. (Note: Quotient-connection FF is turned off before correcting remainder.) Omit clearing of BR 15 at start of divide, because this digit will be lost anyway.
64	New item: Read complement of IO Wd Ctr to R Acc on csw, and omit complementing R Acc afterward.
65	Modification of item 26: Use read out gates for all transfers.

APPENDIX C

Guide to Item Numbers

In the following list, each item number appears only once, although it may belong in several categories.

Underline means heading of discussion in Section 4, or that item is in Appendix B.

Group		
A	Instructions	(7,9,11,13,36,40,43)48,49
B	Matrixes	(6,17,23,25)50,51,52
C	CPO Units and Titles	(27,28,29)53,54,55
D	Clock and TPD	(8,16,30,39,42)
E	Command Sequences	(15,18,20,21)
F	Manual Controls	(33,41)44
G	Memory (and Control)	(12,19,32)37, 46,47,56
H	Test Storage	31
J	Ix Regs	5
K	Arithmetic Element	(1,10,14,22)2,57,58,59,62
L	Divide	(24,34,38)60,61,63
M	In-Out	35,45,64
N	Check Register	3
P	Transfer Paths	(4,26)65

Item #	0	10	20	30	40	50	60
0		K	E	D	A	B	L
1	K	A	E	H	F	B	L
2	K	G	K	G	D	B	K
3	N	A	B	F	A	C	L
4	P	K	L	L	F	C	M
5	J	E	B	M	M	C	P
6	B	D	P	A	G	G	
7	A	B	C	G	G	K	
8	D	E	C	L	A	K	
9	A	G	C	D	A	K	

APPENDIX D

Detailed Table of Contents

APPENDIX D
(Cont.)

Detailed Table of Contents