Chapter 33

The IBM 1800

Introduction

This third-generation computer is constructed with hybrid-circuit technology (semiconductors bonded to ceramic substrates) known as SLT (Solid Logic Technology). It has a core primary memory.

The 1800 is designed for process control and real-time applications. It is nearly identical to the IBM 1130, which is designed for small-scale, general-purpose, and scientific calculation applications. The two C's perform about the same for computation bound problems. The 1130 and 1800 are not program compatible with the "universal" IBM System/360 series, though introduced at about the same time. However, the 1800 uses terminals and secondary memories similar or identical to the System/360. These are organized about the standard IBM System/360 8-bit byte. Thus their common information media provide a link between the two. Hence an 1800 is sometimes connected to the System/360 as a preprocessor. The relative performance of the IBM 1130, 1800, and the IBM System/360 can be seen on page 586. The 1800 has a better cost/performance ratio than a System/360, Model 40 and has the performance of a Model 30. From now on we will refer only to the IBM 1800, although much applies to the IBM 1130.

The 1800's interface facilities include a large number of T's which can connect to different physical processes; a multiple priority interrupt facility with fast response; multiple Pio's which can transfer information at high data rates;[1] and a complete instruction set for real-time, nonarithmetic processing.

We include the 1800 because it is a typical, 16-bit, real-time, process control computer. The ISP is the most straightforward of the IBM computers in the book (and perhaps the nicest). The several different Pio's and their implementations are unusual and should be carefully studied. Important aspects of the 1800 include the PMS structure as it links to real-time processes, e.g., analog processes; the straightforward Pc ISP (Appendix 1 of this chapter); the specialized Pio's for real-time T's; the Pc implementation; and the Pio implementation. The chapter is written to expose and explain these aspects.[2]

By comparing the 1800 with Whirlwind, an evolutionary progression can be seen. Their ISP's are similar but, because of better technology, the 1800 shows an increase in capability. The 1800 Pc has a medium-sized state (ISP has six registers) including three index registers. The implementation is not elegant; a single register array and adder would provide the basis for a straightforward Pc implementation. The 1800 has features which facilitate higher information processing rates compared with Whirlwind. The major change between Whirlwind and the 1800 machines was brought about by the decreasing cost of registers and primary memory. In the 1800, all K's have independent memory (usually $1 \sim 2$ words or characters) so that concurrent operation of almost all the T and Ms via their K's is possible. In contrast, Whirlwind has only a single, shared register in Pc, and only one device can operate at a time.

Lower hardware costs allow multiple Pio's in the 1800. The Pio's represent an unusual approach to information processing in this period. The Pio's which process standard disk, magnetic tape, and card reader are conventional, but the Pio's for analog and process signals are novel and interesting. The latter Pio's are the most unusual part of the 1800, and they allow independent programs in each Pio to do some very trivial processing tasks such as alarm-condition monitoring independent of Pc. However, the Pio's are limited; for example, it is difficult to transmit or receive a data block between Ms and Mp (using a Pio) without surrounding the data block with Pio control words (thereby transmitting the control words).

The interrupt system is typical of second- and third-generation computers and is comparable to the SDS 900 series (Chap. 42). In later computers interrupt conditions are used to determine a fixed address to which the processor interrupts. There are generally many conditions (100 to 1,000), but only a few discrete levels (8 to 20). The 1800 depends on program polling within a discrete interrupt level; each level has a unique, fixed address.

A principal ISP design problem is the addressing of the 65,536-word Mp. Thus, a 16-bit number has to be generated within Pc for an address. In this regard the 1800 behaves like the 12-bit machines which have to address a 2^{12} (4,096) word memory, and the modes or methods the 1800 uses for addressing are reasonable. It should be noted that it is relatively difficult to write programs which do not modify themselves. For example, the instruction, Store Status, is changed by its execution.

[1] Although we refer to the data channels as Pio's, they have a very limited ISP for a Pio; in fact, they might better be called K's.

[2] Some of the material in the chapter has been abstracted from the IBM 1800 Functional Characteristics Manual.

A peculiar feature of the 1800 is its storage protection (see page 408). This feature should provide program relocation capability in addition to protection, but it does not.

PMS structure

A simplified picture of the IBM 1800 structure is given in Fig. I, without Pio('Data Channel)'s and K('Device Adapter)'s. Each T and Ms have a K which connects Pc's In and Out Bus, the S('Pc to K). Some K's attach to Pio's and some directly to Pc. Information can be transferred between Mp and K via Pio at rates up to 0.5 megaword/s or 8 megabits/s. The IBM Configurator (Fig. 2) gives the restrictions on the possible structures, together with minute L details. It is presented as an alternative to the PMS structure (Fig. I). The Configurator is intended to show the "permissible structures" but does not show the logical or physical structure. The PMS diagram (Fig. 3) alternatively shows the physical-logical hardware structure and performance parameters. It should be noted that a PMS diagram with the information of the computer component Configurator (Fig. 2) would require slightly more details (and space).

The central processor[1]—primary memory

The IBM 1800 is a fixed-word-length, binary computer with 4, 8, 16, or 32-kword memories of 16 + 1 + 1 bits, and a memory cycle time of 2 or 4 microseconds. Of the 18 bits 1 bit is used as a parity check (P bit) and 1 bit is used for storage protection (S bit). The Pc instruction set operates on 16-bit and 32-bit words. Indirect addressing and three index registers are used in address modification. The Pc has a 24-level interrupt system, three interval timers, and a console.

The Pc interrupt is a forced branch (jump) in the normal program sequence based upon external or internal Pc conditions. The devices and conditions that cause interrupts are hardwired in fixed priority levels. An interrupt request is not honored while the level of the request itself or any higher level is being serviced, or if the level requested is masked. Examples of interrupt conditions are:

I An external process condition that requires attention is detected.

[1]IBM name: the Processor-Controller or PC.

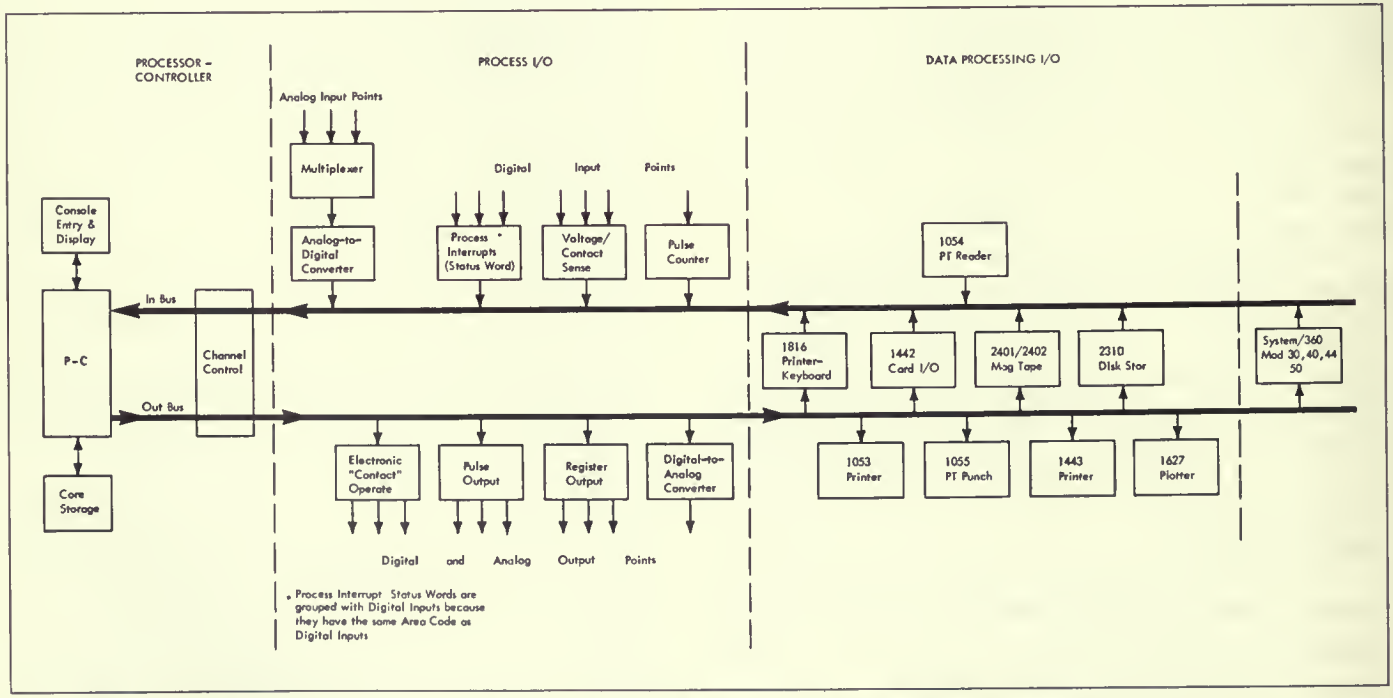


Fig. 1. IBM 1800 data acquisition and control system. (*Courtesy of International Business Machines Corporation.*)

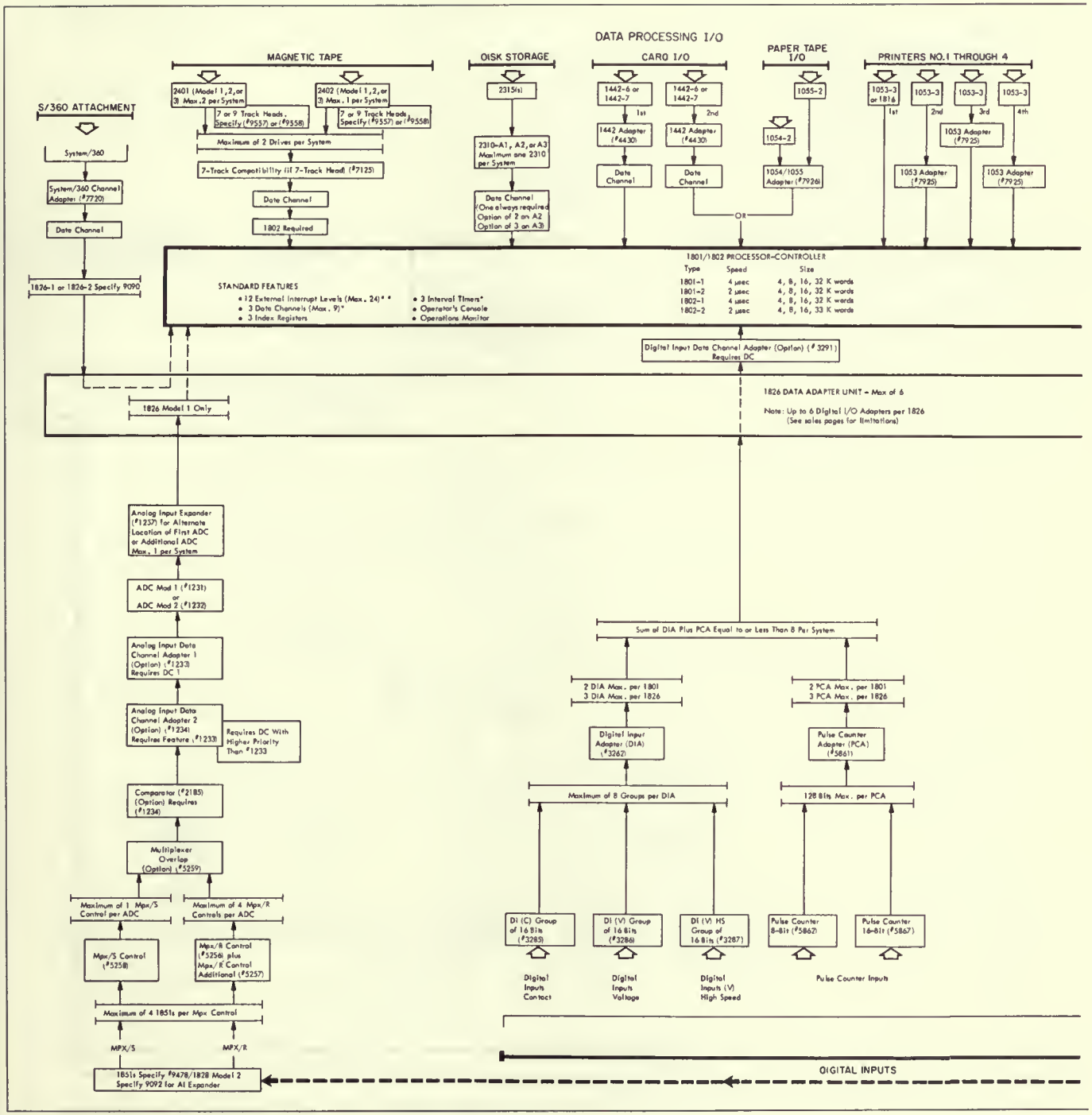
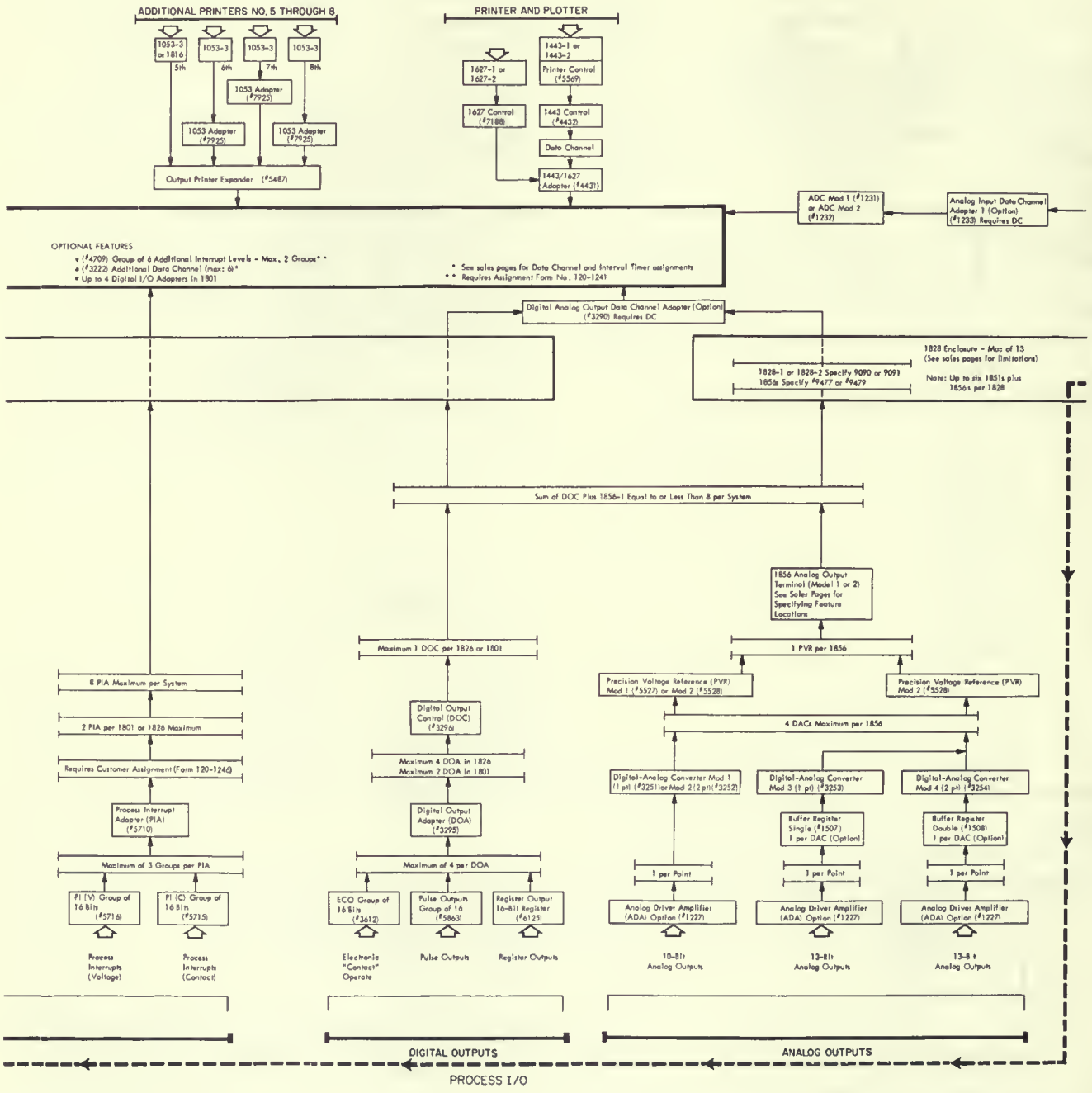
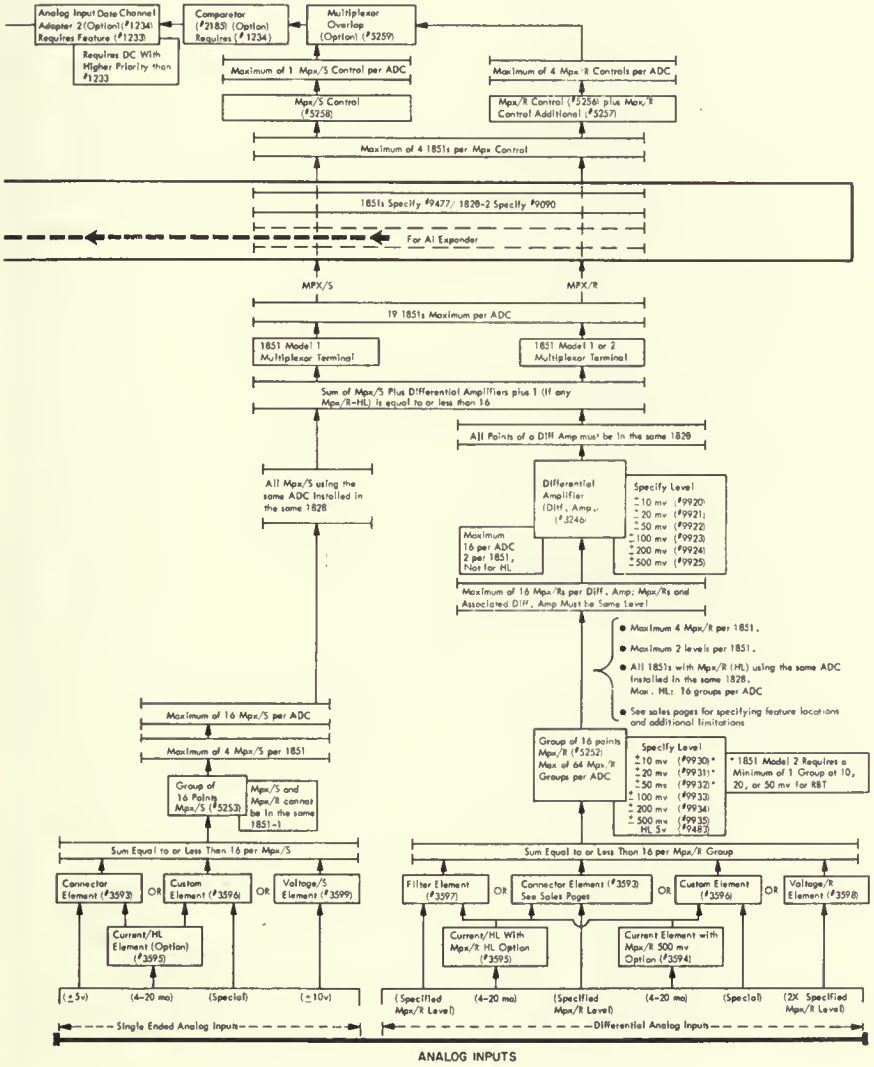
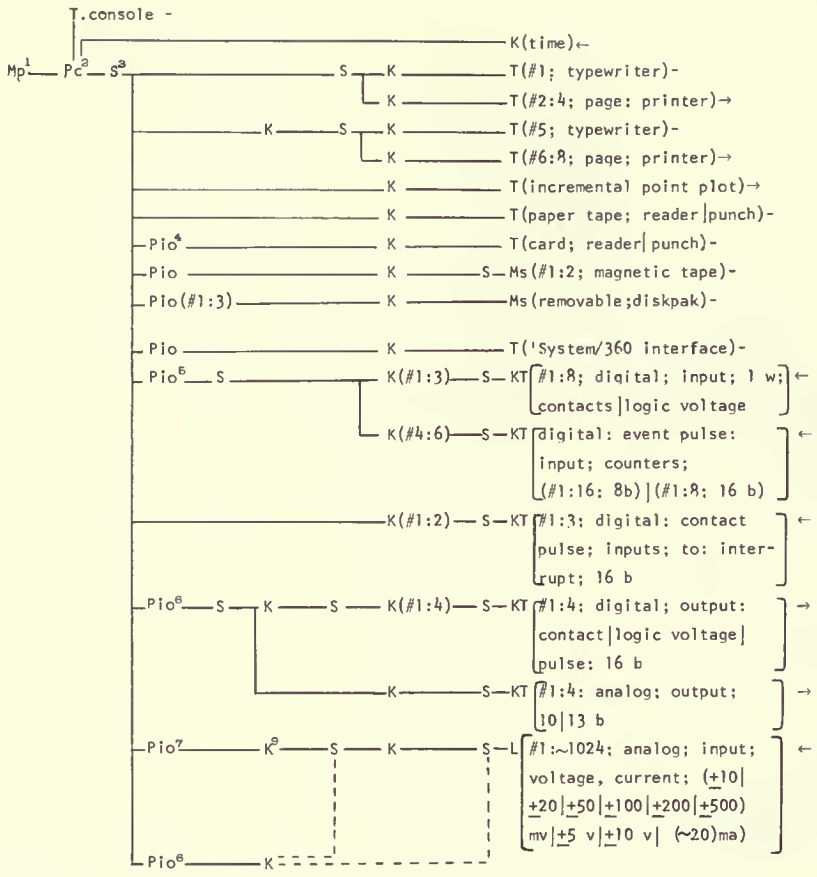


Fig. 2. IBM 1800 data-acquisition and control-system configurator.
(Courtesy of International Business Machines Corporation.)


ADDITIONAL PRINTERS NO. 5 THROUGH 8
1053-3 or 1816
5th
1053-3
6th
1053-3
7th
1053-3
8th
1053 Adapter (#7925)
1053 Adapter (#7925)
1053 Adapter (#7925)
Output Printer Expander (#5487)
PRINTER AND PLOTTER
1627-1 or 1627-2
1443-1 or 1443-2 Printer Control (#5569)
1627 Control (#7188)
1443 Control (#4432)
Data Channel
1443/1627 Adapter (#4431)
ADC Mod 1 (#1231) or ADC Mod 2 (#1232)
Analog Input Data Channel Adapter 1 (Option) (#1233) Requires DC
OPTIONAL FEATURES
• (#4709) Group of 6 Additional Interrupt Levels - Max., 2 Groups* *
• (#3222) Additional Data Channel (max: 6)*
• Up to 4 Digital I/O Adapters in 1801
* See sales pages for Data Channel and Interval Timer assignments
* * Requires Assignment Form No., 120-1241
Digital Analog Output Data Channel Adapter (Option) (#3290) Requires DC
1828 Enclosure - Max of 13 (See sales pages for limitations)
1828-1 or 1828-2 Specify 9090 or 9091
1856s Specify #9477 or #9479
Note: Up to six 1851s plus 1856s per 1828
Sum of DOC Plus 1856-1 Equal to or Less Than 8 per System
1856 Analog Output Terminal (Model 1 or 2) See Sales Pages for Specifying Feature Locations
Maximum 1 DOC per 1826 or 1801
1 PVR per 1856
8 PIA Maximum per System
2 PIA per 1801 or 1826 Maximum
Requires Customer Assignment (Form 120-1246)
Process Interrupt Adapter (PIA) (#5710)
Maximum of 3 Groups per PIA
PI (V) Group of 16 Bits (#5716)
PI (C) Group of 16 Bits (#5715)
Process Interrupts (Voltage)
Process Interrupts (Contact)
Digital Output Control (DOC) (#3296)
Maximum 4 DOA in 1826 Maximum 2 DOA in 1801
Digital Output Adapter (DOA) (#3295)
Maximum of 4 per DOA
ECO Group of 16 Bits (#3612)
Pulse Outputs Group of 16 (#5863)
Register Output 16-Bit Register (#6125)
Electronic "Contact" Operate
Pulse Outputs
Register Outputs
Precision Voltage Reference (PVR) Mod 1 (#5527) or Mod 2 (#5528)
Precision Voltage Reference (PVR) Mod 2 (#5528)
4 DACs Maximum per 1856
Digital-Analog Converter Mod 1 (1 pt) (#3251) or Mod 2 (2 pts) (#3252)
Digital-Analog Converter Mod 3 (1 pt) (#3253)
Digital-Analog Converter Mod 4 (2 pts) (#3254)
Buffer Register Single (#1507) 1 per DAC (Option)
Buffer Register Double (#1508) 1 per DAC (Option)
1 per Point
3 per Point
1 per Point
Analog Driver Amplifier (ADA) Option (#1227)
Analog Driver Amplifier (ADA) Option (#1227)
Analog Driver Amplifier (ADA) Option (#1227)
10-Bit Analog Outputs
13-Bit Analog Outputs
13-Bit Analog Outputs
DIGITAL OUTPUTS
ANALOG OUTPUTS
PROCESS I/O


Analog Input Data Channel Adapter 2 (Option) (#1234) Requires Feature (#1233)
Requires DC With Higher Priority than #1233
Comparator (#2185) (Option) Requires (#1234)
Multiplexor Overlap (Option) (#5259)
Maximum of 1 Mpx/S Control per ADC
Maximum of 4 Mpx/R Controls per ADC
Mpx/S Control (#5258)
Mpx/R Control (#5256) plus Mpx/R Control Additional (#5257)
Maximum of 4 1851s per Mpx Control
1851s Specify #9477/ 1828-2 Specify #9090
For AI Expander
MPX/S
MPX/R
19 1851s Maximum per ADC
1851 Model 1 Multiplexor Terminal
1851 Model 1 or 2 Multiplexor Terminal
Sum of Mpx/S Plus Differential Amplifiers plus 1 (If any Mpx/R-HL) is equal to or less than 16
All Points of a Diff Amp must be in the same 1828
All Mpx/S using the same ADC Installed in the same 1828
Differential Amplifier (Diff. Amp.) (#3246)
Specify Level
±10 mv (#9920)
±20 mv (#9921)
±50 mv (#9922)
±100 mv (#9923)
±200 mv (#9924)
±500 mv (#9925)
Maximum 16 per ADC 2 per 1851, Not for HL
Maximum of 16 Mpx/Rs per Diff. Amp; Mpx/Rs and Associated Diff. Amp Must be Same Level
Maximum 4 Mpx/R per 1851.
Maximum 2 levels per 1851.
All 1851s with Mpx/R (HL) using the same ADC Installed in the same 1828. Max. HL: 16 groups per ADC
See sales pages for specifying feature locations and additional limitations
Maximum of 16 Mpx/S per ADC
Maximum of 4 Mpx/S per 1851
Group of 16 Points Mpx/S (#5253)
Mpx/S and Mpx/R cannot be in the same 1851-1
Group of 16 points Mpx/R (#5252) Max of 64 Mpx/R Groups per ADC
Specify Level
±10 mv (#9930)*
±20 mv (#9931)*
±50 mv (#9932)*
±100 mv (#9933)
±200 mv (#9934)
±500 mv (#9935)
HL 5v (#9483)
* 1851 Model 2 Requires a Minimum of 1 Group at 10, 20, or 50 mv for RBT
Sum Equal to or Less Than 16 per Mpx/S
Sum Equal to or Less Than 16 per Mpx/R Group
Connector Element (#3593)
OR
Custom Element (#3596)
OR
Voltage/S Element (#3599)
Current/HL Element (Option) (#3595)
Filter Element (#3597)
OR
Connector Element (#3593) See Sales Pages
OR
Custom Element (#3596)
OR
Voltage/R Element (#3598)
Current/HL With Mpx/R HL Option (#3595)
Current Element with Mpx/R 500 mv Option (#3594)
(±5v)
(4-20 ma)
(Special)
(±10v)
(Specified Mpx/R Level)
(4-20 ma)
(Specified Mpx/R Level)
(4-20 ma)
(Special)
(2X Specified Mpx/R Level)
Single Ended Analog Inputs
Differential Analog Inputs
ANALOG INPUTS



[1]Mp(core; 2|4 μs/w; 4096 ~ 32768 w; (16, parity, protect) b/w)
[2]Pc('1801|1802; 1 ~ 2 w/instruction; technology: hybrid: Mps(~ 6 w): 1 address/instruction; ~1965)
[3]S('In Bus, Out Bus)
[4]Maximum of 9 Pio per C
[5]Pio('Digital Input Data Channel)
[6]Pio('Digital, Analog Output Data Channel)
[7]Pio('Analog Input Data Channel)
[8]Optional Pio to control analog channel;(structure is greatly simplified)
[9]K('ADC; analog; input; 9, 12, 15 b/w; i.rate: 9 ~ 24 kw/s)

Fig. 3. IBM 1800 PMS diagram (simplified).

2 An interval timer has counted a previously set time interval.

3 A magnetic-tape drive has completed a data transfer previously requested and is ready for another request.

4 An operator has initiated an interrupt from the Pc console.

5 A device such as a typewriter has just printed a character and is ready to receive the next one.

Primary-memory communication and data transmission with terminals and secondary memory

Two methods are used to transmit data between Mp and Ms, or Mp and T. First, low-speed devices are controlled directly by the program. Each character or word of data is transmitted to or from the Pc and onto T by means of an Execute I/O(XIO) instruction. The Pc program and device synchronization are accomplished by using the interrupt mechanism. Devices operating under direct program control include typewriter, printer, plotter, paper tape reader and punch, analog-to-digital converters, contact sense, voltage-level sense, pulse counters, etc.

The second method of transferring data is via the Pio('Data Channel)'s. The Pio program is started by the XIO instruction of the Pc. The transfer of data words then proceeds under control of the specified Pio, completely asynchronous to and in parallel with Pc program operation. The Pio gains Mp access independent of Pc (Pc operation is suspended for one Mp cycle). During the Mp cycle, the data are taken from or placed into core storage by Pio (via internal Pc control and registers). As soon as the Pio has been satisfied, which normally takes one cycle, the Pc proceeds. The logical state of the Pc, or the Instruction-set Processor, is not changed by Pio's access to Mp. This method of access is referred to as "cycle stealing." Devices (Ms and T) operating under Pio control include magnetic tapes, disks, line printer, card reader-punch, and the link to the IBM System/360.

Some devices can operate under both Pc and Pio control, depending on their characteristics and the configuration, e.g., analog input, analog output, digital input, and digital output.

Process I/O, controls and transducers

Analog inputs. Analog-input equipment includes analog-to-digital converters, multiplexors, amplifiers, and signal conditioning equipment to handle various analog-input signals. The data input rates are up to 20,000 16-bit samples per second, with program selectable resolution and external synchronization. There can be 1,024 (via relay) and 256 (via high-speed solid state) multiplexed analog-input channels connected to a single K (analog-to-digital converter). The Configurator (Fig. 2) shows the allowable inputs.

Digital inputs. The Digital Input provides up to 384 process interrupts; up to 1,024 bits of contact sense, digital input, or parallel register input; and 128 bits of event input counters as 1-, 8-, and 16-bit counting registers.

Analog outputs. Up to 128 analog outputs can be provided.

Digital outputs. Digital Outputs provide up to 2,048 bits of pulse output, contacts, and registers.

IO processors (data channels)

Pio('Data Channels) give a T or Ms the ability to communicate directly with Mp. For example, if an input unit requires a primary memory cycle to store data that it has collected, the Pio communicates directly with Mp and stores the data.

The Pio's run even if Pc is waiting. The Pio's have two registers: a Word Count which is used to count the number of words being transferred in a block between a device and Mp memory; and a Channel Address which points to the next word transferred in a block. The Channel Address is also used to select the next instruction in the program for the next block transfer task.

Two basic types of Pio's are used, nonchaining and chaining.[1] The Pio's provide the ability to transfer either a single block (nonchaining) or multiple blocks (chaining) directly to Mp independent of Pc.

The central processor

Registers in the physical processor

Figure 4 shows the relationship of the registers in Pc, together with those in the Instruction-set Processor. Those registers accessible by the program are shown with an °. All the registers are accessible from the console. A description of the functions of each register is given below.

Storage address register (SAR). All Pc references to Mp are selected or accessed by this 16-bit register. Pio references to Mp use the Channel Address Register (CAR) of the active Pio.

Instruction register (I)°. This 16-bit counter register holds the address of the next instruction.

Storage buffer register (B). This 16-bit register is used for buffering all word transfers with Mp.

[1] A descriptive name undoubtedly concocted by one of IBM's marketing departments.

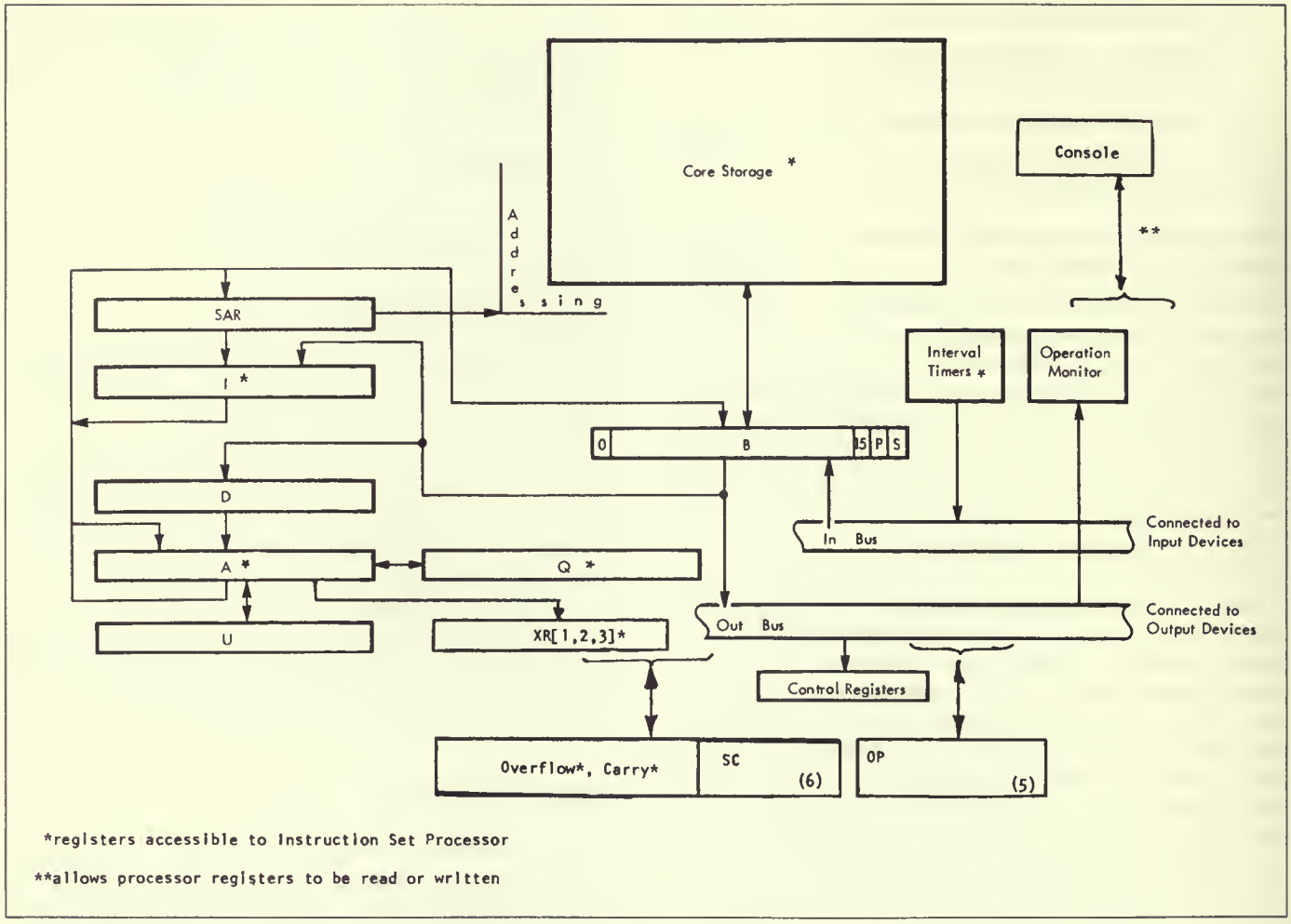


Fig. 4. IBM 1800 Pc data flow. (*Courtesy of International Business Machines Corporation.*)

Arithmetic factor register (*D*). This 16-bit register is used to hold one operand for arithmetic and logical operations. The Accumulator provides the other factor.

Accumulator (*A*)*. This 16-bit register contains the results of any arithmetic operation. It can be loaded from or stored into core storage, shifted right or left, and otherwise manipulated by specific arithmetic and logical instructions.

Accumulator extension (*Q*)*. This register is a 16-bit low-order extension of the Accumulator. It is used during multiply, divide, shifting, and double-precision arithmetic.

Shift control counter (*SC*). This 6-bit counter is used primarily to control shift operations.

Accumulator temporary (*U*). The U register is used to store A temporarily during an instruction or an operation which requires the A's facilities.

OP register (*OP*). This 5-bit register is used to hold the operation code portion of an instruction.

*Index registers**. The three 16-bit registers are used in effective-address calculations.

Overflow and carry indicators°. The two indicator bits associated with the Accumulator are Overflow and Carry. The Overflow indicator can be turned on by Add, Subtract, or Divide instruction and indicates a result larger than can be represented in the Accumulator. The Overflow indicator can also be turned on by a Load-status instruction. Once Overflow is on, it will not be changed except by testing the indicator, or by a Load-status or Store-status instruction. The Carry indicator provides the information that a carry (or borrow) from the high-order position of the Accumulator has occurred.

The Carry indicator is used with the Add, Subtract, Shift-left, Load-status, Store-status, and Compare instructions.

In-bus. This 18-bit bus is a link(L) used to carry information from a K to Pc. Generally only 16 of the 18 bits are used, although transfers to magnetic tape can be made three 6-bit characters.

Out-bus. This 18-bit bus is used to carry information from Pc to a K.

Instruction-set processor

The operation of the Pc from a program viewpoint follows. The ISP registers were declared (°) in the previous section and in Fig. 4. The ISP registers are the 16-bit I, A, Q, XR [1, 2, 3], and the 1-bit Overflow and Carry.

An ISP description of the 1800 appears in Appendix I of this chapter. It is incomplete in the following respects: The memory protect bit checking is not described; the illegal (undefined) instruction action is not described; double word data must be aligned on even and odd address word boundaries or else a fault occurs; and the IO instruction and interrupt operation are not given.

Instruction formats. Two basic instruction-word formats are used, one word (Fig. 5) and two word (Fig. 6). The bits within the instruction words are used in the following manner:

OP — Operation Code. These 5 bits define the instruction.

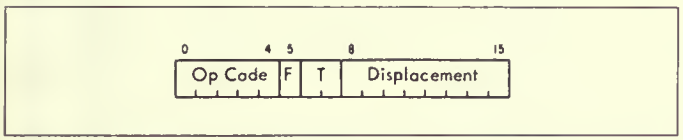


Fig. 5. IBM 1800 one-word-instruction format. (*Courtesy of International Business Machines Corporation.*)

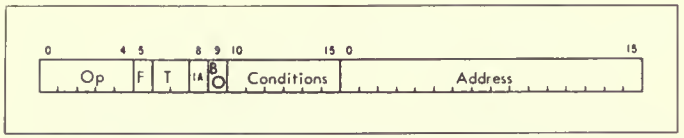


Fig. 6. IBM 1800 two-word-instruction format. (*Courtesy of International Business Machines Corporation.*)

F — Format bit. A 0 indicates a single-word instruction, and I a two-word instruction.

T — Tag. These 2 bits specify which of the three index registers is used in address modification or the shift count.

DISP — Displacement. These 8 bits are usually added to the instruction register or the index register specified by T for one-word instructions. The modified address is defined as the Effective Address (EA). If T is 00, the displacement is added to the instruction register (then EA = I + DISP). The displacement is in two's complement form if negative, with the sign in bit 8. The bit in position 8 is automatically extended to the higher-ordered bits (0 to 7) when the displacement is used in EA generation.

IA — Indirect addressing. This bit is used only in the two-word-instruction format. If 0, addressing will be direct. If a I, addressing will be indirect. Only one level of indirect addressing is permitted. (The Load Index and Modify Index and Skip instructions have exceptions, as shown in the ISP description.)

BO — Branch Out. This bit is used to specify that the Branch or Skip on Condition (BSC) instruction is to be interpreted as a Branch Out (BOSC) when used in an interrupt routine.

COND — Conditions. These 6 bits select the indicators that are to be interrogated on a BSC or BSI instruction. The bit assignments for conditions are:

Cond⟨10⟩ A = 0
Cond⟨11⟩ A < 0
Cond⟨12⟩ A > 0
Cond⟨13⟩ (A⟨15⟩ = 0) *that is, A is even*
Cond⟨14⟩ (Carry = 0)
Cond⟨15⟩ (Overflow = 0)

ADDRESS — These 16 bits usually specify a core storage address

Table 1 Determining effective addresses

	$F = 0$ *(direct addressing)*†	$(F = 1) \wedge (IA = 0)$ *(direct addressing)*	$(F = 1) \wedge (IA = 1)$ *(indirect addressing)*
T = 00	EA ← I + Disp‡	EA ← Address	EA ← C(Address)§
T = 01	EA ← XR[1] + Disp	EA ← Address + XR[1]	EA ← C(Address + XR[1])
T = 10	EA ← XR[2] + Disp	EA ← Address + XR[2]	EA ← C(Address + XR[2])
T = 11	EA ← XR[3] + Disp	EA ← Address + XR[3]	EA ← C(Address + XR[3])

† Contents of instruction register (I) or index register (XR[1], XR[2], XR[3]).

‡ May be true positive quantity or negative two's complement quantity.

§ C specifies "contents" at location specified by Address or Address + XR[1], XR[2], or XR[3].

in a two-word instruction. The address can be modified by the contents of an index register or used as an indirect address if the IA bit is on.

Effective-address generation. The Effective Address (EA) is developed as shown in Table 1. The instruction set is divided into five classes as shown in Table 2.

Storage protection. The storage-protection facility protects the contents of specified individual locations of Mp from change due to the erroneous storing of information during the execution of a program. The status of each location is identified as "read only" or "read/write" by the condition of the Storage Protect Bit, S.

The Store-status instruction is used to write and clear Storage Protect Bits. The execution of this instruction is under control of the Write Storage Protect Bits switch on the console. Any attempt by the program to write into a read-only protected location results in a storage-protect violation which causes the Internal Interrupt (the highest priority interrupt).

Instruction interpretation process

The simplified Pc data-flow block diagram (Fig. 4) shows instructions and data entering and leaving memory via the B register. Additional bits in Pc hold the P and S bits for Mp. Input devices send data and instructions to the B register via the 18-bit In-bus. Output devices receive data from the B register via the 18-bit Out-bus. Eighteen bits can be transferred between Pc and K(magnetic tape). As each stored-program instruction is selected, its various parts (op code, format bit, etc.) are directed to the control registers via the B register and the Out-bus. The control registers decode and interpret each instruction before the instruction is executed.

Except for Pio operations, all instructions and data in memory are addressed by the Storage Address Register (SAR). SAR obtains the memory address from the I register or the A register. The

Table 2 Instruction set

Class	*Instruction*	*Indirect addressing*	*Mnemonic*
Load and store	Load accumulator	Yes	LD
	Double load	Yes	LDD
	Store accumulator	Yes	STO
	Double store	Yes	STD
	Load index	‡	LDX
	Store index	Yes	STX
	Load status	No	LDS
	Store status	Yes	STS
Arithmetic	Add	Yes	A
	Double add	Yes	AD
	Subtract	Yes	S
	Double subtract	Yes	SD
	Multiply	Yes	M
	Divide	Yes	D
	And	Yes	AND
	Or	Yes	OR
	Exclusive Or	Yes	EOR
Shift	Shift Left instructions:		
	Shift left logical (A)†	No	SLA
	Shift left logical (AQ)†	No	SLT
	Shift left and count (AQ)†	No	SLC
	Shift left and count (A)†	No	SLCA
	Shift Right instructions:		
	Shift right logical (A)†	No	SRA
	Shift right arithmetically (AQ)†	No	SRT
	Rotate right (AQ)*	No	RTE
Branch	Branch and store I	Yes	BSI
	Branch or skip on condition	Yes	BSC (BOSC)
	Modify index and skip	‡	MDX
	Wait	No	WAIT
	Compare	Yes	CMP
	Double compare	Yes	DCM
I/O	Execute I/O	Yes	XIO

† Letters in parentheses indicate registers involved in shift operations.

‡ See the section for the individual instruction (MDX and LDX).

contents of the I register are developed by one of the following means, depending on the Pc operation:

1 The I register is incremented for each instruction.

2 The effective address of each instruction is developed in the accumulator (A register) and then transferred to SAR. The contents of the accumulator are saved in an auxiliary (U) register during effective-address computation. If the instruction was a branch, the contents of SAR is transferred to the I register.

The following examples illustrate the data flow or instruction interpretation process for the Load Accumulator (LD) instruction.

One-word load instruction
Instruction Cycle

1 A register transfers to U register.

2 I register transfers to SAR (I register is then incremented).

3 SAR addresses the memory location containing the instruction.

4 Memory location transfers to the B register and Out-bus.

5 Control registers store various parts of the instruction (op code, format, and tag).

6 Displacement is stored in the D register.

7 *a* If tag = 00, I register transfers to A register.
b If tag $\neq$ 00, the specified XR transfers to A register.

8 Displacement (D register) is added to A register.

Execute Cycle

9 A register transfers to SAR (effective address).

10 U register transfers to A register.

11 SAR addresses data word.

12 Data word transfers to B register.

13 B register loads into A register (via D register).

Two-word load instruction, direct addressing
Instruction Cycle 1

1 A register transfers to U register.

2 I register transfers to SAR (I register is then incremented).

3 SAR addresses the memory location containing the instruction (first word).

4 Memory location transfers to B register and Out-bus.

5 Control registers store various parts of the instruction (op code, format, and tag).

6 If tag $\neq$ 00, the specified XR transfers to A register.

Instruction Cycle 2

7 I register transfers to SAR (I register is then incremented).

8 SAR addresses second word of instruction.

9 Second word of instruction (address) is read into B register.

10 Address (from B register) is stored in D register.

11 *a* If tag = 00, D register transfers to A register.
b If tag $\neq$ 00, D register is added to A register (A register contains contents of XR).

Execute Cycle

12 A register transfers to SAR (effective address).

13 U register transfers to A register.

14 SAR addresses memory at effective address (data word).

15 Data word transfers to B register.

16 B register loads into A register (through D register).

Central-processor communication with the controls[1]

Direct program control of the controls

Pc direct programmed control of I/O devices is on the basis of single-word or character-at-a-time transfers for each XIO instruction executed. One data word or character is transferred to or from Mp to K. The XIO instruction specifies an I/O Control Command (IOCC) with a function of Control, Sense, Read, or Write to a controlled device. This command is either directly to a device or to a Pio.

It is possible for the program sequence to execute an XIO instruction to a device that is busy responding to a previous XIO instruction. Each device has a Busy indicator, which signals whether or not the device can accept data or control information. (Incorrect program sequence timing may cause undetected errors.)

[1]IBM name: Adapter or Device Adapter.

It is possible for a device operating synchronously with the program to request a data word transfer before the program sequence is ready to service the request. Devices with this potential have a "program check" indicator to signal when data have been lost (that is, Pc has not kept up with the device).

Execute I/O instruction (XIO)

This instruction is used for programmed I/O operations and to initialize Pio; it may be either one or two words in length, as specified by the F bit. In the two-word instruction the address is either a direct or indirect address, as specified by the IA bit. For proper operation the effective address must be an even address. The effective address is used to select a two-word I/O Control Command (IOCC) from storage.

The IOCC specifies the I/O operation, I/O device, and core storage address. The format of the two-word IOCC follows, with an explanation of the assigned fields:

Area := *IOCC*[*1*]⟨*0:4*⟩. The area field specifies a unique segment of I/O which may be a single device (1442 Card Read-Punch, 1443 Printer, etc.) or a group of several units (magnetic-tape drives, serial I/O units, contact sense units, etc.). (Area 00000 is used to address system devices such as the console and the Interrupt Mask Register.)

Function := *IOCC*[*1*]⟨*5:7*⟩. The primary I/O functions are specified by the 3-bit function code of the IOCC:

000 Removes an I/O device from on-line status and places it in a "free" mode.

001 Write
Transfers a single word from storage to an I/O unit. The address of the storage location is provided by the Address field of the I/O Control Command.

010 Read
Transfers a single word from an I/O unit to storage. The address of the storage location is provided by the Address field of the I/O Control Command.

011 Sense Interrupt Level
Directs the selected I/O device to make its status available in the Accumulator as the Interrupt Level Status Word (ILSW).

100 Control
Causes the selected device to interpret the address and/or Modifier of the IOCC as a specific control action. Examples are feed card and load interrupt mask register.

101 Initialize Write
Initiates a Write operation on a device or unit which will subsequently make data transfers from storage via a Pc.

110 Initialize Read
Initiates a Read operation from a device or unit which will subsequently make data transfers to storage via a Data Channel.

111 Sense Device
Reads the selected device status word into the Accumulator. A Device Status Word (DSW) and the Process Interrupt Status Word (PISW) are sensed with this instruction.

If Area 00000 is specified, the Console status and Interval Timer status may be brought into the Accumulator as specified by a unit address code in the Modifier field.

The current contents of the Accumulator are destroyed by the execution of Sense Interrupt Level, Sense Device, Initialize Read, Initialize Write, Read, or Write.

Modifier := *IOCC*[*1*]⟨*8:15*⟩. This 8-bit field provides additional detail for either Function or Area. For example, if the Area specifies a disk and if the Function specifies Control (100) then a particular modifier code specifies the direction of the Seek operation. In this case, the Modifier serves to extend the function.

If, however, the Area specifies a group of I/O devices, and if the Function specifies Write (001), then the particular unit address is specified by the modifier.

Address := *IOCC*[*0*]⟨*0:15*⟩. The meaning prescribed for this 16-bit field is dependent upon the Function specified by this I/O Control Command:

1 If Function is Initialize Write (101) or Initialize Read (110), then Address specifies the starting address of a table in storage (an I/O block). The contents of this table are data words and control information.

2 If Function is Control (100) and if, for example, Area specifies the 1443 Printer, the Address may specify a specific control action.

3 If Function is Sense (011 or 111), the Address field is ignored. Instead, an increment of time equivalent to a memory cycle is taken, during which the selected I/O device or Interrupt Level places its status word in the accumulator.

4 If Function is Write (001) or Read (010), the Address specifies the storage location of the data word.

XIO execution interpretation process

1 The EA of the XIO is developed in the accumulator (A) and routed to the Storage Address Register (SAR) to locate the IOCC (as for any EA).

2 Bit position 15 of SAR is forced on to select the EA + 1 where the IOCC Area, Function, and Modifier are found.

3 The Area, Function, and Modifier are routed through the B register to the Out-bus to the control of the device specified by the Area.

4 Bit position 15 of SAR is turned off to allow the address portion of the IOCC word to be transferred from the Mp location specified by the Effective Address (EA) to the B register.

5 If the Function is an Initialize Read, Initialize Write, or Control, the address part of the IOCC is routed through the B register to the Out-bus. The address part of the Initialize Read/Write IOCC goes to the Channel Address Register (CAR) of Pio. If the Function is Read or Write, the address is routed from the B register through the A register to the SAR. SAR addresses the memory location to or from which the data are transmitted.

Interval timers

Three timers are provided to supply real-time information to the program. They are in core-storage locations 0004 (Timer A), 0005 (Timer B), and 0006 (Timer C). Each timer is incremented according to its associated or permanent time base and can be hardwired to be 0.125, 0.250, 0.5, 1, 2, 4, 8, 16, 32, 64, or 128 milliseconds.

The timers can be started or stopped under program control. When the count reaches zero, an interrupt is requested on the level assigned to the timers.

Interrupt

The interrupt feature provides an automatic branch from the normal program sequence, based upon an external condition. A maximum of 24 external interrupt levels (groups) are available, arranged in order of priority. Twelve external interrupt levels are standard. Each interrupt level has a unique core-storage address assigned to it. Several devices may be connected to a single interrupt level, and program polling can be used to differentiate the possible signals causing the interrupt. The Interrupt Level Status Word, ILSW, is used to identify the specific condition causing its interrupt level to request service.

Internal interrupt. When any one of the following error conditions occur, there is an internal interrupt in Pc: an invalid op code; a Mp parity error (an even number of bits); a storage-protect violation; and Channel Address Register check error. The internal interrupt takes priority over all external interrupts and cannot be masked.

A mask register exists for the masking and unmasking of interrupt levels. An interrupt level that is masked cannot initiate a request for service until it has been unmasked.

Device status word (DSW). DSW indicators usually fall into three general categories:

1 Error or exception interrupt conditions

2 Normal data or service-required interrupts

3 Routine status conditions

Process interrupt status word indicators (PISW). The PISW indicators are physically located in Pc and are turned on by events external to the computer, e.g., contact closures or voltage shifts.

IO processors[1]

The Pc initializes each Pio with an XIO instruction. The Pio has priority to the extent that, when the I/O device is ready to send or receive a data word, the Pc is stopped while the word transfers to or from core storage. Pc data and conditions are undisturbed except for the memory locations that receive data from an input device.

I/O devices that are to be operated concurrently must be on separate Pio's.

The XIO instruction for a Pio specifies an I/O Control Command (IOCC) with a function of Initialize Read or Initialize Write. However, even though a device operates with a Pio, the XIO instructions in Pc are used to sense device status and for control.

Registers

Channel address register. The Channel Address Register (CAR) is a 16-bit register used to store the Mp address of the next word that will be addressed by the Pio. Each Pio has a CAR. Pio and its associated CAR are selected when their assigned I/O device is selected by the Area Code and Modifier of an IOCC word. CAR is incremented by 1 after each transfer of its contents to CAB.

[1]IBM name: Data Channel (DC).

Channel address buffer. A common Channel Address Buffer (CAB) is used by all Channel Address Registers to address Mp. When a cycle steal request occurs, the CAR for the requesting Pio is transferred into the Channel Address Buffer.

Channel-address-register check bit. Channel Address Register (CAR) checking is provided to ensure that the first word addressed by a selected CAR is the first word of the correct data table. Thus the check determines if a Pc program has set up the Pio program correctly.[1] A CAR check is made for all devices after the address from the IOCC word is transferred to the selected CAR. A bit-by-bit comparison is made between the contents of the selected CAR and the contents of the B register. If any of the corresponding bits are not equal, a CAR check error has occurred. This CAR check error terminates the Pio task and initiates an internal interrupt.

Word count register. A Word Count Register is provided in each Pio. The Word Count Register is loaded with the contents of the word-count portion of the data table, ⟨2:15⟩. This register is decremented each time a data word is transferred from (to) the data table.

Scan control register. A Scan Control Register is provided in each Pio that has chaining ability. Scan Control register bits are stored in the first word of the first data table (bit positions 0 and 1) and in the second word (bit positions 0 and 1) of the second data table and all subsequent data tables in a chain.

The Scan Control Register controls the I/O device and the Pio operation at the end of the data table as follows: single scan of data table and stop with an interrupt; single scan of data table and stop (no interrupt); continuous scan of this data table or a different data table with an interrupt at the end of this table; and continuous scan of this data table or a different data table with no interrupt.

The IO processor program operation

The sequence of steps for a Pio program is given below. The memory map or format of the program is shown in Fig. 7.

1 Pc issues an XIO instruction which references the IOCC word and initializes Pio.

2 The Area Code and Modifier of the IOCC select the I/O device. Function specifies the type of operation (Initialize Read or Initialize Write, etc.).

3 *a* The address portion of the IOCC word is stored in CAR for the selected Data Channel and I/O device.

b A CAR check is made between the selected CAR and the B register.

4 A cycle steal is requested by Pio; CAR transfers to CAB.

5 CAB addresses core storage for the first word of the data table while CAR is being incremented by 1.

6 The first word of the data table contains
a Scan Control bits (bit positions 0 and 1)
b Word Count (bit position 2 to 15)
These are transferred to their respective registers in the I/O device. This is the end of the first cycle steal.

7 When another cycle-steal request from Pio occurs, CAR, which was incremented in step 5, now transfers the next higher address to CAB. CAB then addresses core storage while CAR is being incremented.

8 The first data word is transferred to or from the I/O device via the B register and Data Channel. The Word Count Register in the I/O device is decremented by 1. This is the end of the second cycle-steal cycle.

Steps 7 and 8 now continue on a cycle-steal basis; that is, they occur as the I/O device requests data transfers. The CAR is incremented with each data transfer and the WCR is decremented. This sequence continues until the last data word of the data table is transferred. The last word transfer is sensed by the WCR reaching zero or through some indicator in the device. If the device does not have chaining ability, no more demands for data transfer are made until the device is reinitialized with another XIO instruction.

Chaining. These steps are for the second and all subsequent data tables. See above for steps 1 through 8.

9 The contents of the word following the last data word in the first data table are transferred to CAR. This word must contain the address of the next data table.

10 *a* When the next cycle is requested, CAR is transferred to CAB to address core storage. The contents of the first word of the next data table is transferred to the B register. This word must contain the address of itself.

10 *b* CAR check is performed and CAR is incremented by 1.

11 When the next cycle steal is requested, CAR is transferred to CAB and CAB addresses Mp. The Scan-control bits and Word-count bits are transferred from the second word of

[1] Not a completely arbitrary program fault to check, since processors are involved.

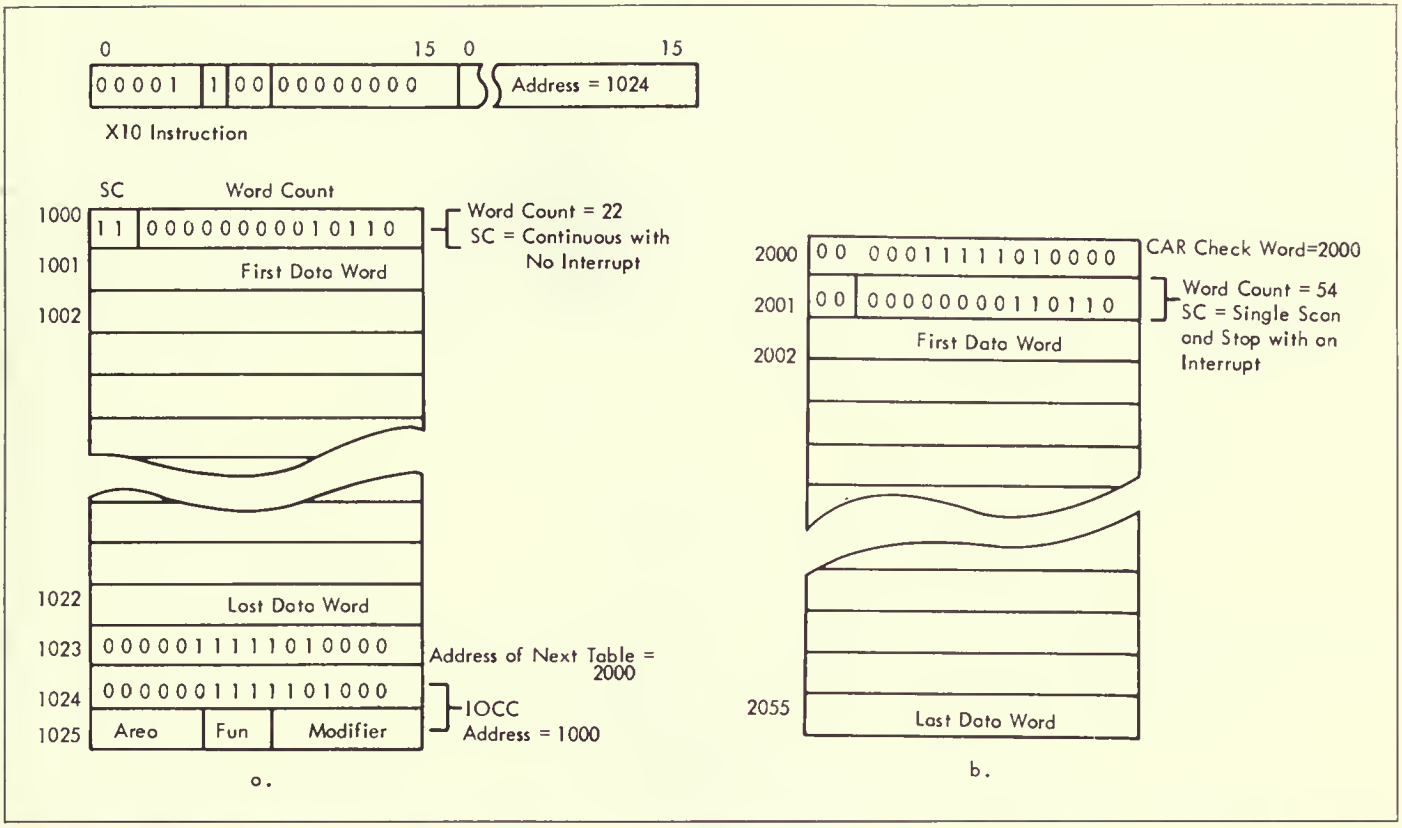


Fig. 7. IBM 1800 data-channel tables for chaining memory maps. (*a*) First data table; (*b*) second data table. (*Courtesy of International Business Machines Corporation.*)

the data table to their respective registers. CAR is incremented by 1.

12 Data are transferred to (from) the I/O device on a cycle-steal basis via the B register and the Data Channel. CAB addresses core storage to transfer a data word to the B register. Each time CAB addresses core storage, CAR is incremented by 1. When the next cycle-steal request occurs, CAR is transferred to CAB. The Word-count Register is decremented for each word transferred.

13 When the last data character is transferred (word count is decremented to zero), operation will continue as specified by the Scan Control Register. (See above section for *Scan-Control Register*.)

Special data channels

The four Pio types for special functions are:

1 Analog input (block data transfers, and comparisons of analog inputs for limits)

2 Digital input/output

3 Analog output

4 Digital output

Analog-input data channels. Memory maps (Fig. 8*a* and *b*) illustrate the command formats interpreted in the Analog Data Channel programs. A list of limit values is placed in a table (Fig. 8*a*), and each analog input is compared with the limits. The operation sequence is: Read a specific addressed analog voltage, called the multiplex[1] point (mpx); compare the input voltage with the limits stored in the table following the analog address (the limit word contains a high and low value in bits ⟨0:7⟩ and ⟨8:15⟩, respec-

[1]The IBM multiplexor is an S which allows multiple inputs to be read into the T(Analog to Digital Converter) sequentially.

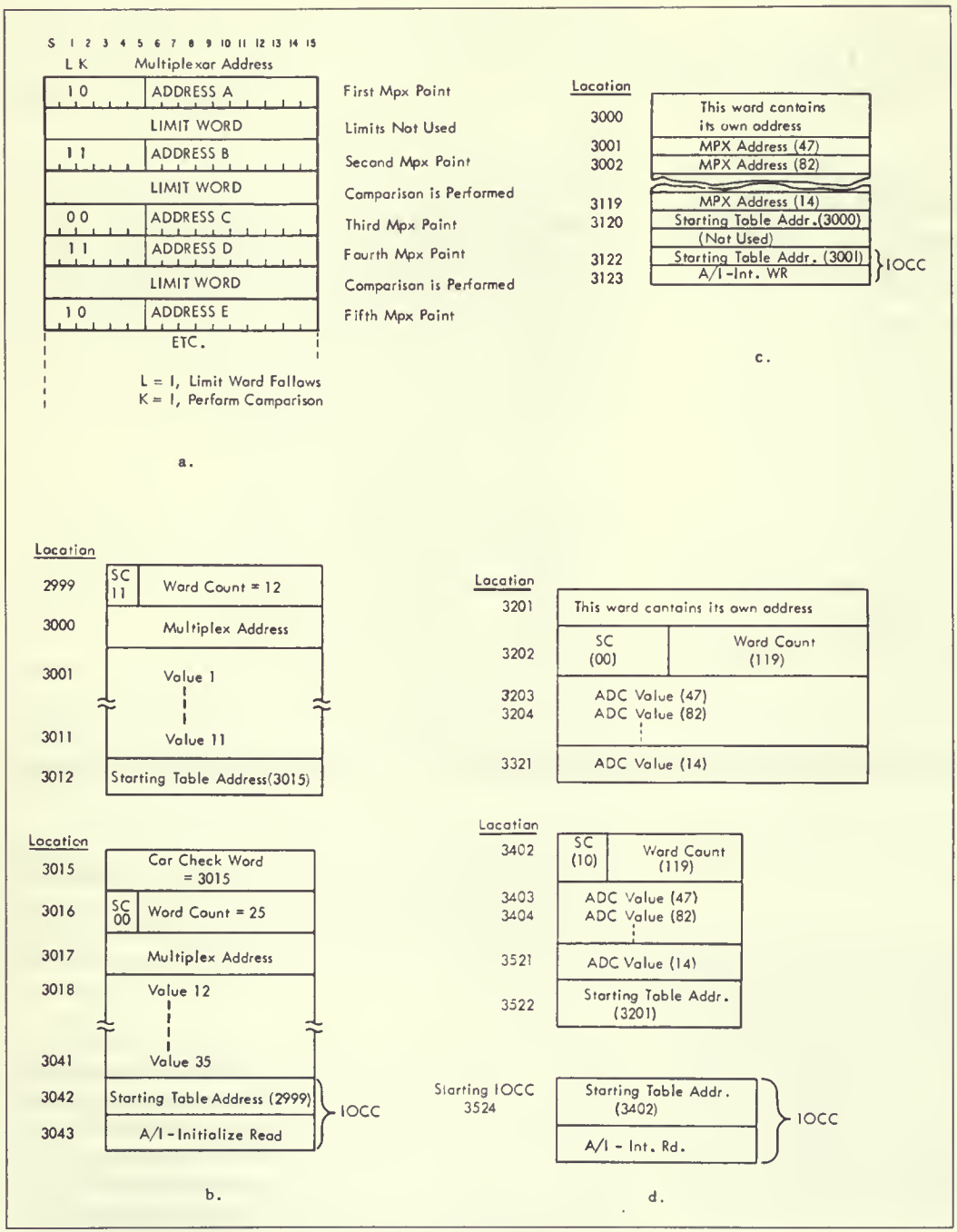


Fig. 8. IBM 1800 data-channel analog-input instruction format and memory maps. (a) Multiplexor address table with limit words for comparisons. (b) Data table, chained sequential control. (c) Multiplexor address table, random addressing. (d) Analog-to-digital converter storage tables, random addressing (used with a second data channel).

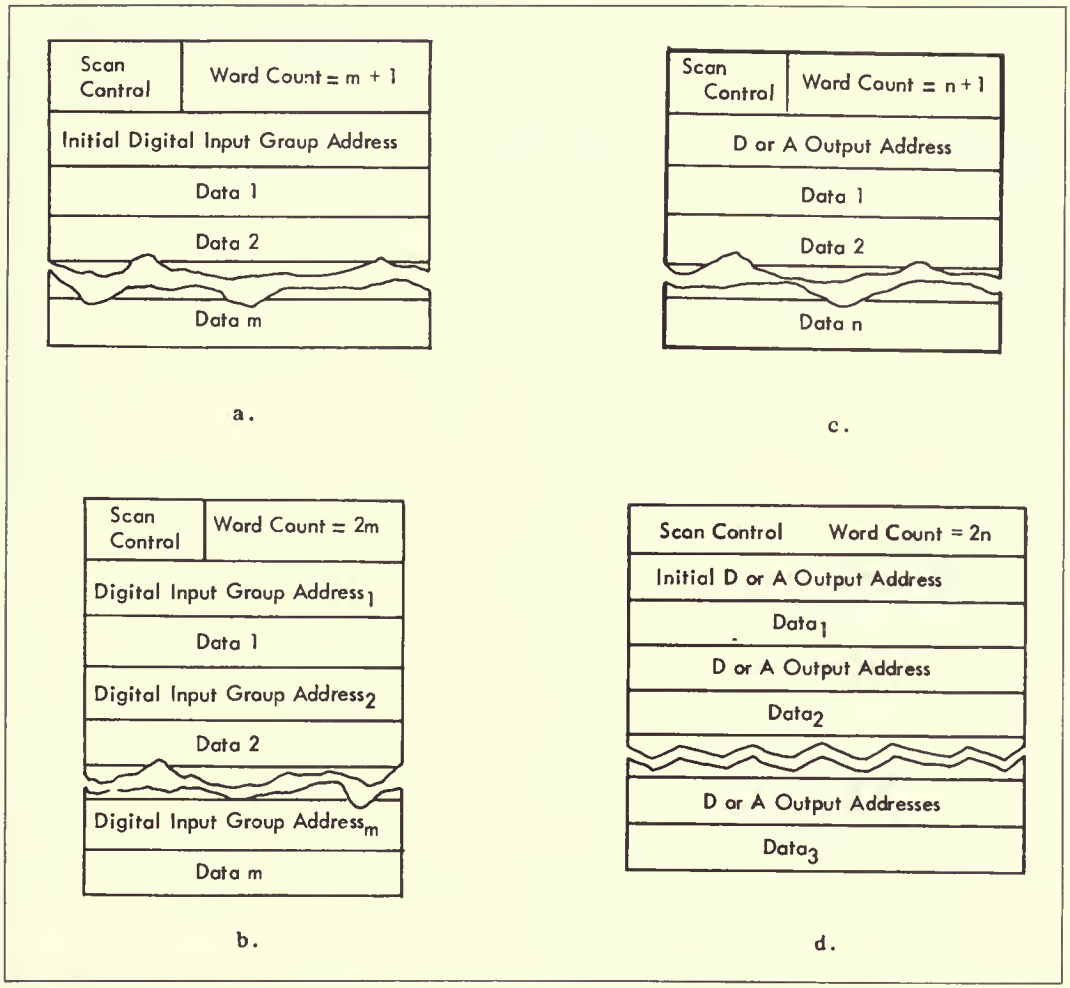


Fig. 9. IBM 1800 data-channel digital or analog-output instruction formats and memory maps. (*a*) Digital input, sequential; (*b*) digital input, random addressing; (c) digital or analog output, sequential; (*d*) digital or analog output, random addressing. (Courtesy of *International Business Machines Corporation*.)

tively); and if the analog-input value lies outside the limit range, initiate an interrupt.

Figure 8*b* describes a second use of this data channel. Pio accepts a sequence of analog inputs and packs them into a table following the address initiation instruction. The analog inputs from the T's are either fixed or selected in a cyclic fashion from a Multiplexor.

Two Pio's can be used concurrently: One Pio controls the input from a series of analog-input addresses (Fig. 8*c*); the second Pio packs the corresponding analog values in a second table (Fig. 8*d*).

Digital-input data channels. Digital parameters or events can be read into Mp under the control of a Digital-input Data Channel. The memory map (Fig. 9*a*) shows the control format for selecting and inputting a block or sequence of external data. The memory map (Fig. 9*b*) illustrates a more general ability to address inputs at random and read them into succeeding Mp locations.

Digital- and analog-output data channels. Memory maps (Fig. 9*c* and *d*) show the program format used by the Digital- or Analog-output Data Channels. These channels output selected data points

to external analog or digital K's. This Pio is similar to the Digital-input Data Channel.

Conclusions

We have tried to show a typical, third-generation computer used for process control. Many of the facilities the 1800 possesses are general. The Pio's are rather special, designed to monitor and control a process, independent of Pc. Although the Pio's are powerful (by providing parallel data transmission), their use, like other multiprocessing systems, is nontrivial. The Pc ISP is fairly straightforward, and one should write a program using it to appreciate its simplicity.

APPENDIX 1 THE IBM 1800 ISP DESCRIPTION

```
                                Appendix 1

                        IBM 1800 ISP Description

Pc State
    A<0:15>                          Accumulator
    Q<0:15>                          Accumulator Extension for multiplier, quotient and double
                                      length
    I<0:15>                          Instruction Location Counter
    XR[1:3]<0:15>                    Index Registers
    Ov                               Overflow Indicator
    C                                Carry Indicator
    Run                              denotes running computer

Mp State
    M[0:FFFF16]<P,S,0:15>            Mp with Parity and Protect bits

Pc Console State
    Check Stop Switch                Pc stops if storage protect violation occurs
    WSPB Switch                      Write Storage Protect Bits; enables the writing of bits in
                                      a word
    SPV Indicator                    Storage Protect Violation indicator; set to 1 if a memory
                                      reference is made to a protected word

Instruction Format
    instruction/i[0:1]<0:15>
      op<0:4>    := i[0]<0:4>        operation code
      shop<0:7>  := op□i[0]<5,8,9>   shift operation code count
      f          := i[0]<5>          format; specifies a 1 or 2 word instruction
      t<0:1>     := i[0]<6:7>        tag: index register specification
      d<8:15>    := i[0]<8:15>       displacement or short address
      dsgn<0:15>:= sign_extend(d<8>□d<9:15>)
      a<0:15>    := i[1]<0:15>       address
      ia         := i[0]<8>          indirect address bit
      bo         := i[0]<8>          branch out bit
      cond<0:5> := i[0]<10:15>       conditions for test

Effective Address Calculation Process
    z<0:15>:= (                      effective address
      (t = 0)∧ ¬ f       →(dsgn + 1);      1 word, relative
      (t ≠ 0)∧ ¬ f       →(dsgn + XR[t]);  1 word, relative, indexed
      (t = 0) ∧ f ∧ ¬ ia→ a;               2 word, direct
      (t ≠ 0) ∧ f ∧ ¬ ia→(a + XR[t]);      2 word, direct, indexed
      (t = 0) ∧ f ∧ ia   → M[a];           2 word, indirect
      (t ≠ 0) ∧ f ∧ ia   →(M[a + XR[t]]))  2 word, indirect, indexed
    z'<0:15> := (¬ f →(dsgn + 1);    effective address for index register instructions
                  f ∧ ¬ ia →a;
                  f ∧ ia →M[a])
```

APPENDIX 1 THE IBM 1800 ISP DESCRIPTION (Continued)

```
zd<0:15> := (¬z<15> →z + 1;                          process for locating second operand for double length
             z<15> →z)
xi<0:15> := (¬ f →dsgn;                              index increment
             f ∧ ¬ ia →a;
             f ∧ ia →M[a])
s<0:5>   := (                                        shift count calculation
   (t = 0) →d<10:15>
   (t ≠ 0) →XR[t]<10:15>)

Instruction Interpretation Process

Run → (instruction[0:1] ←M[I:I + 1]; next            fetch
      ¬ f →(I ←I + 1); f →(I ←I + 2); next           1 or 2 word instruction
      Instruction_execution)                         execute

Instruction Set and Instruction Execution Process

Instruction_execution := (
Load and Arithmetic
   LO  (:= op = 11000) → (A ←M[z]);                  load accumulator
   LDO (:= op = 11001) → (A□Q ←M[z]□M[zd]);           double load
   STO (:= op = 11010) → (M[z] ←A);                  store accumulator
   STD (:= op = 11011) → (M[z]□M[zd] ←A□Q);          double store
   A   (:= op = 10000) → (Ov,C□A ←A + M[z]);         add
   AD  (:= op = 10001) → (Ov,C□A□Q ←A□Q + M[z]□M[zd]); double add
   S   (:= op = 10010) → (Ov,C□A ←A - M[z]);         subtract
   SD  (:= op = 10011) → (Ov,C□A□Q ←A□Q - M[z]□M[zd]); double subtract
   M   (:= op = 10100) → (A□Q ←A x M[z]);            multiply
   D   (:= op = 10101) → (Ov,Q ←A□Q / M[z];          divide
                          A ←A□Q mod M[z]);
Logical instructions
   AND (:= op = 11100) → (A ←A ∧ M[z]);              logical and
   OR  (:= op = 11101) → (A ←A ∨ M[z]);              logical or
   EOR (:= op = 11110) → (A ←A ⊕ M[z]);              logical exclusive or
Compare
   CMP (:= op = 10110) → ((A < M[z]) → (I ← I + 1);  compare
                          (A = M[z]) → (I ← I + 2));
   DCM (:= op = 10111) → ((A□Q < M[z]□M[zd]) → (I ← I + 1);  double compare
                          (A□Q = M[z]□M[zd]) → (I ← I + 2));
Shifts
   SLA (:= shop = 00010□0□00) → (                    shift left logical
      A ←A x 2^S {logical}; C ←A<s-1>);
   SLT (:= shop = 00010□0□10) → (                    shift double left logical
      A□Q ←A□Q x 2^S {logical}; C ←A<s-1>);
   SRA (:= shop = 00011□0□00) → (A ←A / 2^S {logical}); shift right logical
   SRT (:= shop = 00011□0□10) → (A□Q ←A□Q / 2^S):    shift right A and Q
   RTE (:= shop = 00011□0□11) → (A□Q ←A□Q / 2^S {rotate}): rotate right A and Q
   SLCA(:= shop = 00010□0□01) → (                    shift left and count A
```

APPENDIX 1 THE IBM 1800 ISP DESCRIPTION (Continued)

```
      (t = 0) → (A ← A x 2^S; C ← A<s-1>);
      (t ≠ 0) → (A ← normalize(A);
                 C□XR[t]<10:15> ← normalize_exponent(A);
                 XR[t]<8,9> ← 0));
   SLC (:= shop = 00010□0□11) → (¬ ((s = 0) ∨ A<0>) → (          shift left and count
      (t = 0) → (A□Q ← A□Q x 2^S; C ← A<s-1>);
      (t ≠ 0) → (A□Q ← normalize(A□Q);
                 C□XR[t] ← normalize_exponent(A□Q))));
   LOX (:= op = 01100) → ((t = 0) → (I ← z');                    load index or instruction counter
                          (t ≠ 0) → (XR[t] ← z'));
   STX (:= op = 01101) → ((t = 0) → (M[z'] ← I);                 store index or instruction counter
                          (t ≠ 0) → (M[z'] ← XR[t]));
   STS (:= op = 00101) → (                                       store status
      (f ∧ bo) → M[z]<P>← cond<15>;
         ¬bo → (M[z]<8:15> ← 00000□C□0v; C□0v ← 00));
   LDS (:= i[0] = 00100□0□00□000000%%) → (C ← i[0]<14>;          load status
                                          0v ← i[0]<15>);
   BSC (:= (op = 01001) ∧ ¬ i<9>) → (                            branch or skip on condition
      ( skip_condition ∧ ¬ f) → (I ← I + 1);
      (¬skip_condition ∧ f) → (I ← z);
      d<15> → 0v ← 0);
      skip_condition := (
         (¬0v ∧ d<15>) ∨                                         overflow off
         (¬ C ∧ d<14>) ∨                                         carry off
         (A<15> ∧ d<13>) ∨                                       Accumulator even
         ((A > 0) ∧ d<12>) ∨                                     Accumulator greater than zero
         (A<0> ∧ d<11>) ∨                                        Accumulator negative
         ((A=0) ∧ d<10>))                                        Accumulator zero
   BOSC (:=(op = 01001) ∧ I<9>) → (                              branch out of interrupts
      (skip_condition ∧ ¬ f) → (I ← I + 1; Interrupt ← 1);
      (¬ skip_condition ∧ f) → (I ← z; Interrupt ← 1);
      d<15> → (0v ← 0));
   BSI (:= op = 01000) → (                                       branch and store instruction register
    ¬f → (I ← z + 1; M[z] ← I);
      f → (d<15> → 0v ← 0);
    ¬skip_condition → (I ← z + 1; M[z] ← I));
   MDX (:= op = 01110) → (                                       modify index and skip
     (t = 0) ∧ ¬ f → (I ← I + dsgn);                             local branch
     (t = 0) ∧ f → (M[a] ←M[a] + dsgn;
        (Msum=0) ∨ (M[0]<0> ⊕ Msum<0>) → (I ← I + 1));           result zero or sign change
           Msum<0:15> := (M[a] + dsgn)
     (t ≠ 0) →(XR[t] ←XR[t] + xi;
        (Xsum=0) ∨ (XR[t]<0> ⊕ Xsum<0>) → (I ← I + 1)));         result zero or sign change
           Xsum<0:15> := (XR[t] + dsgn)
  Wait (:= I = 3000_16) → (I ← I - 1);
```

APPENDIX 1 THE IBM 1800 ISP DESCRIPTION (Continued)

```
IO Control Instruction:
   XIO (:= op = 00001) → (                                  Execute I/O, not defined
      IOCC[0:1] ← M[z]□M[zd]; next
      Execute␣IO␣instruction)
              )                                             end Instruction␣execution
IO Instruction Format:
   IO Address<0:15>         := IOCC[0]                      address if IO data
   IO Device or Area<0:4>   := IOCC[1]<0:4>                 io device name
   IO Function<5:7>         := IOCC[1]<5:7>
   IO Modifier<8:25>        := IOCC[1]<8:15>                device function details
      Device mode off line := (IO Function = 0)
      Device mode write     := (IO Function = 1)
      Device mode read      := (IO Function = 2)
      Device mode sense interrupt level := (IO Function = 3)
      Device mode control := (IO Function = 4)
      Device mode initialize write := (IO Function = 5)
      Device mode initialize read  := (IO Function = 6)
      Device mode sense     := (IO Function = 7)
```