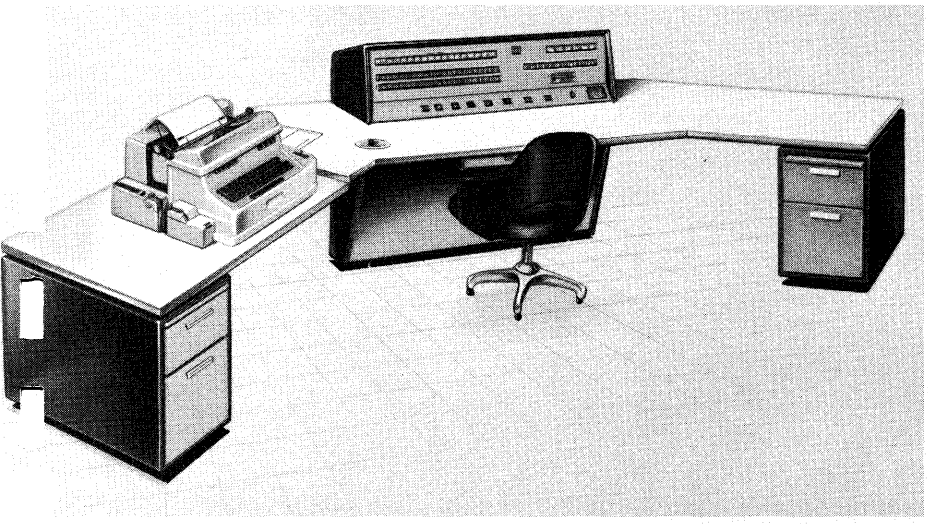
12-G 1



PHILCO®

2000

INFORMATION MANUAL

2 MICROSECOND CORE STORAGE

MODEL 212 COMPUTER

REAL TIME SYSTEM

PHILCO 2000 SYSTEM

INFORMATION MANUAL

PHILCO CORPORATION
Government and Industrial Group
Computer Division
3900 Welsh Road
Willow Grove, Pennsylvania

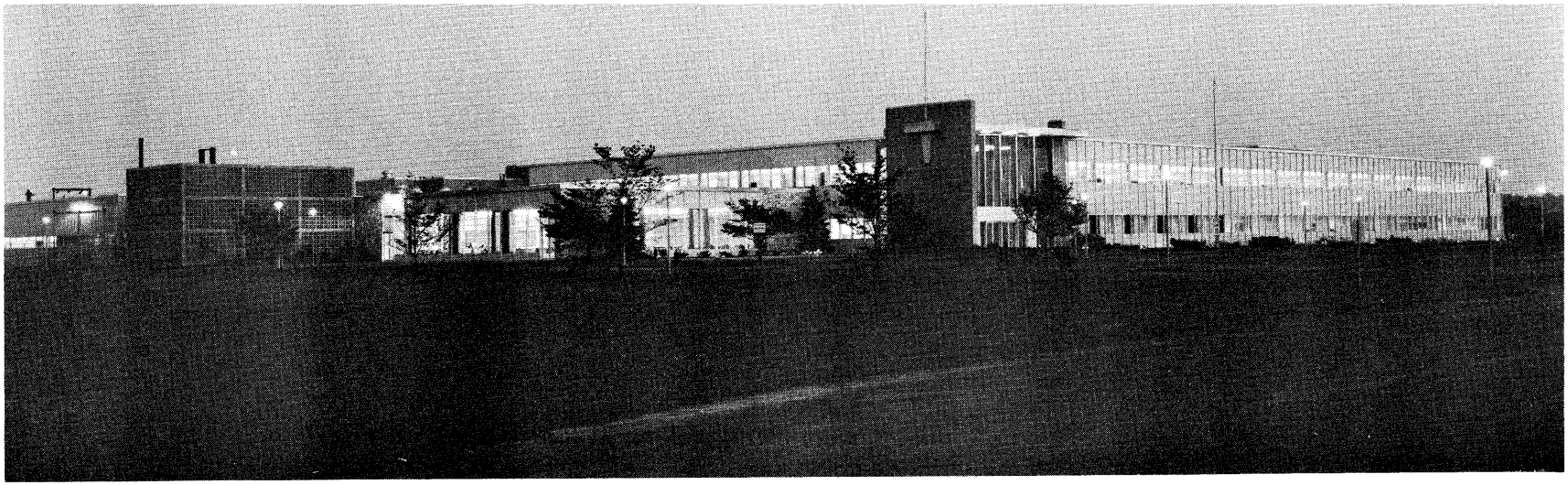
CONTENTS

AN INTRODUCTION

TO

PHILCO CORPORATION'S

COMPUTER DIVISION



Philco Computer Division's new multi-million dollar facility in Willow Grove, Pennsylvania, just a few miles north of Philadelphia proper.

A. PHILCO FACILITIES

Philco Corporation is a major manufacturer in the electronics industry, with 15 plants in the Philadelphia area and additional plants in nine cities across the country.

Facilities of the Government and Industrial Group are centralized in four plants in Philadelphia. A total floor area of over 1,700,000 square feet is devoted to the production of electronic and ordnance equipment. Administrative offices of the division and engineering and research activity are located in the main plant at 4700 Wissahickon Avenue, Philadelphia 44, Pennsylvania.

In February of 1960, all Philco computer activities moved to a new plant located in Willow Grove, Pennsylvania (suburban Philadelphia). The new Philco plant, with more than 200,000 square feet of floor space, contains national sales offices, and research, engineering, and manufacturing facilities for production of the PHILCO 2000 large-scale electronic data processing system as well as Philco industrial process control computers, and mobile field computers for the military.

The Government and Industrial Group Engineering Department includes both electronic and mechanical sections, and combines the abilities of approximately 1100 electrical and mechanical engineers and their technical assistants. The Military Research Department has an additional group of scientists and engineers, making a total of approximately 1,300 persons.

The laboratory areas devoted to the engineering phases total approximately 161,000 square feet. This area is divided into several large laboratories, each of which is fully equipped with the associated test equipment and facilities required to perform its particular assignments. Some of these laboratories deal only with the design and development of prototype equipment. Others deal with the evaluation of components and unit parts. There are, in addition, various specialized areas such as a dust-free, air-conditioned optical laboratory.

Advanced work on vacuum tubes and transistors is performed in research and engineering laboratories located in the Philco Tube Plant, Lansdale, Pa., and a fully automated transistor production line at Philco's Spring City, Pa. plant.

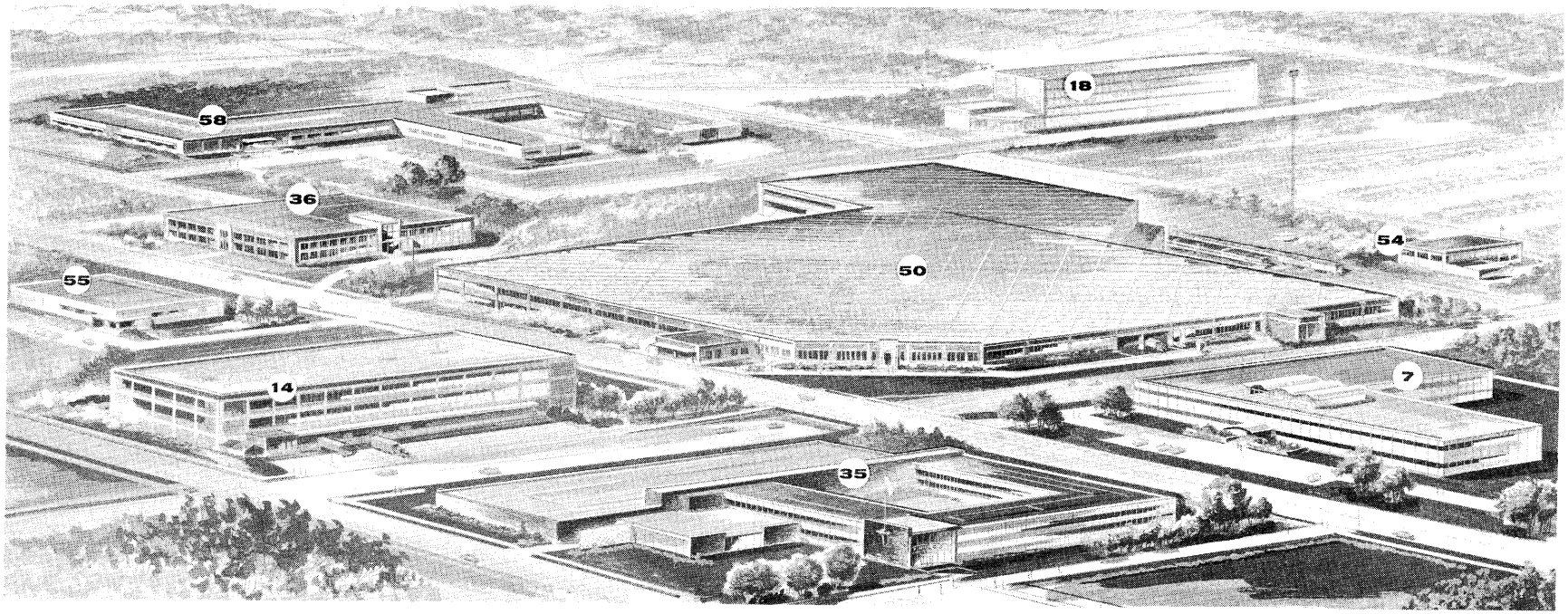
The Engineering Department of the Government and Industrial Group has many laboratories concerned with the development and design of equipment in the fields of electronics, mechanics and physics. Several of these specific areas are described in the following paragraphs.

The Government and Industrial Group maintains a complete gage and measurements laboratory to insure compliance with exacting tolerances by our suppliers as well as ourselves. A complete chemistry laboratory is maintained to support all divisional activities. A modern test equipment laboratory is charged with responsibility of supplying production departments with necessary electronic test equipment for testing all manufactured assemblies and systems. In addition, the test engineering section fabricates and services test equipment for all engineering departments.

The engineering laboratories service group includes a components laboratory staffed by specialists in the fields of connectors, motors, meters, relays, capacitors, transformers, vacuum tubes, switches, wiring, etc. These specialists, completely familiar with the latest developments throughout the industry, provide assistance, advice, and experienced know-how in selecting the best component for each particular task.

Philco is proud of the environmental facilities established in its Government and Industrial Group. These facilities are as complate as any within the geographical area, with modern facilities for production, assembly, and testing of electronic and electromechanical equipment. Production facilities include a complete machine shop augmented by various precision machine tools for the manufacture of metal parts and assemblies. Additional facilities include a sheet-metal shop, silk-screening facilities, all types of welding equipment, painting and baking booths and ovens, facilities for plating all types of finishes and printing circuit panel equipment. Skilled personnel are available for fabrication, assembly, and testing of all equipment.

The production facilities are complemented by an integrated quality control department having a production quality section and a test equipment quality section fully responsible for in-process control, and a statistical analysis section to develop and maintain procedures and check lists for the department. The quality control system at Philco has been qualified for some time under the Signal Corps Reduced Inspection Quality Assurance Plan. Also, in recent evaluation of missile contractors by Navy BuOrd, Philco received top score among its fellow missile contractors for quality performance.



PHYSICAL FACILITIES -- GOVERNMENT AND INDUSTRIAL GROUP

Since its inauguration in 1951, an aggressive expansion plan has steadily accelerated the growth of the Government and Industrial Group. The two latest facilities house Philco's Computer Division and Research Division.

PLANT 50

--Headquarters of the Group; houses research, engineering, manufacturing, environmental testing, sales and general offices for industrial, commercial, and military products and systems.

PLANT 35

--Computer development, engineering, manufacturing, sales, and general offices for commercial, industrial, and military data processing systems.

PLANT 54

--complete graphic arts facility for the preparation of proposals, manuals, and data sheets, advertising and sales promotion; includes photo, art, model, and printing departments.

PLANT 18

--complete manufacturing and testing of microwave, closed circuit television, radar, test equipment, and advanced communications products; plus replacement part subassemblies and equipment overhaul and repair.

PLANT 58

--research, development, systems analysis, human factors engineering and systems management, plus sales and general offices; engaged in numerous advanced systems development for satellite tracking and space communications systems.

PLANT 55

--research, development, engineering, production, sales, and general offices for the manufacture of special purpose electronic devices and systems.

PLANT 7

--basic research, development, and applications of new electronic and electromechanical systems.

PLANT 36

--development, engineering, and systems analysis facilities for advanced military and government data processing and communications systems.

PLANT 14

--warehousing facility for all products of the Division.

B. PHILCO CAPABILITIES

DIGITAL COMPUTERS

THE PHILCO 2000

Philco produced a major breakthrough in the state of the art with the PHILCO 2000, the first all-transistorized large-scale computer. The reputation of the PHILCO 2000 is now firmly established, as evidenced both by numerous systems in operation and millions of dollars' worth of orders for others.

- Complex Scientific Applications
- Sophisticated Data Processing Operations for
  - Government Agencies
  - Commercial Firms

MOBILE COMPUTERS - BASICPAC Series

Philco has developed this series of vehicular mounted computers to provide the Military with high-speed, high-capacity computational and data processing systems for operation in the field by ground forces. Designed to operate under the most rugged environmental conditions, these mobile computers possess outstanding capabilities for such applications as:

- Combat Control
- Artillery and Missile Fire Control
- Missile and Drone Guidance
- Special Commercial Applications
- Combat Logistic Control
- Data Filtering
- Combat Intelligence Processing

FIRST IN TRANSISTORIZED DATA PROCESSING

C. SOME DETAILS OF PHILCO'S COMPUTER EXPERIENCE

The Philco Computer Division has been responsible for several major steps forward in the design and development of digital systems in general, and is thoroughly experienced in computers of all descriptions. Special design and production techniques ensure unusually high reliability and ease of maintenance of completed systems.

Philco has pioneered in the development of high-speed transistors for computer applications, transistorized circuits, transistorized data processing systems and real-time control computers. Transistorized data processing systems developed by Philco include the C-series computers, the large-scale PHILCO 1000 and 2000 systems, and the medium-sized general purpose computers developed for the Signal Corps Laboratories under the FIELDATA program.

The C-series of computers work in closed-loop control systems where the input and output are provided by analog/digital converters working from shaft position, synchro voltages, and currents to and from magnetic amplifiers. The C-series is also employed to take information from and provide information to human operators via display systems and control panel.

The most widely known model of the C-series computers was developed by Philco for marketing by Leeds and Northrup Co. Known as the C-3000, it has important applications in industrial process control and in scientific and engineering computation, including data processing.

In addition to work on control computers for airborne applications, the Engineering Department has developed two large-scale data processing systems for business, scientific, and military applications. They are the PHILCO 1000 and the PHILCO 2000, both of which represent a major advance in the state of the art.

The first of these, the PHILCO 1000, was built for a Government agency. It is a scientific computer implemented by direct-coupled circuits. The PHILCO 2000 began as a large-scale, high-speed, all transistorized computer developed for the Navy Bureau of Ships, and called "CXPQ". The CXPQ was the forerunner of the modern PHILCO 2000 systems.

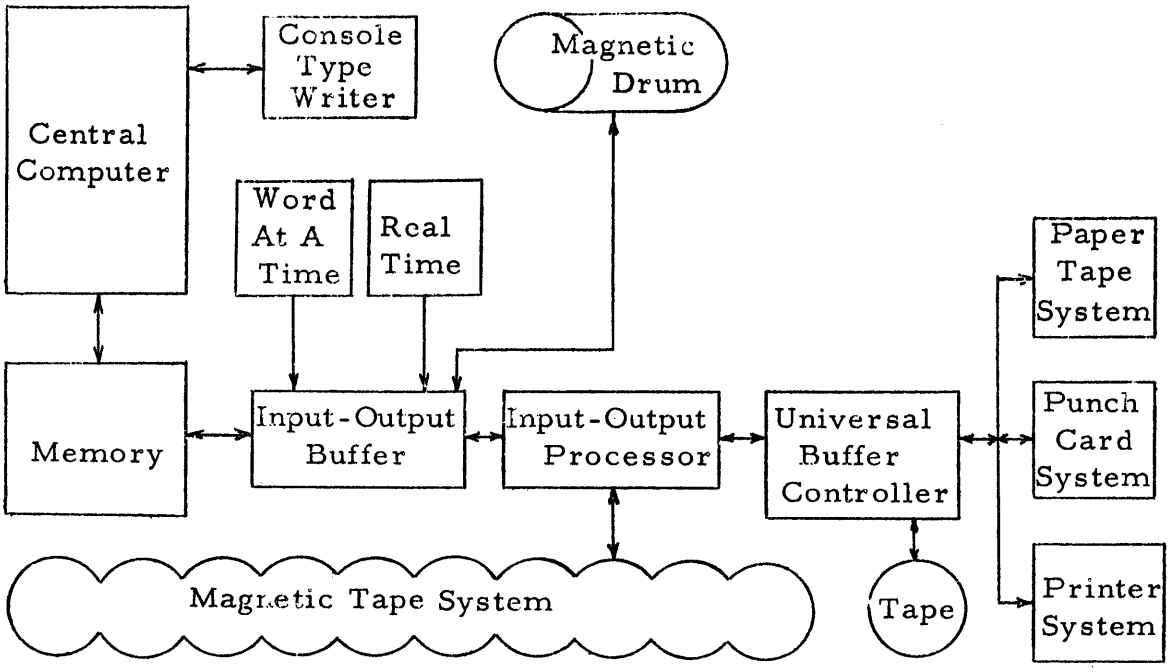
Philco has also designed and developed a medium-sized, stored-program, general-purpose computer for the Signal Corps Engineering Laboratories, Fort Monmouth, N. J. This machine, BASICPAC, is designed to process and compute data and to meet environmental requirements under Army combat conditions.

In addition to the work described above, which provides Philco with a wealth of experience, Engineering and Research groups are currently engaged in projects designed to advance the state of the art in transistor circuits, logical design, memory, display, switching, and auxiliary devices.

The PHILCO 2000 serves as a prime example of the Philco policy that reliability must be designed into equipment. System implementation and system reliability are made a single program, carried through from the initial study phase to the field evaluation of production units. At every level, engineering reliability design is reinforced and augmented by comprehensive quality assurance and statistical evaluation. Further, a continuous research effort is devoted to advancing the state of the art in the field of electronic equipment reliability. Considerable work has recently been done in the Philco computer laboratories and research division on the automatic generation of computer diagnostic routines and on the reliability of electronic equipment under various types of nuclear radiation.

THE PHILCO 2000 SYSTEM

GENERAL DESCRIPTION OF THE PHILCO 2000 SYSTEM



Functionally, the PHILCO 2000 System consists of three main sections: the Central Computer, Storage, and Input-Output. The Central Computer controls the system and performs the processing by means of an internally stored computer program, and also provides a means of external control and manual intervention by an operator. The Storage section contains the computer program and provides a fast-access working storage for the input and output of the system. The Input-Output section provides a means of converting data from input media form to a language acceptable to the computer and from computer language to output media forms. The operational components which form these three functional sections of the computer can be assembled into various computer configurations.

The size and scope of the problem to be solved by the computing system will dictate the computer configuration. Real-time or scheduled processing of commercial, scientific, and military problems is accomplished economically on a PHILCO 2000 System. This wide range of capability demonstrates the variety and flexibility of the system components.

SYSTEM COMPONENTS

- Storage Section
  - Core Memory - Model 2200 - ten microsecond
    Model 2100 - two microsecond
  - Magnetic Drum

- Central Computer Section
  - Model 210
  - Model 211
  - Model 212

- Input-Output Section
  - Input-Output Processor
  - Magnetic Tape System
  - Paper Tape Systems
  - Accounting Clock System
  - Universal Buffer Controller
    - Punch Card System
    - Printer Systems
    - Data Link Systems
  - Real-Time System
    - Auto-Control Unit
    - Interval Timer Unit

In this manual, major system components are presented in the logical order shown on the preceding page. Each component is described on three levels:

General Information
Functional Description
Technical Details

Non-technical personnel may rapidly acquire a general picture of the PHILCO 2000 system by scanning the manual, reading only the general sections. Technical personnel will read the manual to a level dictated by their interests.

PHILCO 2000 SYSTEM PHILOSOPHY

The basic design characteristics of the PHILCO 2000 Systems include parallel asynchronous implementation of central computer operations, a sub-system approach to input-output requirements, and modularity of design and construction. This philosophy of design results in computing systems with inherent qualities of system balance, flexibility, expansibility, non-obsolescence, economy, and automatic programming capabilities.

The following paragraphs first define these design characteristics, then relate them to more obvious qualities of economy and performance.

Parallel Asynchronous Logic. An asynchronous computer is one in which each new operation begins on a signal that a previous operation has been completed. By contrast, in a synchronous computer, operations begin according to equally-timed signals from a master clock. Asynchronous design has the advantage of eliminating waiting time for the clock to catch up to completed operations. Parallel transmission means that multiple bits or characters move through the system simultaneously. By contrast, serial transmission moves information one bit or character at a time. Parallel transmission provides a time advantage. The PHILCO 2000 system is parallel and asynchronous.

Input-Output Sub-System. Computers are much faster than the peripheral equipments which serve them. Direct communication with and control of peripheral devices by the computer reduces system operational speeds to the rate of the slowest device operating. To balance these speed differentials, all processing responsibility which can be removed from PHILCO 2000 computers is delegated to intermediate devices which stand between the central computer and input-output equipments. These intermediate devices and the input-output devices form a sub-system which processes information independently and concurrently with the central computer. In this scheme, the central computer processes at its best rate and multiple input-output devices keep the computer busy.

Modular Design. There are three levels of modularity - construction, component, and system - which differ only in the size and character of the module. Printed-circuit cards are constructed as standard plug-in modules. Memory is composed of interchangeable core-storage modules. Multiple input-output devices provide system modularity.

These basic design characteristics are closely related to the important considerations of economy and operating efficiency.

System Balance. The ideal computing system is one in which all components operate at rated speed during a given processing run. This is a perfectly balanced system. A computing system's efficiency can be measured as the degree to which it conforms to this ideal arrangement. Basically, the type and volume of input-output and computing complexity are the variables which must be balanced because of processing time differentials. The demand on a computing system varies from commercial data processing with high-volume input-output and relatively little computing, through real-time processing (which by its very nature is balanced), to scientific problems with low-volume input-output and extreme computing complexity. The total modularity of a PHILCO 2000 system, its size and speed, the sub-system input-output approach, and its real-time sub-system permit assembly of a highly efficient computing system to meet any type of processing demands. The very design characteristics which allow the initial assembly of a well-balanced computing system are the factors which guarantee a flexible computing system. Flexibility is nothing more than the ability to re-balance the computing system to meet changing times, conditions, and volumes; the fact that all its components are entirely compatible makes PHILCO 2000 systems completely flexible.

The characteristics of design which allow the creation of a flexible, well-balanced computing system are also the factors which permit expansion. The implied difference between flexibility of system and the expansion of system is that the former requires different components and the latter more of the same. PHILCO 2000 systems are expandable as well as flexible.

Non-Obsolescence. Data processing equipment becomes obsolete when it can no longer be practically altered to take advantage of technological developments. Whether to improve existing equipment or to design and construct new equipment is a question of simple economics, and decisions are dictated by the basic design and construction characteristics of the existing equipment. In general, it is economically impractical to change synchronous computers which communicate directly with their input-output devices. As the characteristics of computing systems approach total modularity, sub-system input-output, and asynchronism, the possibility of improving the computer to use the results of the latest technical research increases. The expense of changing a module, component, or sub-system is minor compared to the expense of acquiring a new system whose command structure may require re-programming. The PHILCO 2000 system has all of the design characteristics which make obsolescence impossible in the foreseeable future, as it has been in the past.

Economy. The great economy of the PHILCO 2000 system lies in a design which does not permit the system to become obsolete and in the flexibility of the system. The actual cost of a computing system is its life divided into the sum of all the costs incurred during its life, including installation, system design, programming, etc. A history of the computing industry for the past ten years shows that computing systems have a predictable life expectancy, with mortality caused by lack of flexibility and obsolescence. Some pioneer computer users are now in their third cycle of site alteration, systems re-design, and re-programming costs; others are bearing the burden of high unit-cost processing on low-speed obsolete equipment. PHILCO 2000 systems avoid these pitfalls of design. Once installed and programmed, the 2000 series becomes less expensive by the year, because there are no repeated "first time" costs, and initial costs are amortized over a broader and broader base as time progresses.

MODEL 2100

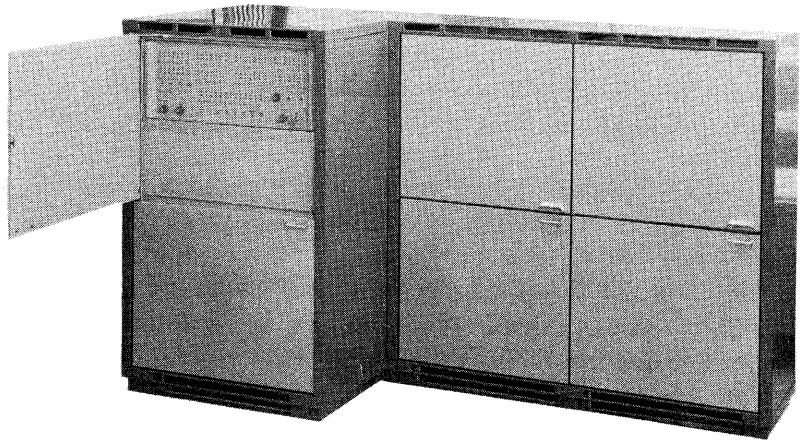
TWO-MICROSECOND CORE STORAGE

INDEX

PHILCO 2000 INFORMATION MANUAL

Model 2100 Two-Microsecond Core Storage

GENERAL INFORMATION
TWO-MICROSECOND CORE STORAGE



The Model 2100 memory is a two-microsecond, linear-selection core storage device. Originally designed as a fast replacement for the ten-microsecond memory, the Model 2100 incorporates the standard Philco features of modularity, asynchronism, and memory-sharing with an advanced and sophisticated design, resulting in an extremely fast memory.

The Model 2100 memory is the major internal storage medium for Philco computing systems which require extremely fast memory access time. Compatibility of the two-microsecond memory with the Philco 2000 family of computing systems insures that all existing programs and peripheral devices can take advantage of the two-microsecond cycle time.

The Model 2100 is available in memory sizes of 8,192 words, 16,384 words, or 32,768 words, a word consisting of forty-eight bits of information. The different memory sizes are obtained by combining basic modules of 8,192 words each. Each module has its own address and data registers, and can thus proceed independently as soon as the information is presented either to or from the system. For example, if a word is to be written into memory, the word and its associated address are transferred to the memory module and the rest of the system is released as soon as the information is received.

With this method of operation, effective memory access time can be reduced to approximately one microsecond with a computer that can efficiently utilize information at this rate.

Technically, this extremely fast effective memory access time is achieved by a priority control system of granting memory access to the various computer components and by a method of randomizing memory accesses. These features result in the overlapping of memory accesses and input-output functions which are executed by fast circuitry and logic. These asynchronous techniques demonstrate the value of the philosophy of design which enables Philco to offer a memory of this speed.

FUNCTIONAL DESCRIPTION

MODEL 2100 CORE STORAGE

MODEL 2100 CORE STORAGE
FUNCTIONAL DESCRIPTION

FUNCTIONAL CHARACTERISTICS

Word	-	48 information bits
Module	-	8,192 words
Memory sizes	-	1, 2, or 4 modules (8,192; 16,384; or 32,768 words)
Access Mode	-	Parallel
Access Time	-	2 microsecond cycle time

Each memory module is a self-contained unit with an independent Memory Data Register (MDR) and an independent Memory Address Register (MAR); it executes a read/write cycle in two microseconds. By definition, a read/write cycle is the interval of time between the acceptance of a memory access request by a core storage module and its availability to accept a subsequent memory access request.

The 13-bit Memory Address Register allows the selection of the desired word after receipt of the addressing information by the selected module. The Memory Data Register serves as an intermediate buffer for words entering or exiting the memory module.

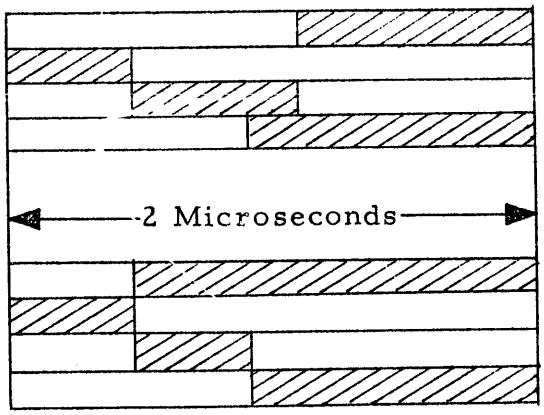
The independent nature of the core modules allows the asynchronous release of the external system at that point in the read/write cycle when the memory module has sufficient information to complete its own cycle independently. This early releasing feature allows the possibility of overlapping memory accesses in different modules and overlapping input-output functions. (A detailed description of the overlap feature is given in another section.) The following diagram shows the read/write cycle and the relative sequence of information transfers and system releases.

OUTPUT CYCLE

External System Releases
Transfer to MAR; Decode
Read
Write and Clear MAR and MDR

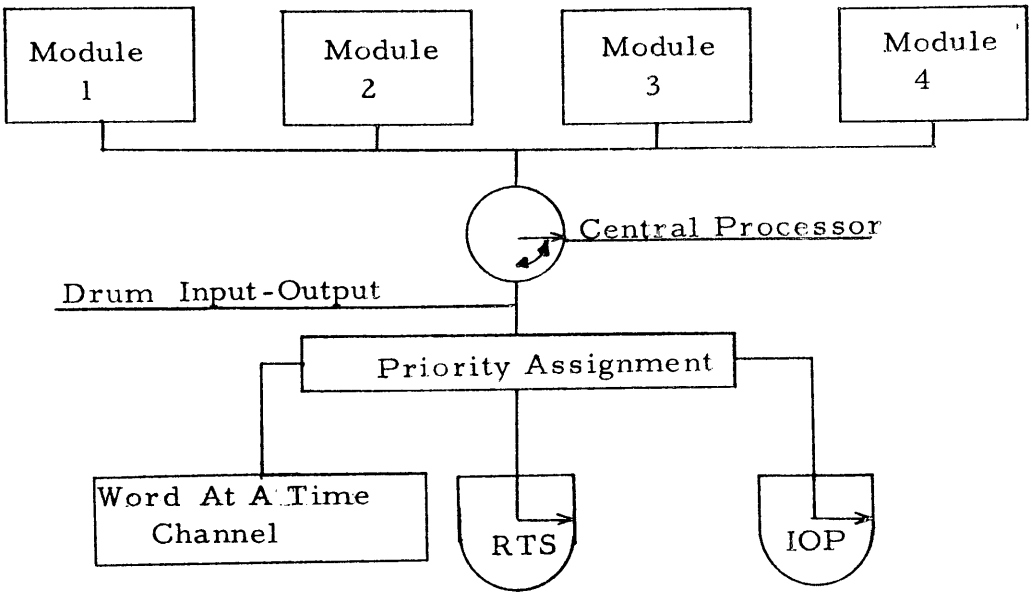
INPUT CYCLE

External System Releases
Transfer to MDR and MAR; Decode
Clear and Set Controls
Write and Clear MAR and MDR



Read/Write Cycle Diagram

Three categories of information are transferred through the memory system: Access Request and Acceptance (priority control), Address and Control Information, and Data. Actually, there are several levels of priority to determine which section or device will be given access to memory at any particular instant. The levels of memory access priority are Central Processor, Input-Output Processor, Real-Time, Word-At-A-Time Channel, and an Open channel. Descriptions of the components in the priority chain are presented elsewhere in the manual. The chain of priority control is illustrated below.



Priority Control Diagram

A computer memory consisting of one module of core storage contains 8,192 words with discrete addresses ranging from 0000 to 8191. For example, an array of memory from 0052 to 0059 is the sequential arrangement shown below.

MEMORY ADDRESS		ONE MODULE
DECIMAL	BINARY	
0052	110100	x
0053	110101	x
0054	110110	x
0055	110111	x
0056	111000	x
0057	111001	x
0058	111010	x
0059	111011	x

8,192 Word Memory

The combination of two modules of core storage to form a 16,384-word memory allows for a more sophisticated method of addressing. The fact that the low-order position of the binary memory address must always be a "0" bit or a "1" bit permits the assignments of the 8,192 odd addresses to module one and the 8,192 even addresses to module two. This assignment is made through a unit selector which uses the low order position of the memory address as a control. For example, an array of memory from 0052 to 0059 is the sequential and alternate arrangement below.

MEMORY ADDRESS		16,384-Word MEMORY	
DECIMAL	BINARY	MODULE 1	MODULE 2
0052	110100	→ x	
0053	110101		→ x
0054	110110	→ x	
0055	110111		→ x
0056	111000	→ x	
0057	111001		→ x
0058	111010	→ x	
0059	111011		→ x

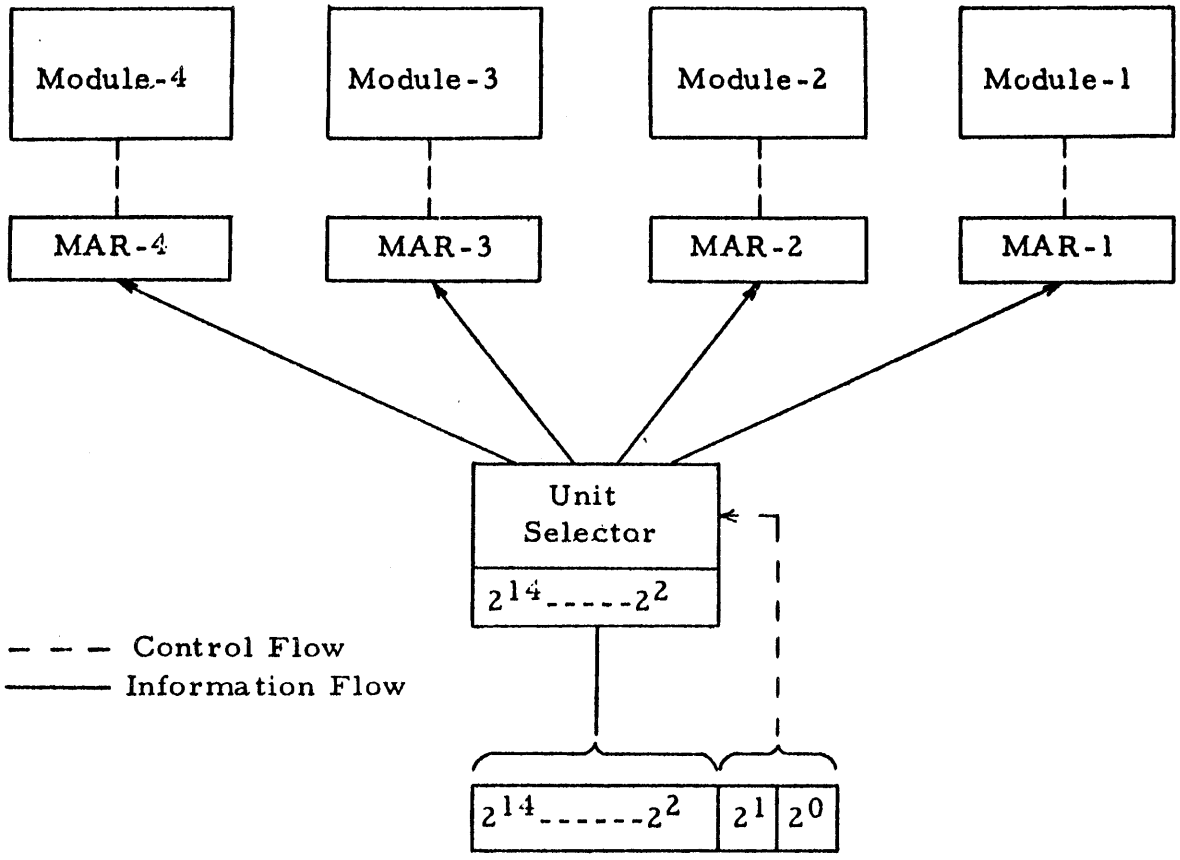
16,384-Word Memory

Four modules of core storage are combined to create a 32,768-word memory. The two low-order positions of the binary memory address contain only four possible combinations (00-01-10-11), permitting the distribution of the 32,768 words over the four modules. Distribution is made through the unit selector, which uses the two low-order positions of the memory address as a control. The assignment is sequential, with memory addresses whose low-order positions are 00 going to module one, 01 going to module two, 10 going to module three, and 11 going to module four. For example, memory address 0052 to 0059 would be assigned to modules as follows:

MEMORY ADDRESS		32,768-Word Memory			
DECIMAL	BINARY	Module 1	Module 2	Module 3	Module 4
0052	110100	x			
0053	110101		x		
0054	110110			x	
0055	110111				x
0056	111000	x			
0057	111001		x		
0058	111010			x	
0059	111011				x

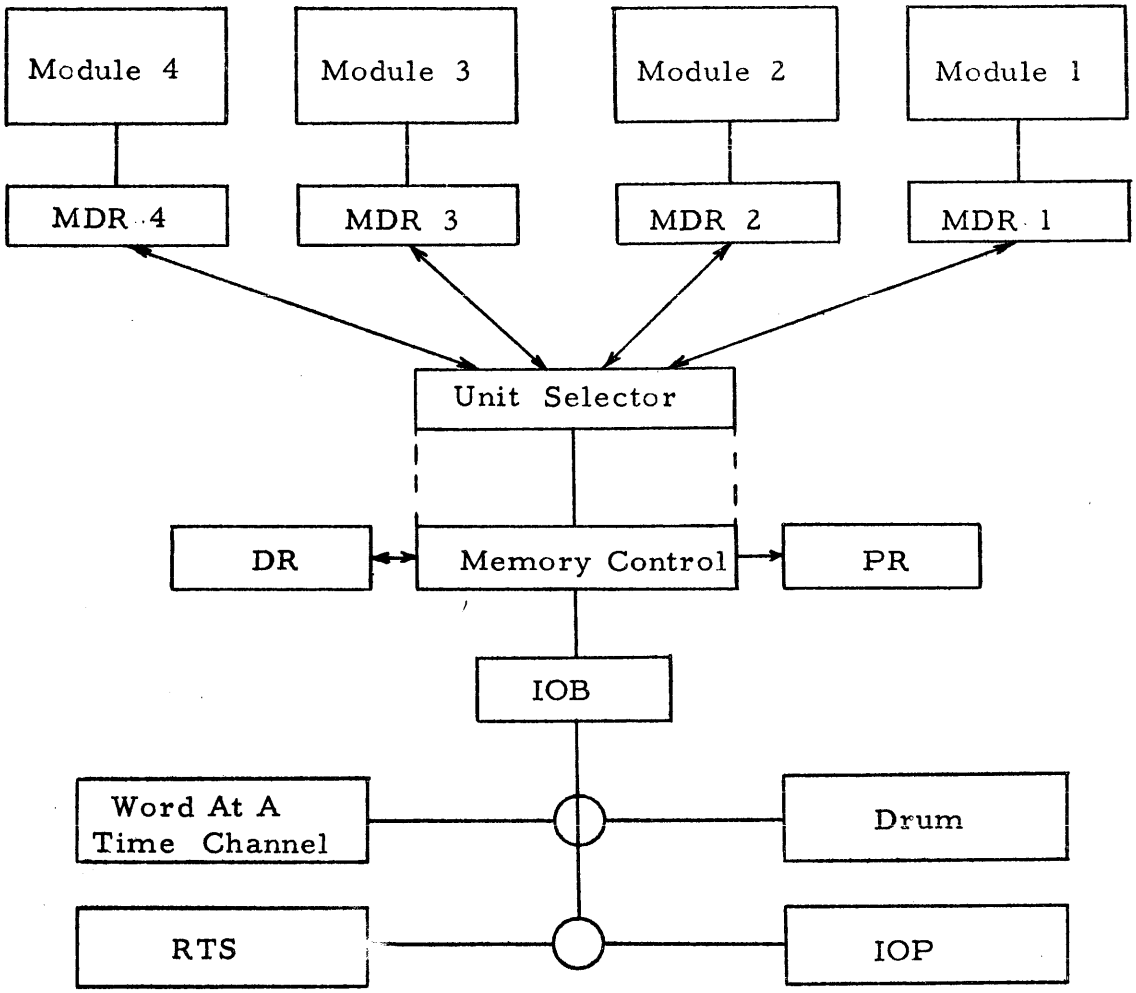
32,768-Word Memory

It is a function of the system configuration to decode the low-order address positions. Thus, as the system configuration changes, the physical array of addresses and decoding ability will change as shown below. This memory addressing technique automatically randomizes memory accesses and sets the conditions which allow the overlap both of memory accesses and of memory access and input-output operations.



Address and Control Flow Diagram

Memory accesses which refer to different units can be overlapped, so that a second access may be initiated while a first is being completed. Since sequence of memory addresses is between units, automatic randomizing provides optimum use of the overlap feature without requiring special programming.



Data Flow Diagram

The overlap feature is also valuable in the case of input-output operations. A normal input-output cycle (time from granting request to completion of write operation) is between three and six microseconds. Of this time, the memory is actually tied up for less than two microseconds. During the remainder of the input-output cycle the central processor requests memory access during the time when the input-output cycle is using memory. The central processor takes advantage of the overlap if it is referencing a memory unit other than the one the input-output is using.

TECHNICAL DETAILS

MODEL 2100 CORE STORAGE

MODEL 2100 CORE STORAGE

TECHNICAL DETAILS

PHYSICAL CHARACTERISTICS

	Core Unit	Control Unit
Height	57-1/2"	57"
Width	97"	32-1/2"
Depth	40"	18-1/2"
Weight	2284 lbs.	681 lbs.

LINEAR SELECTION

The operational speed of a coincident-current memory is governed primarily by the sum of the read and write core-switching times. These times, in turn, are determined solely by the core characteristics at a current equal to the sum of two half-select currents.

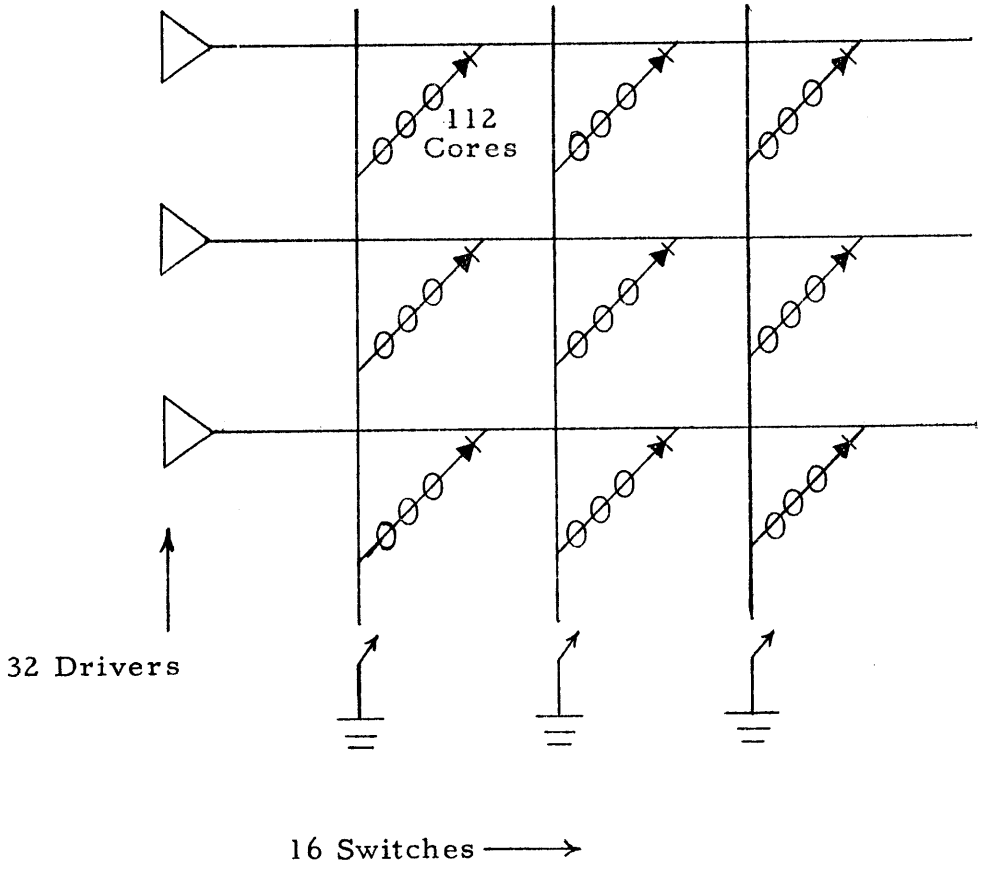
In a linear selection memory, on the other hand, the magnetic cores, in addition to performing the storage function, are used to perform the logical AND function for selection only during the write phase. Once this mode of operation is obtained, larger currents can be supplied to the individual memory cores during the read phase with a corresponding decrease in core switching time and a subsequent decrease in memory cycle time. The decrease in time is directly related to the magnitude of the read current.

In the Model 2100, all the cores storing the individual bits of a given word are placed along a single pair of address wires and the word is accessed by selecting the switches placed at the ends of the address wires. In addition to the address lines, a digit winding at right angles to the address wires passes through corresponding bits of each word.

During the read phase only the read address current flows and the core output voltages appear at each of the terminals of the sense windings. During the write phase, currents may flow in both the write address line and the digit winding. Note that the memory elements are not used for selection during read.

The address current during the read operation is approximately three times the half-select current. This current is larger than a corresponding full-select current used in coincident-current operation, thus considerably reducing the core switching time.

During the write portion of the cycle, a partial-select current is applied to the address line but the magnetizing force applied to the cores is now in a direction opposite to that which occurred during read. The current alone is not of sufficient amplitude to switch the memory cores and the cores remain in the "0" state. However, if a partial-select current is also applied to the digit winding, the total current at a selected core will be greater than the full-select value, thus switching the core to a "1". The maximum amplitude of the digit current is such that no switching occurs in any of the other cores on a digit winding.



Functional Schematic of Core Storage Read Array

Philco 2000
Information Manual

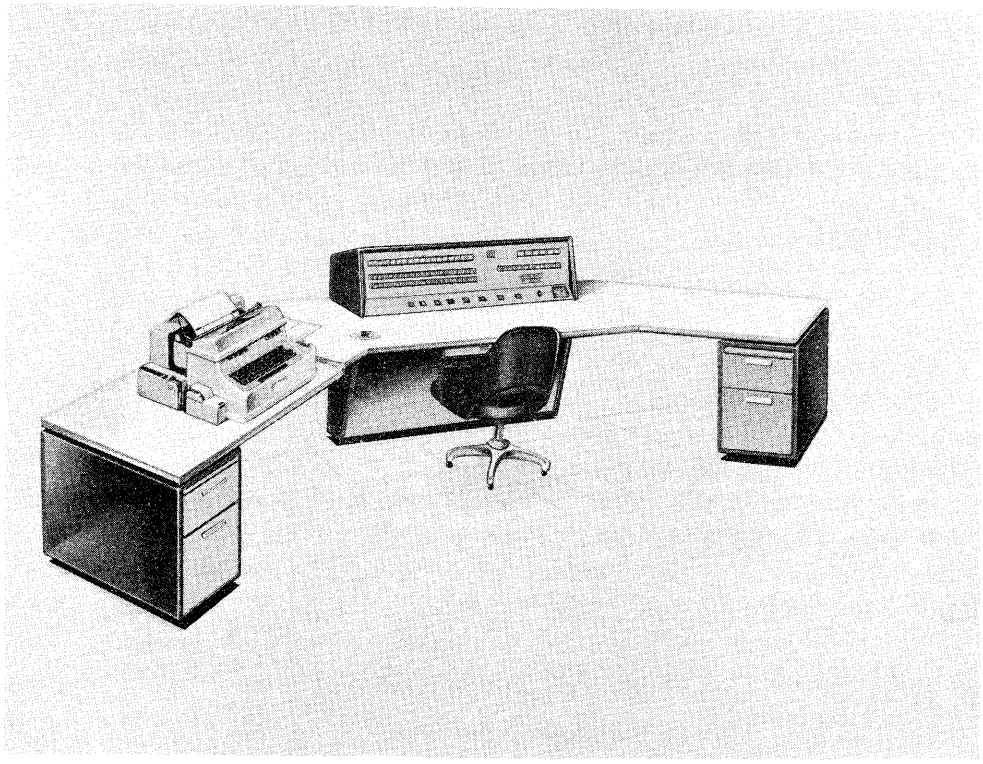
MODEL 212 COMPUTER

INDEX

PHILCO 2000 INFORMATION MANUAL

Model 212 Computer

GENERAL INFORMATION
MODEL 212 COMPUTER

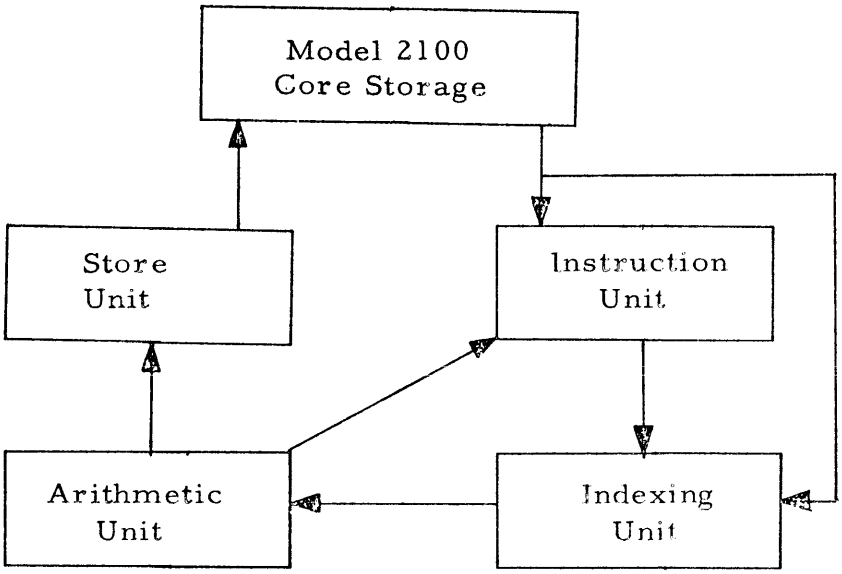


The Model 212 Computer is the newest addition to the PHILCO 2000 series. It has been designed to increase the capabilities of the PHILCO 2000 Electronic Data Processing System while remaining compatible with its predecessors and utilizing to the full the high-speed characteristics of the Model 2100 Core Storage. The Model 212 brings to the PHILCO 2000 series a degree of computer/memory balance never before achieved by the industry. Since the Model 212 computer was designed after the Model 2100 two-microsecond memory, it was possible to tailor the computer to the memory and so take full advantage of the characteristics of the storage design. New instructions and control concepts have been implemented which further amplify the versatility of the computer.

The Model 212 computer is a natural outgrowth of the PHILCO 2000 systems which have long been solving a wide range of problems in customer installations. From its experience with these installations Philco determined where improvements in computer operation would most readily benefit the user. For example, a survey of instruction frequency showed that the number of occurrences of a small group of instructions was significantly higher than the average. This was obviously one area in which even the slightest increase in speed would result in major improvements in overall efficiency. All possibilities for improving PHILCO 2000 operation were then investigated and evaluated. Those which proved economically sound were incorporated into the design. The end goal was to provide a maximum computation-per-dollar ratio by increasing speed and improving performance.

The computers in the PHILCO 2000 series have many common characteristics. A forty-eight bit word, providing two instructions per word, is used throughout the series. A set of 225 instructions allows ease of programming with all the computers. In the Model 212, this set has been expanded to 250 for even more program flexibility. Automatic counting and indexing are provided by eight index registers.

The Model 212 is four to eight times faster than its predecessors. This speed-up, achieved by the implementation of faster circuitry and a unique organization, is illustrated by the 0.26-microsecond average add cycle time of the Model 212.



Much of the speed of the Model 212 is due to its logical organization. Advanced four-way processing is possible because of a logical partitioning of the central computer into four interdependent units. These units asynchronously process as much work as is presented to them, ensuring that operations are performed without unnecessary delays. The use of memory is improved because the computer itself is capable of processing three accesses within a two-microsecond period.

The Model 212 computer is one more step in Philco's overall plan to supply the most advanced customized computer systems to discerning users. Its speed and versatility are matched to the PHILCO 2000 system in a way which permits indefinite future expansion. Advances in the state of the art are immediately translated into new hardware which can be added to PHILCO 2000 systems to provide even more economy and efficiency.

Philco 2000
Information Manual

FUNCTIONAL DESCRIPTION

MODEL 212 COMPUTER

FUNCTIONAL DESCRIPTION

MODEL 212 COMPUTER

GENERAL

The Model 212 parallel binary computer is a direct extension of the PHILCO 2000 series. It is compatible with its forerunners, yet the logic has been implemented in a manner which allows improved design and additional functions.

COMPATIBILITY

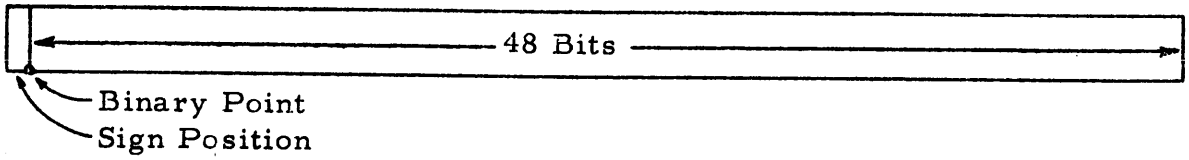
The Model 212 Central Processor has ascending compatibility with the Models 210 and 211. Any program which is used by a Model 210 or 211 with a maximum configuration of eight index registers, floating point option, 32,768 words of memory, and a 16 x 4 Input-Output Processor can be run on the Model 212.

The Model 212 has instructions and features not available in the Models 210 and 211 but these additional capabilities do not affect the operation of present programs. New programs written for the 212 which make use of these features cannot be used on the Model 210 or 211 central processor.

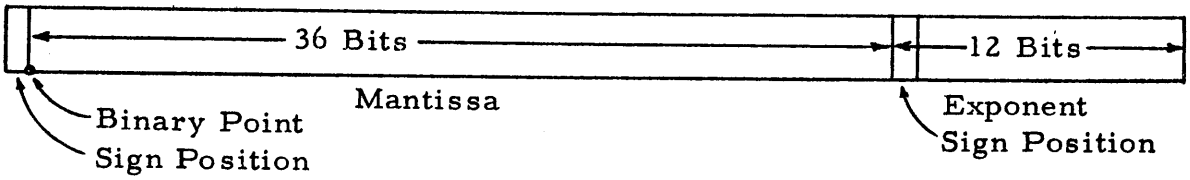
The Model 212 can work with either a 16,384 or 32,768 word Model 2100 two-microsecond core storage. Within the processor, the 48 information bits in each word are used in many formats, some examples of which are shown below.

WORD FORMATS

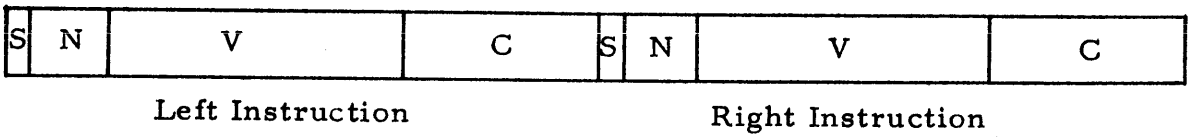
Fixed Point Data Word



Floating-Point Data Word

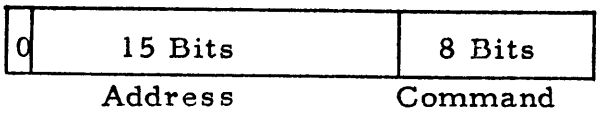


Instruction Word

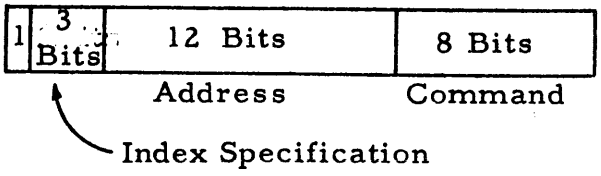


INSTRUCTION FORMATS

Unindexed Instruction



Indexed Instruction



LOGICAL ORGANIZATION

The Model 212 computer is more than four times as fast as the Model 211, its immediate predecessor. Much of this speed is derived from the implementation of a unique logical organization.

Advanced four-way processing is possible because the computer is logically partitioned into four interdependent units. The asynchronism of these units permits operations to proceed without unnecessary delays. The use of the memory is improved because, rather than handling a single memory request at a time, the computer is so organized that three of its units may be processing memory requests while a fourth is doing arithmetic. Input-output operations proceed concurrently with this processing. A simplified block diagram of the Model 212 Computer is given on the following page.

The Instruction Unit supplies instructions to the rest of the computer for processing. This unit contains the Program Address Register and the Program Register, which allow a total of four instructions to be held in the unit. The Indexing Unit performs that part of the instruction which can be done prior to algorithm execution. Its major function is to obtain operands and have them ready when the Arithmetic Unit needs them; this function involves determining the effective address of the operand. The Arithmetic Unit receives instructions from the Indexing Unit, executes them, and transfers results to the Store Unit, which writes the results into memory.

FOUR-WAY PROCESSING

The units described above combine to form a computer organization which can process stages of several instructions at a time and, even more important, permit continuous operation of the Arithmetic Unit. Information necessary to that unit is prepared in anticipation of the need, and all results of operations are released immediately upon their availability. Under these conditions, the effective execution time of an instruction can be reduced to that time which is allocated to the instruction by the Arithmetic Unit.

THE INSTRUCTION UNIT

The flow of information through this unit is such that when a memory request is initiated from the Program Address Register, a pair of instructions is read out of memory into the Program Buffer (PR*). When a signal is received from the Program Register that both of the instructions have been presented to the Indexing Unit for

processing, the contents of the Program Buffer are transferred to the Program Register, and the contents of the Program Address Register are transferred to the Address Buffer (PA*). At this point, the address in PA* is incremented and transferred to the Program Address Register while remaining in PA* in its original form. Another memory request is then initiated and the cycle is continued. Controls in the Instruction Unit sequentially select the left-half or right-half of the word in the Program Register for processing. Similar controls associated with the Program Buffer allow the repeat mode to loop on the four instructions contained in the Instruction Unit.

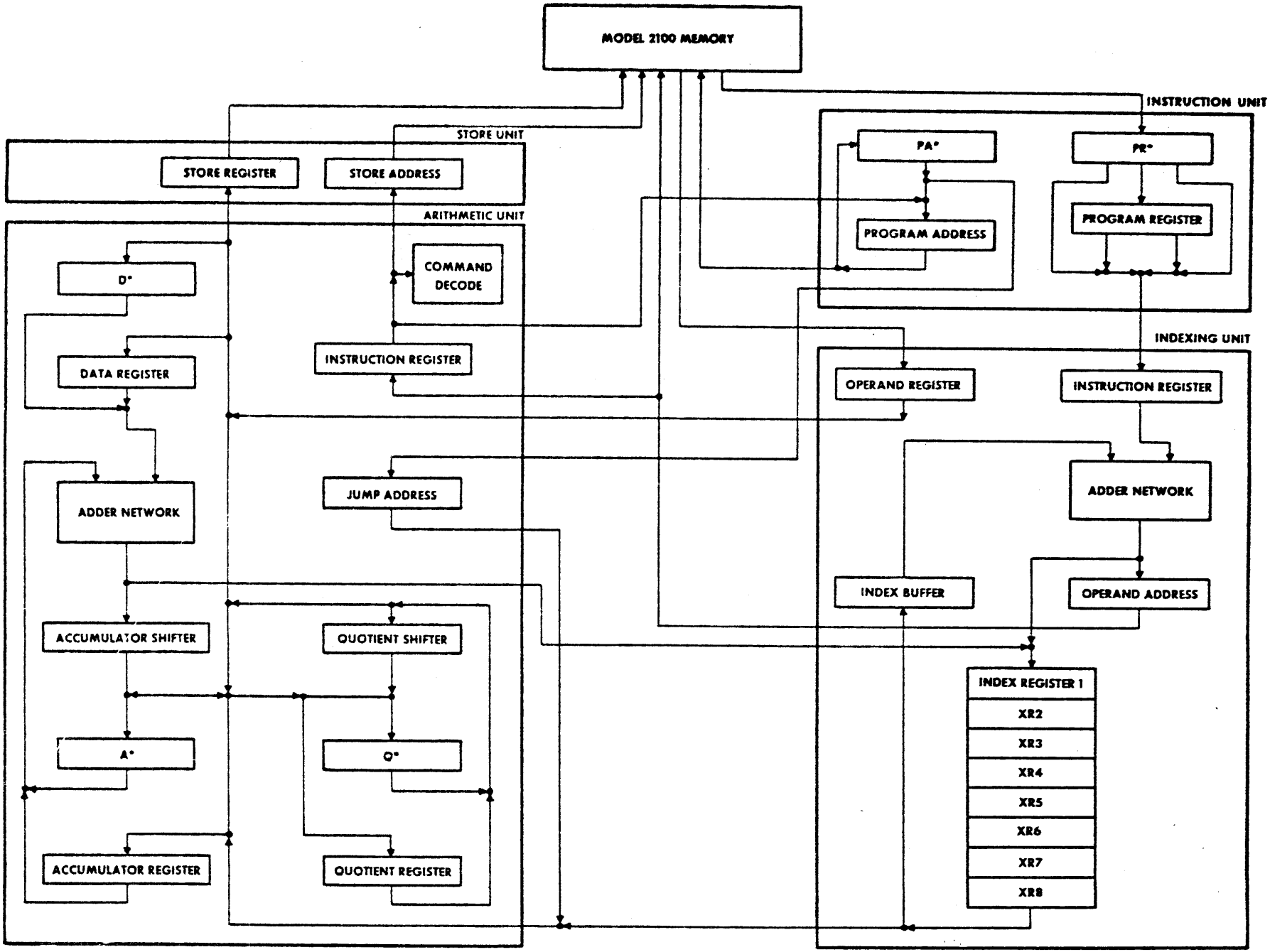
Use of the Instruction Unit virtually eliminates instruction access time, because whenever a new instruction is needed it is ready and waiting for processing.

THE INDEXING UNIT

The Indexing Unit pre-processes instructions in anticipation of the needs of the Arithmetic Unit, determining the effective address of the operand when required. When an instruction is loaded into the Indexing Unit Instruction Register, it is classified to determine whether an effective address is necessary and, if so, how that address will be used. If it is determined that an index register is to be used, the contents of that index register are transferred to an Index Register Buffer and added to (or subtracted from) the instruction address. The effective address of the instruction thus created is placed in the Operand Address Register. If an operand is required from memory, a request is initiated and the operand is read into the Operand Register where it is held until needed by the Arithmetic Unit. Thus, in the Model 212 computer, the instruction is readied for execution during a period when the Arithmetic Unit is working on a previous instruction. In some cases, such as with the repeat instruction, the instruction is completed in the Indexing Unit and not even sent to the Arithmetic Unit. The independent memory accessing and anticipation of the Indexing Unit make it a most important link in the 212 organization.

THE ARITHMETIC UNIT

The above paragraphs describe how the Instruction and Indexing Units of the Model 212 computer prepare information for processing. The asynchronous nature of the Arithmetic Unit permits it to begin working on this information as soon as it has completed previous tasks. Each time the Arithmetic Unit completes an operation it signals the Indexing Unit that is ready to begin a new one. Since the Indexing Unit anticipates this request, continuous operation of the Arithmetic Unit is possible.



Simplified Block Diagram, Model 212 Computer

There are three major addressable registers in the PHILCO 2000 computers: the Accumulator (A), the Quotient register (Q), and the Data register (D). In the Model 212, all three of these registers are duplicated in an unusual manner. At any particular instant only the computer knows which of the pair of registers is actually the one being addressed. For instance, if a number in the Accumulator is to be shifted, it is sent through the shifter to the other Accumulator and the computer remembers which register is being used as the true Accumulator. This anonymity eliminates redundant transfers of information.

The Arithmetic Unit has its own Instruction Register, from which the command is decoded to set up controls for the instruction. It also includes control hardware such as the Jump Address Register, which keeps track of the flow of the program by recording the locations from which jumps take place.

Rather than perform a series of single-bit shifts, the 212 shifts information more than one position at a time where necessary. Combinations of four positions right, two right, one right, two left, one left, and zero shifts satisfy the requirements of any shift orders called for. The 212 can therefore shift right one, two, or four places in one cycle, three, five, six, or eight places in two cycles, and so forth.

THE STORE UNIT

Any computer must be able to preserve the results of its operations. In the Model 212, when the results of an operation in the Arithmetic Unit are to be stored in memory the information and its address are transferred to the Store Unit. The Arithmetic Unit is then free to go on to another instruction while the Store Unit takes over and requests a memory access. Thus, the Store Unit takes charge of writing all information from the computer into memory, and also generates the parity bits which are transferred to the memory with the information.

MONITORING FEATURES

In a computer composed of semi-independent units as described above, some means must be provided to coordinate the speeds of these units. The 212 has a complete address monitoring scheme which prevents its units from doing redundant or unnecessary work which would have to be "undone" later.

As each instruction is classified in the Indexing Unit and an effective address generated, it is determined whether the possibility of conflict with another unit exists. If the possibility does exist, a check is made to determine if there is a conflict. If so, a sequence of operation is established to avoid interaction. For instance, if an instruction in the Arithmetic Unit is modifying the operand to be used by the instruction in the Indexing Unit, the Indexing Unit access is delayed until the memory operand has been changed. This interaction holds true in all cases where this type of conflict exists.

Another example of possible conflict occurs when a conditional jump instruction appears in the Arithmetic Unit. Since the Indexing Unit is processing the following instruction, which may or may not be executed, the Indexing Unit delays the modification of any index register until it has been determined that the jump will not take place. In this way it is possible to avoid reconstruction of registers if the jump is accepted. No time is lost, because the modification is overlapped.

These examples show that even when the 212 is processing up to seven instructions at a given time, a fail-safe technique of handling conflicting functions prevents undesired results and unnecessary time loss due to unit interaction.

INDEX REGISTERS

The Model 212 Computer uses eight index registers, which perform all the functions of the Model 211 index registers as well as the additional functions detailed below.

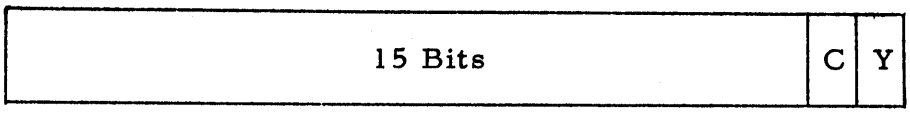
Model 212 index registers contain a maximum of 17 bits, consisting of 15 data bits, a C-bit, and a Y-bit. The C-bit and the Y-bit are control bits, and provide the following four index register options with instructions which are index register modified. (A complete description of these instructions is given in a later section.)

C = 0, Y = 0 — The effective operand address is the sum of the V field of the instruction and the contents of the index register. The contents of the index register is not modified. The TCXZ instruction sets the bits to perform this option.

C = 1, Y = 0 — The effective operand address is the sum of the V field of the instruction and the contents of the index register. After execution, the contents of the index register is incremented by one. The TCXS instruction sets the bits to perform this option.

C = 0, Y = 1 — The effective operand address is the contents of the index register. After execution, the contents of the index register is incremented by the contents of the V field of the instruction. The TYXZ sets the bits to perform this option.

C = 1, Y = 1 — The effective operand address is the contents of the index register. After execution, the contents of the index register is decremented by the contents of the V field of the instruction. The TYXS instruction sets the bits to perform this option.

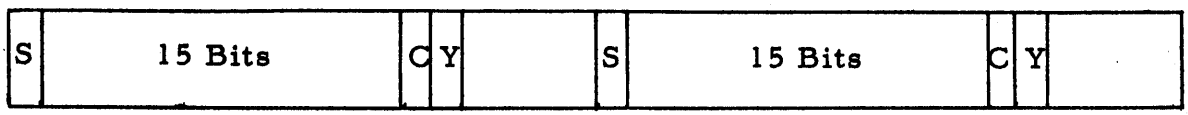


C	Y	
0	0	(X) UNALTERED
1	0	(X) + 1 → X
0	1	(X) + (V) → X
1	1	(X) - (V) → X

Index Register Data Format

When the C-and Y-bits of an index register are transferred to the D Register, they occupy the two high-order positions of the COMMAND portion of the designated half of the D Register. For example, if the contents of the index register are transferred to the left half of the D Register, the data bits occupy bit positions 1 through 15, the C-bit occupies bit position 16, and the Y-bit occupies bit position 17.

If the contents of the index register are transferred to the right half of the D Register, the data bits occupy bit positions 25 through 39, the C-bit occupies bit position 40, and the Y-bit occupies bit position 41 (see below).



Storing of Contents of Index Registers in the D Register

INSTRUCTION CATALOG

The following additions to the instruction catalog standard to all PHILCO 2000 computers have been made for the Model 212 computer.

REPEAT INSTRUCTIONS

Repeat instructions allow one, two, three, or four instructions to be repeated. If the single repeat (RPT) instruction is a left instruction, the next instruction is repeated. If it is a right instruction, the next pair of instructions is repeated. The instructions can be repeated up to 4095 times, as specified by the V field of the RPT instruction. If the double-repeat (DRPT) instruction is a left instruction, the next three instructions are repeated. If it is a right instruction, the next four instructions are repeated. The instructions can be repeated up to 255 times, as specified in the V field of the DRPT instruction.

The high-order bits of these instructions are used to specify the type of index register modification desired. The bits are divided into pairs, with each successive pair of bits applicable to each successive repeated instruction. If either one or three instructions are to be repeated, the first pair of bits is not used.

The formats for both the single and double repeat instructions are shown below. The modifier bits (M) have the following significance:

M = 0 (N) - Normal index register modification using C and Y is specified.

M = 10 (A) - Actual settings of C and Y are ignored. The instruction is performed as if C = 0 and Y = 1.

M = 11 (S) - Actual settings of C and Y are ignored. The instruction is performed as if C = 1 and Y = 1.

Repeat Instruction Format*

One Instruction Repeat -
Mnemonic: L RPTM COUNT

Not Used	M	Count	RPT

M is the modifier specification for the repeated instruction.

Two Instruction Repeat -
Mnemonic: R RPTMM COUNT

M_L	M_R	Count	RPT

M_L refers to the first instruction being repeated.
M_R refers to the second instruction being repeated.

Three Instruction Repeat -
Mnemonic: L DRPTMMM COUNT

Not Used	M_{R1}	M_L	M_{R2}	Count	DRPT

M_{R1} refers to the first instruction being repeated.
M_L refers to the second instruction being repeated.
M_{R2} refers to the third instruction being repeated.

Four Instruction Repeat -
Mnemonic: R DRPTMMMM COUNT

M_{L1}	M_{R1}	M_{L2}	M_{R2}	Count	DRPT

M_{L1} refers to the first instruction being repeated.
M_{R1} refers to the second instruction being repeated.
M_{L1} refers to the third instruction being repeated.
M_{R2} refers to the fourth instruction being repeated.

*L or R specifies whether the instruction is located in the left or right half of the program word.

INDEX REGISTER INSTRUCTIONS

The 212 index register instructions expand the versatility of the index registers and provide the means for manipulating the Y-bits. In order to facilitate use of the Y-bit, the following instructions have the additional function of clearing the Y-bit to zero: TCXZ; TCXS; TCXSC; TDXL, R; TDXLC; TDXRC; TIXZ; TIXS.

The instructions applicable to index register comparison cause the contents of an index register to be incremented or decremented by the contents of the V field. The new contents of the index register are compared with the left-half address field of the D Register and, if the required conditions are met, a jump is effected. These instructions are explained below.

COMMAND	ACTION
AXJL	a. Increments the contents of the index register specified by the N field of the instruction with the contents of the V field of the instruction. b. Jumps to the memory location specified by the right address field of the D Register if the contents of the incremented index register are less than the contents of the left address field of the D Register.
AXJG	Same as AXJL except that the jump is made if the contents of the incremented index register are equal to or greater than the contents of the address portion of the D Register.
SXJL	a. Decrements the contents of the index register specified by the N field of the instruction with the contents of the V field of the instruction. b. Jumps to the memory location specified by the right address field of the D Register if the contents of the decremented index register are less than or equal to the contents of the left address field of the D Register.

COMMAND	ACTION
SXJG	Same as SXJL except that the jump is made if the contents of the decremented index register are greater than the contents of the left address field of the D Register.
TXYDL	Transfers the contents of the index register specified by the address portion of the instruction to the left address portion of the D Register. The C-bit is transferred to D_{16} and the Y-bit is transferred to D_{17}.
TXYDR	Transfers the contents of the index register specified by the address portion of the instruction to the right address portion of the D Register. The C-bit is transferred to D_{40} and the Y-bit is transferred to D_{41}.
TDXYL	Transfers the contents of the left address portion of the D Register to the index register specified by the address portion of the instruction. D_{16} is transferred to the C-bit and D_{17} is transferred to the Y-bit
TDXYR	Transfers the contents of the right address portion of the D Register to the index register specified by the address portion of the instruction. D_{40} is transferred to the C-bit and D_{41} is transferred to the Y-bit.

JUMP INSTRUCTIONS

Besides the jump instructions explained in the previous section, two other jumps have been added. These are unconditional jumps (left and right) which do not affect the contents of the JA Register and are designated JL and JR.

PHILCO 2000 COMMAND CODE CHART - MODEL 212

Quaternary

QUATERNARY	-00	-01	-02	-03	-10	-11	-12	-13	-20	-21	-22	-23	-30	-31	-32	-33	BINARY
00-	HLTL	JBTL	ICOZ	NOPL	TIO	TCM	SKC	TCXZ	TJML	INCAL	TIJL	RPT	ETD	DORMS	EI	LWD	0000-
01-	CM	TMA	TMQ	TMD	TAM	CA	TAQ	TAD	TQM	TQA	CQ	TQD	TDM	TDA	TDQ	CD	0001-
02-	JMPL	JAZL	JNOL	JOFL	JAPL	JANL	JAEQL	JAEDL	JQPL	JQNL	JQEL	JQOL	JDPL	JAGQFL	JAGQL	JAGDL	0010-
03-	TDXL	TDXLC	TXDL	TXDLC	ADXL	SDXL	AXJG	SXJG	JL	TIXZ	TDXYL	TXYDL	AIXJ	SIXJ	AIXOL	SIXOL	0011-
10-	AM	AMS	CAM	CAMS	AMA	AMAS	CAMA	CAMAS	AQ	AQS	CAQ	CAQS	AQA	AQAS	CAQA	CAQAS	0100-
11-	SM	SMS	CSM	CSMS	SMA	SMAS	CSMA	CSMAS	SQ	SQS	CSQ	CSQS	SQA	SQAS	CSQA	CSQAS	0101-
12-	MM	MMS	MMR	MMRS	MMA	MMAS	MMAR	MMARS	MA	MAS	MAR	MARS	MAA	MAAS	MAAR	MAARS	0110-
13-	DAQ	DAQS	DA	DAS	*	*	DRPT	RPT	MAD	MSU	EA	ES	AD	SD	TYXZ	TYXS	0111-
20-	HLTR	JBTR	ICOS	NOPR	TTD	TDC	SKF	TCXS	TJMR	INCAR	TIJR	RPT	ETA	AWCS	EIS	SWD	1000-
21-	SLAQ	SRAQ	SLAQN	SRAQN	SLA	SRA	SLAN	SRAN	SLQ	SRQ	SLQN	SRQN	SCD	SRD	SCD	SRDN	1001-
22-	JMPR	JAZR	JNOR	JOFR	JAPR	JANR	JAEQR	JAEDR	JQPR	JQNR	JQER	JQOR	JDPR	JAGQFR	JAGQR	JAGDR	1010-
23-	TDXR	TDXRC	TXDR	TXDRC	ADXR	SDXR	AXJL	SXJL	JR	TIXS	TDXYR	TXYDR	AIXJ	SIXJ	AIXOR	SIXOR	1011-
30-	FAM	FAMS	FCAM	FCAMS	FAMA	FAMAS	FCAMA	FCAMAS	FAQ	FAQS	FCAQ	FCAQS	FAQA	FAQAS	FCAQA	FCAQAS	1100-
31-	FSM	FSMS	FCSM	FCSMS	FSMA	FSMAS	FCSMA	FCSMAS	FSQ	FSQS	FCSQ	FCSQS	FSQA	FSQAS	FCSQA	FCSQAS	1101-
32-	FMM	FMMS	FMMR	FMMRS	FMMA	FMMAS	FMMAR	FMMARS	FMA	FMAS	FMAR	FMARS	FMAA	FMAAS	FMAAR	FMAARS	1110-
33-	FDAQ	FDAQS	FDA	FDAS	*	*	*	*	FMAD	FMSU	FEA	FES	FAD	FSD	*	*	1111-
	-0000	-0001	-0010	-0011	-0100	-0101	-0110	-0111	-1000	-1001	-1010	-1011	-1100	-1101	-1110	-1111	

Binary

*Command Faults

PHILCO 2000 INSTRUCTIONS

Instruction Class	Instruction	Mnemonic Code: Operation	Mnemonic Code: Register or Condition	Mnemonic Code: Option	Description of Operation	Notes	Code Example
Addition	Add	A	M Q	A, S	1. (A) + Operand ⟶ A 2. When result is stored: (A) ⟶ M (and D) The operand is from M or Q and may be in absolute value. Before step 1, A is cleared to zero for Clear Add.	Options: A = absolute operand S = result stored F = floating point Overflow: The overflow indicator is set when the result ≥ 1 or < -1.	AM AMA CAQS CAQAS
	Clear Add	CA					
	Add D	AD			(A) + (D) ⟶ A		
Subtraction	Subtract	S	M Q	A, S	1. (A) - Operand ⟶ A 2. When result stored: (A) ⟶ M (and D) The operand is from M or Q and may be in absolute value. Before step 1, A is cleared to zero for Clear Subtract.	Options: A = absolute operand S = result stored Overflow: The overflow indicator is set when the result ≥1 or < -1.	SM SMS SQA CSQAS
	Clear Subtract	CS					
	Subtract D	SD			(A) - (D) ⟶ A		
Multiplication	Multiply	M	A M	A R S	1. Operand x (Q) ⟶ A, Q or A rounded* 2. When result stored: (A) ⟶ M (and D) The operand is from M or A and may be in absolute value.	Options: A = absolute operand R = rounded product S = result stored Overflow: when the result = 1 * (Q) are unaltered when rounded.	MAR MMRS
	Multiply and Add	MAD			[(M) x (Q)] + (A) ⟶ A	The product is rounded. (Q) are unaltered.	
	Multiply and Subtract	MSU			[(M) x (Q)] - (A) ⟶ A		
Division	Divide A register	DA		S	1. [(A) or (A, Q)] ÷ (M) ⟶ Q, remainder ⟶ A 2. When result stored: (Q) ⟶ M (and D)	Option: S = result stored Potential overflow is detected if |M| < (A), ∝ |M| = |A| and (A) positive.	DA DAS
	Divide A and Q registers	DAQ					DAQ DAQS
Transfer	Clear	C	M, A, Q, D		0 ⟶ M or A or Q or D		CM
	Transfer	T	M, A, Q, D	M, A, Q, D*	* Transfer [(M) or (A) or (Q) or (D)] to [M or A or Q or D]	*These are not optional. One letter must be selected. TMM, TAA, TQQ and TDD are not permitted.	TMA
Shift	Shift Left	SL	A AQ Q D	N	Shift the contents of the register (s) the number of places specified by the address. A numerical shift will preserve the sign of a word.	Option: N = numerical shift. No option specifies ordinary shift. (D) may only be shifted right	SLA SRQN
	Shift Right	SR					
	Shift Circular (D)	SCD			Shift (D) in circular mode right		
Jump	Jump	JMP			Unconditional Jump	1. Address of next instruction ⟶ JA 2. Effective address ⟶ PA *Shift (Q) in circular mode left (for P or N) or right (for O or E) 1 position regardless of conditions. In these cases positive is defined as sign bit = 0; negative is sign bit = 1. ‡See notes for NOP and TJM. †JAGD treats words as alpha-numeric. For A and Q Comparisions, (Q) ⟶ D. Then (A) are compared to (D). In JAGQF the numbers should be normalized.	
	Breakpoint Jump	JBT			Stop if breakpoint switch set, jump if not.		
	Jump if overflow	JOF			Jump if overflow indicator is set.		
	Jump if no overflow	JNO			Jump if overflow indicator is not set.		
	Jump if (D) are Positive	JDP		L‡ or R‡	Jump if (D) are positive.		
	Jump if (A) are +, -, 0	JA	P, N, Z		Jump if (A) are positive or negative, or zero.		JAP
	*Jump if (Q) are +, -, odd, even	JQ	P, N, O, E		Jump if (Q) are positive or negative, or odd or even.		JQE
	Jump if (A) Equal (D) or (Q)	JAE	D, Q		Jump if (A) equal (D) or (Q).		JAED
	†Jump if (A) are Greater than or equal to (D) or (Q) or (Q)-floating point	JAG	D, Q, QF		Jump if (A) are greater than or equal to (D), or (Q), or (Q) if (A) and (Q) are floating point numbers.		JAGQ

PHILCO 2000 INSTRUCTIONS (cont'd)

Instruction Class	Instruction	Mnemonic Code: Operation	Operation or Half Word	Option	Description of Operation	Notes
Index Register	Transfer Counter to Index	TCX	S, Z, SC		$1 \rightarrow X_C$ if S; $0 \rightarrow X_C$ if Z; $1 \rightarrow X_C$, $(X)+1 \rightarrow X$ if SC	Option: C = Counter indicator is transferred. L and R specify left or right half of D register.
	Transfer Instruction address to Index	TIX			$I_V \rightarrow X$; $1 \rightarrow X_C$ if S, $0 \rightarrow X_C$ if Z.	
	Transfer from D to Index	TDX	L, R	C	D address $\rightarrow$ X	
	Transfer from Index to D	TXD			(X) $\rightarrow$ D address	
	Add (D) to Index	ADX	L, R		(X) + D address $\rightarrow$ X	
	Subtract (D) from Index	SDX			(X) - D address $\rightarrow$ X	
	Add Instruction Address to Index and Jump	AIXJ			$(X) + I_V \rightarrow X$ } Jump to D right address if	
	Subtract Instruction Address from Index and Jump	SIXJ			$(X) - I_V \rightarrow X$ } $(X) \neq$ D left address	
	Add Instruction Address to Index and set Overflow	AIXO	L, R		$(X) + I_V \rightarrow X$ } Set Overflow if (X) = D address	
	Subtract Instruction address from Index and set Overflow	SIXO			$(X) - I_V \rightarrow X$ }	
	Repeat					*N, A, and S are not optional and specify no modification, add to and subtract from the index register(s) specified by the repeated instruction(s).
	Repeat	RPT	N, A, S	N, A, S*	The next instruction or instruction pair is repeated the number of times specified by the address part of the RPT. If RPT is left half instruction, next instruction is repeated; if RPT is right half instruction, next pair of instructions is repeated.	
Extract	Extract from memory and:				Extract: bit by bit logical multiply (M) . (Q) $\rightarrow$ D; or mask (M) $\rightarrow$ D according to (Q)	Floating point mode is possible with EA and ES but is only in effect after the extraction.
	Transfer to D	ETD			1. e.g. M · Q $\rightarrow$ D: 1·0=0; 0·0=0; 1·1=1; 0·1=0	
	Transfer to A	ETA				
	Add	EA				
	Subtract	ES				
	Insert	EI			2. (D) $\rightarrow$ A or (A) $\pm$ (D) $\rightarrow$ A	After EI: (D) = (M) . (Q)
	Insert and Store	EIS			1. M Q A A after 2. When result stored: 1 0 x x; 0 0 x x; 1 1 x 1; 0 1 x 0 — 2. (A) $\rightarrow$ D, (D) $\rightarrow$ M	
Special	Larger Word	LWD			* If (M) > (A), (M) $\rightarrow$ A and M address $\rightarrow$ JA	*Actually, (M) $\rightarrow$ D and (D) and (A) are compared in the alpha-numeric sense.
	Smaller Word	SWD			* If (M) < (A), (M) $\rightarrow$ A and M address $\rightarrow$ JA	
	No Operation	†NOP		L or R	No operation	†L or R specifies I_F as 0 or 1
	Halt	†HLT			Stop Computation	
	Transfer (JA) to Memory	‡TJM			(JA) $\rightarrow$ M address, $JA_F \rightarrow M_F$	‡L or R specifies the particular half word of M
	Transfer Instruction Address to JA	‡TIJ			Effective address $\rightarrow$ JA; $0 \rightarrow JA_F$ if L, $1 \rightarrow JA_F$ if R.	
	Increase Address in Memory	‡INCA			1 + M address $\rightarrow$ M address	
	Inhibit Clearing of Overflow Indicator, Set	ICOS			Inhibits clearing of O.F. indicator.	
	Inhibition on Clearing of Overflow indicator made Zero	ICOZ			Removes inhibition on clearing the overflow indicator (set by ICOS).	
	Transfer from Console Typewriter to Memory	TCM			One character $\rightarrow$ 6 right bit positions of M and D	#The skip instructions have a number of options described elsewhere in connection with the input-output instructions.
	Transfer from D to Console Typewriter	TDC			Transfer left (6 bits) character of D $\rightarrow$ typewriter	
	Transfer from Toggle Register to D	TTD			Word set up with toggle switches $\rightarrow$ D	
	Transfer Control to Input Output	TIO			(D) $\rightarrow$ input-output control; execute this I/O instruction.	
	Skipping Fault	SKF	I/O		If a fault exists, the next instruction is skipped.	
	Skip Check	SKC	I/O		Skip the next instruction if I_V > the number in the specified input counter.	
	(D) or (M) bit by bit stored	DORMS			Binary ones from (D) or (M) $\rightarrow$ D, M (1+0=0+1=1+1=1; 0+0=0)	
	Add without carry and store	AWCS			(M) + (A) without carries $\rightarrow$ D, M (1+1=0+0=0; 0+1=1+0=1)	

CHECKING FEATURES

All transfers of information between the 212 and memory are checked for odd parity. Each six bits in memory have an associated parity bit. Functionally, there are four checks performed within the system:

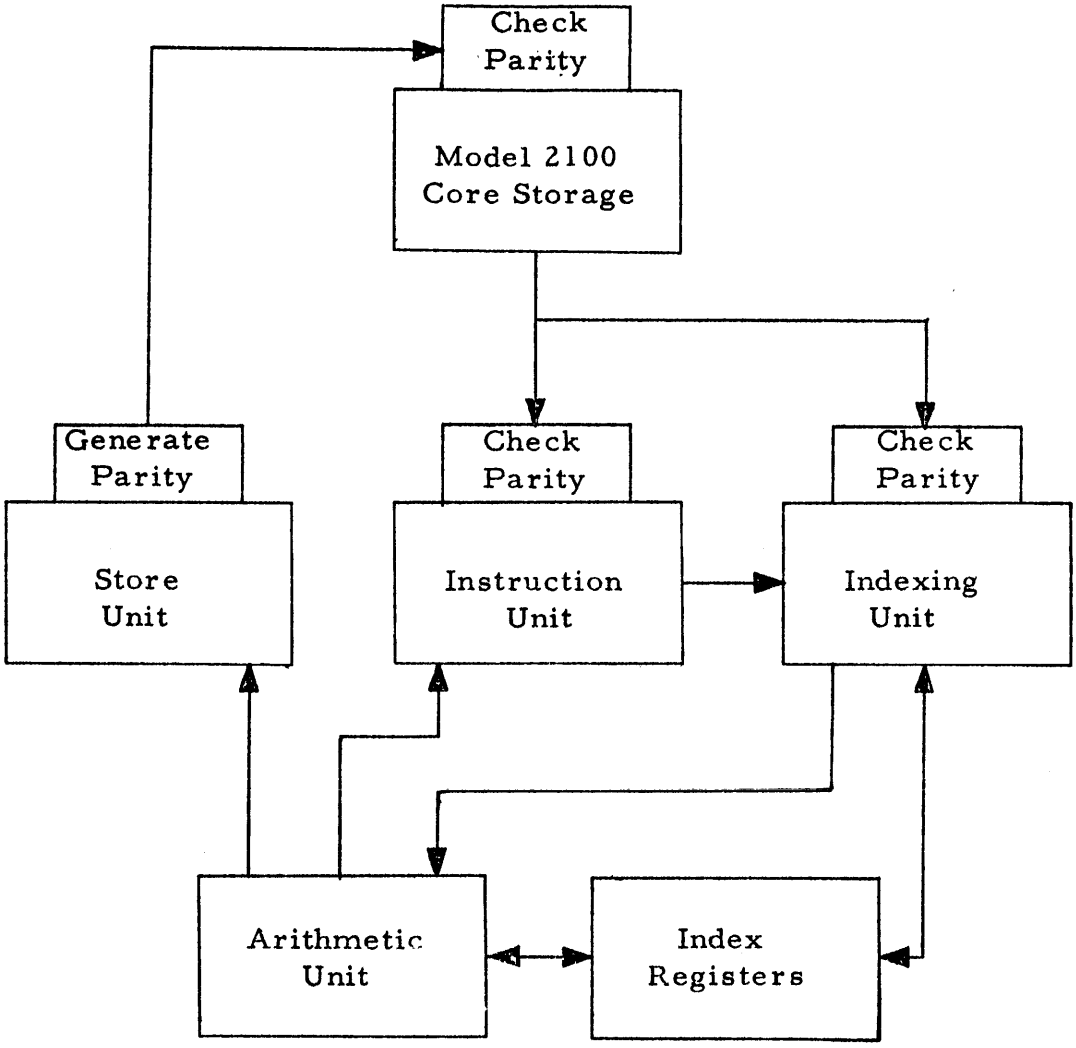
Instruction Parity Error (IPE)	-	A parity check is made at the Instruction Unit on each instruction word read from memory.
Operand Parity Error (OPE)	-	A check is made at the Indexing Unit on all operands read from memory.
Store Parity Error (SPE)	-	A check is made at the memory on all information received from the Store Unit.
Input-Output Parity Error (IOPE)	-	Information transmitted to the input-output section of the system is checked upon read-out from the memory. Information transmitted to the memory has parity generated at the memory prior to write-in.

A block diagram of the checking system in the Model 212 is given on the following page.

If there is no Auto-Control unit in the system (see Real-Time section), the occurrence of a parity error causes the computer to halt and an indication of the error is visually displayed on the console. Input-output operations in progress when the error is detected continue until completed. If an Auto-Control unit is in the system, the occurence of a parity error causes an interrupt which in turn allows programmed recovery.

INTERRUPT FEATURES

The Model 212 has built-in features which allow it to use an interrupt system to full advantage. Interrupt in the PHILCO 2000 System is accomplished by the Auto-Control Unit, which can monitor any of forty-eight system conditions and interrupt the program whenever any of these conditions occur.



Simplified Block Diagram, Model 212 Computer Checking System

Internal conditions in the Model 212 are included in the set monitored by Auto-Control. For instance, all conditions which could cause a halt in computer activity are monitored and, if desired, an interrupt is initiated instead of a halt. In this manner it is possible to conceive and execute programs which will never (except under extreme conditions) cause the computer to halt operation.

The internal conditions which are capable of initiating an interrupt are defined below.

IPE - An Instruction Parity Error has been detected.

OPE - An Operand Parity Error has been detected.

SPE - A Store Parity Error has been detected.

IOPE - An Input-Output Parity Error has been detected.

HALT - A Halt instruction has been encountered in the program.

BREAK - A JBT instruction has been encountered and the console Break Switch is set to halt.

CF - A Command Fault has been encountered in the program.

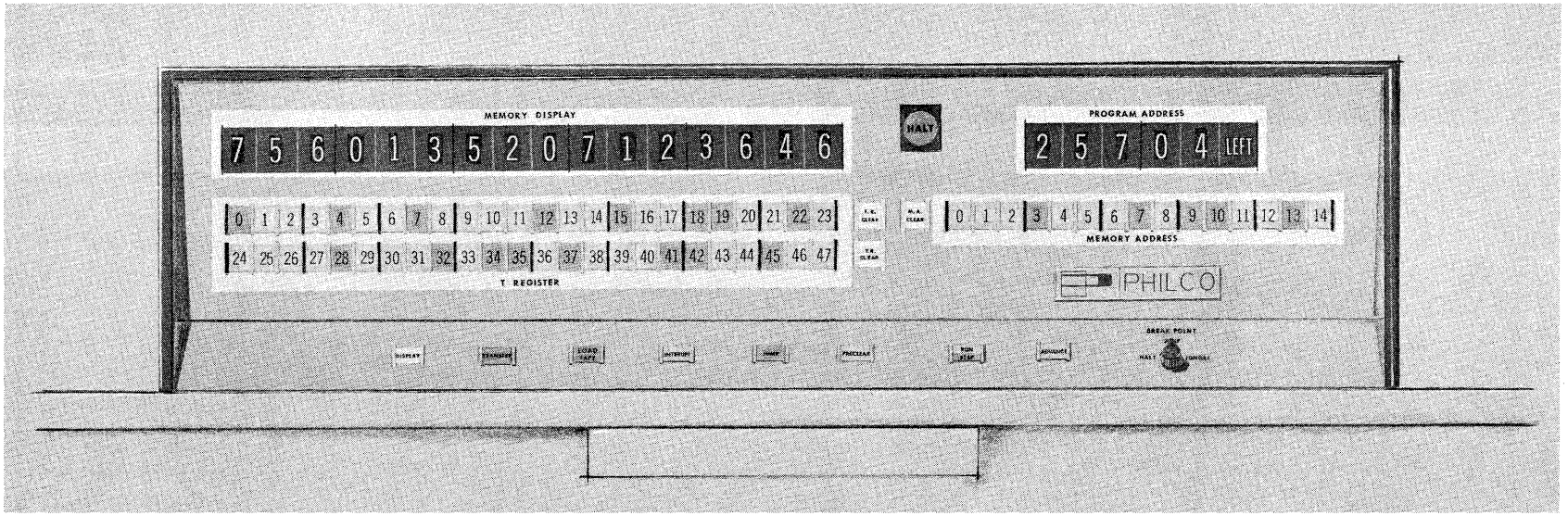
When an Auto-Control Unit is used with the Model 212, there are two restrictions as to when interrupt can occur:

a. Interrupt cannot take place when the computer is in the Repeat or Step modes.

b. If an interrupt is requested while a previous interrupt is being processed, further interrupt is inhibited until such time as TIO to the Auto-Control Unit and JL or JR instructions have been executed, in that order.

OPERATING CONTROLS

A simplified console provides the operating controls for the 212 system. The program address and contents of the memory location specified by the Memory Address Register are displayed in octal. The Memory Address Register and T Register are controlled by ranks of two-color indicating switches, blue indicating an Off



Model 212 Console Arrangement

state and yellow indicating an On state. Pushing the switch changes the condition. The T (Toggle) register holds all data to be manually inserted through the console. It is program addressable.

The console provides facilities to display the contents of any memory location, transfer the contents of the T register to any memory location, automatically load a program from tape, jump to any memory location, and interrupt a program. Other controls include a Run-Step mode switch and a Preclear switch.

TIMING CONSIDERATIONS

In any truly asynchronous computer, the algorithms, the operands, and the system status can all affect the time of any particular operation. Timing in the Model 212 is even more complex, because asynchronism exists between instructions as well as within an individual instruction. Determining the time required by a 212 operation therefore requires knowledge not only of the identity of the current instruction but also the degree of simultaneity between the processing of this instruction and the previous instruction.

The timing of a program is involved with memory use, which is a strictly variable factor depending on the frequency of use of memory banks. Unless a situation occurs in which a particular bank is used on every access, conflict between accesses is not a significant problem. Therefore, in all further discussion of times, memory conflict is ignored.

Time used in the Instruction Unit is largely a matter of instruction access. Except in "jump situations", this access time does not affect the time which must be allocated to an instruction except under extremely improbable situations. When a jump has taken place and instruction access is seen in the actual program time, 1.4 μsec is used in the Instruction Unit.

From a timing viewpoint, four types of instructions are processed by the Indexing Unit. The time required to generate an indexed effective address is 0.65 μsec; an unindexed effective address required 0.45 μsec. If a memory operand is desired, that operand appears in the Operand Register 1.1 μsec after the address is generated. Thus the Indexing Unit time for the four cases is as follows:

TECHNICAL DETAILS

MODEL 212 COMPUTER

The Model 212 computer contains the following program and data registers:

Register		Unit	Size	Function
Program Address	PA	IU	Address Length	Contains the address of the next word to be transferred from memory to the IU
Address Buffer	PA*	IU	Address Length	Contains the address of either the instruction word being processed or the address of the next instruction to be transferred to the IU Fills the JA Register during a jump instruction
Program Register	PR	IU	Word Length	Contains the next two instructions to be processed by the XU (except during jump instructions)
Program Buffer	PR*	IU	Word Length	Receives the instruction word from memory and transfers this word to the PR
Operand Address Register	OAR	XU	Address Length	Contains the effective address of the instruction being processed by the XU
Operand Register	OR	XU	Word Length	Receives and buffers the operand between memory and the AU
Index Unit Instruction Register	XIR	XU	Instruction Length	Contains the instruction being processed by the XU

Register		Unit	Size	Function
Index Buffer Register	XB	XU	Address Length	Contains the contents of the index register being processed in the XU
Repeat Register	RR	XU	11 bits	Contains control information for the Repeat mode
Arithmetic Unit Instruction Register	AIR	AU	Instruction Length	Contains the command and effective address of the instruction being performed in the AU
Jump Address Register	JA	AU	Address Length	Receives the return address for program on jump instructions
Accumulator Register	A	AU	Word Length	Contains the augend in addition, minuend in subtraction, and dividend in division Contains the sum in addition, difference in subtraction, product in multiplication, and remainder in division Contains one of two factors in a comparison
Accumulator Register*	A*	AU	Word Length	Alternate A Register
Data Register	D		Word Length	Receives all data transferred between memory and the AU Receives all data transferred between arithmetic registers

Register		Unit	Size	Function
Data Register	D		Word Length	Contains the addend in addition, subtrahend in subtraction, multiplicand in multiplication, and divisor in division Contains one of two factors in a comparison
Data Register*	D*	AU	Word Length	Alternate D Register
Quotient Register	Q	AU	Word Length	Contains multiplier in multiplication, less significant half of a double-length product in multiplication, the less significant half of a double-length dividend in division, and the quotient in division May contain a factor in a comparison Contains a masking pattern during an extracting operation
Quotient Register*	Q*	AU	Word Length	Alternate Q Register
Store Address Register	SAR	SU	Address Length	Receives the effective address of information to be stored in memory
Store Register	SR	SU	Word Length	Receives results of operations from the AU to be transferred to memory

COMPATIBILITY

The Model 212 is so designed that it can accept programs written for the Model 210 and 211 computers and execute them as a 210 or 211. In some few instances, deviation from this strict definition of ascending compatibility has produced a significant improvement in operation of the 212. These deviations and their implications are listed below.

a. Division - The Model 210 and 211 computers utilize a division algorithm which in some cases produces results that need correction. A new division algorithm is used in the Model 212 which produces exact division. To overcome this variation in operation, the correction routine used in the 210/211 programs must be removed when the program is to be used on the 212.

b. MAD, MSU, FMAD, FMSU - In the Model 210 and 211 computers, an overflow or exponent fault condition encountered in the multiply part of this instruction is sometimes corrected by the add (or subtract) cycle. These cases yield a false indication of the overflow or exponent fault condition. This false indication has been eliminated on the 212; since it was false, no change need be made for running a 210 or 211 program on the 212.

c. Skips - The unique action of skip instructions (TIO, SKC, and SKF) under repeat mode in the 210 and 211 is not considered useful on the 212, and these instructions will have no effect (except as fillers) when encountered under repeat on the 212.

d. Command Faults - Certain codes which are command faults on the 210 and 211 are legitimate instructions on the 212. Since these codes do not normally appear in 210/211 programs they do not violate compatibility.

e. Exponent Fault - Because of the logical organization of the 212, the index register involved is modified prior to detection of an exponent fault. This modification can be adjusted if necessary by the program which handles the exponent fault condition.

Certain functions have been added which do not violate ascending compatibility but which should be mentioned here. For instance, the addition of the Y-bits to the computer must be recognized. If a 212 program is written which uses the Y-bits and which also includes subroutines originally written for 210/211 use, compensation must be made in the form of minor modification to the subroutine entry and exit to allow for the proper storing and restoring of the Y-bits.

Furthermore, in order to ensure compatibility with the 210/211 programs, instructions which set the "C" bit on 210/211 computers also clear the Y-bit on the 212. The 210/211 programmer who writes for the 212 should note that he is working with the pair of bits rather than with just the "C" bit.

TIMING CONSIDERATIONS

Timing of the Model 212 can be determined only by considering the status of the entire computer/memory complex. It is necessary to understand first what each of the four units is doing, and the interaction of each unit on the others.

IU TIMING

The Instruction Unit timing consists of a sequence of transfers, each of which triggers the next. The sequence can be started in two ways.

A jump instruction causes a break in the normal sequence of selecting instructions, thus necessitating the following actions on the part of the Instruction Unit.

AIR	→	PA			
PA	→	MC			1.1 μsec
M	→	PR*			
PR*	→	PR	PA	→ PA*	0.15 μsec
PR_L	→	XIR	PA*	+ 1 → PA	0.15 μsec
			PA	→ MC	
			M	→ PR*	

The transfer of the right-hand instruction from PR indicates that both instructions in PR have been sent to the Indexing Unit. This then initiates the selection of another instruction-pair from memory as follows:

$PR_R \longrightarrow$ XIR		
PA $\longrightarrow$ PA*, PR* $\longrightarrow$ PR		0.15 μsec
PA* + 1 $\longrightarrow$ PA	}	1.1 μsec
PA $\longrightarrow$ MC		
M $\longrightarrow$ PR*		

XU TIMING

The timing of instructions in the Indexing Unit depends on the instruction, its operand, and whether or not it is indexed. The following cases illustrate the sequence and times for various types of instructions:

Instruction not indexed - no memory operand

PR $\longrightarrow$ XIR	0.15 μsec
XIR $\longrightarrow$ Adder $\longrightarrow$ OAR	0.45 μsec

Instruction not indexed - memory operand

PR $\longrightarrow$ XIR	0.15 μsec
XIR $\longrightarrow$ Adder $\longrightarrow$ OAR	0.45 μsec
OAR $\longrightarrow$ MC	
M $\longrightarrow$ OR	1.1 μsec

Instruction indexed - no memory operand

PR	⟶ XIR	0.15 μsec
XIR and (XR)	⟶ Adder ⟶ OAR	0.65 μsec

Instruction indexed - memory operand

PR	⟶ XIR	0.15 μsec
XIR and (XR)	⟶ Adder ⟶ OAR	0.65 μsec
OAR	⟶ MC	
M	⟶ OR	1.1 μsec

The function of the Indexing Unit is twofold; it prepares the instruction for execution and fetches an operand from memory. Thus if the instruction is ready for processing and the Arithmetic Unit is free, execution of the instruction is initiated even though the operand may not yet be available. The operation of the Arithmetic Unit is not delayed by the lack of an operand unless it can no longer continue without that operand.

An example of this operation is the case in which the low-order bits of a multiplier are zeroes. Since the result up to a point will be zeroes in any event, the multiply operation begins without the multiplicand. At the point where significant results appear, the multiplication is delayed until the multiplicand becomes available.

AU TIMING

Two timings occur in the Arithmetic Unit on each instruction executed in the AU. The transfer of the instruction to the Arithmetic Unit Instruction Register (XIR ⟶ AIR) and the decoding of the instruction each requires 0.15 μsec. The arithmetic algorithm itself involves combinations of the following times.

a.	Register Transfer	0.20 μsec
b.	Average Carry Time	0.06 μsec

c. Average Add Cycle (a+b)		0.26 μsec
d. Shift Cycle		0.18 μsec
e. Instruction Transfer		0.15 μsec

For instance, the algorithm time for a typical floating-point addition with a memory operand can be broken down as follows:

Transfer (OR) → D	(a, above)	0.20 μsec
Exponent Comparison	(a, above)	0.20 μsec
Arrange Cycles (2)	(d, above)	0.36 μsec
Add Cycle	(c, above)	0.26 μsec
Normalize Cycle (1)	(d, above)	0.18 μsec
Total Algorithm Time		1.20 μsec

Combining the above algorithm time with the instruction transfer time and decode time yields a total AU time for the add operation of:

(XIR) → AIR	0.15 μsec
AU Decode	0.15 μsec
Algorithm	1.20 μsec
Total AU time	1.50 μsec

SU TIMING

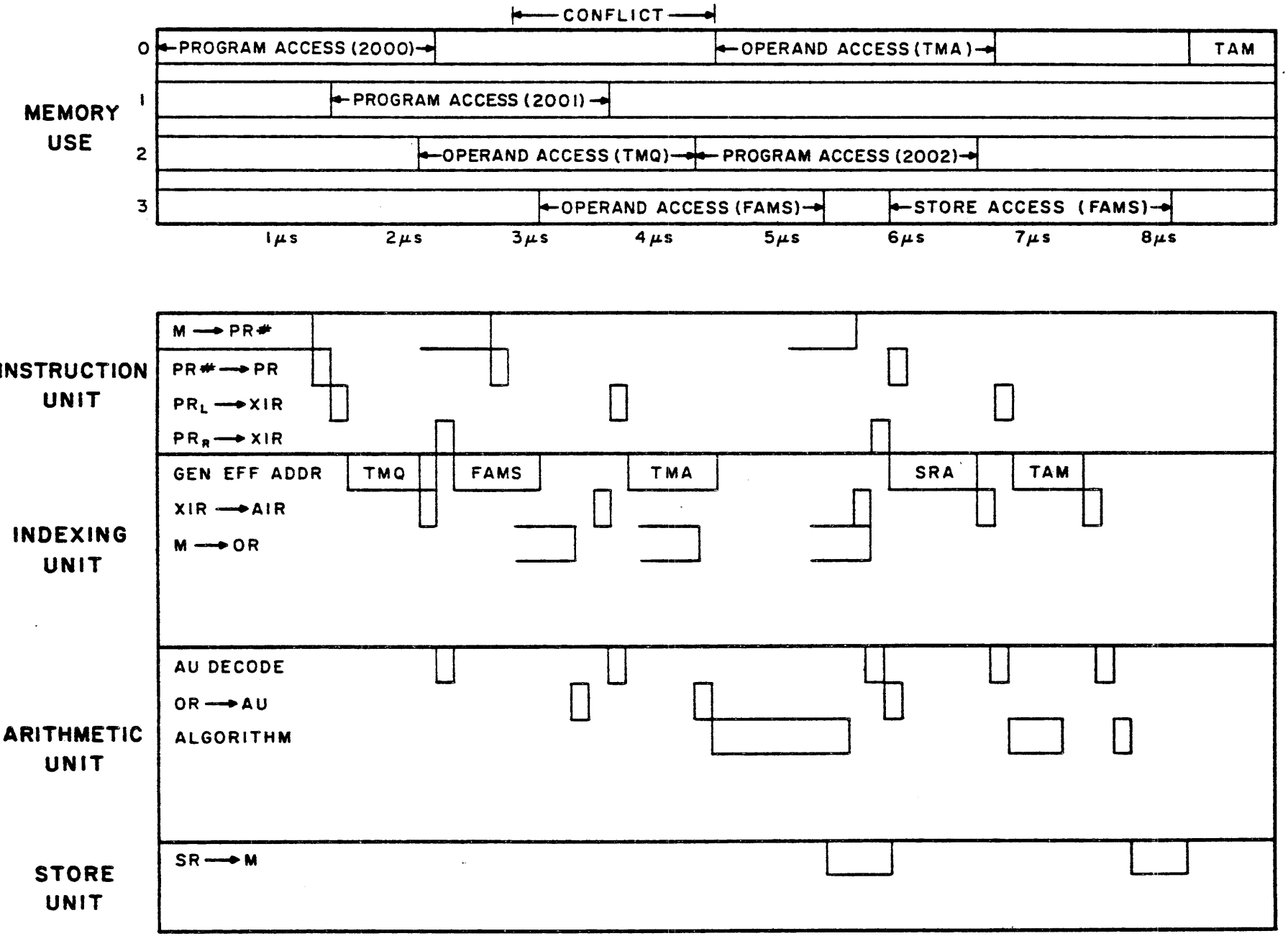
There are three basic timings associated with the Store Unit. The function of the Store Unit is stable; however, in that it performs the same task each time it is used. That is, it receives address and data information from the Arithmetic Unit, generates parity for the data word, and transfers that information to the memory. The total time for this operation is 0.5 μsec. Since this time is independent of AU time, it will always be covered by the following algorithm time in the overall program time.

PROGRAM TIMING

The complete timing picture can be assembled from the foregoing information. The timing of the following set of instructions, showing memory use and timing and the method of overlap between the computer units, is diagrammed on the following page.

OCTAL LOCATION	INSTRUCTION	OCTAL ADDRESS	OCTAL CONTENTS OF INDEX
1000	JMPL	2000	
2000 L	TMQ	3042	
R	FAMS	2, X	1051
2001 L	TMA	3, X	1051
R	SRA	0, X	6
2002 L	TAM	2450	

212 INSTRUCTION TIMING CHART



OPERATOR'S CONSOLE

The Model 212 Operator's Console is illustrated on the following page. The controls and indicators perform the following functions:

Status Indicator

This in-line display is located at the top center of the console. It presents a visual indication of the computer status as described below:

a. Green Dot: The computer is running and there is no internal reason for operator attention.

b. Red Dot: The computer is in the Step Mode.

c. BREAKPOINT: A JBT instruction has been encountered, the Breakpoint Switch is set to halt, and interrupt has not taken place.

d. COMMAND FAULT: A command fault has been detected and interrupt has not taken place.

e. HALT: A halt instruction has been executed and interrupt has not taken place.

f. TYPE: The computer is waiting on a TCM instruction for input from the console typewriter.

g. IPE: An instruction parity error has been detected and interrupt has not taken place.

h. OPE: An operand parity error has been detected and interrupt has not taken place.

i. SPE: A store parity error has been detected and interrupt has not taken place.

j. IOPE: An IO parity error has been detected and interrupt has not taken place.

k. INTERRUPT: The console interrupt button has been depressed, but interrupt has not taken place.

The indicator is so designed so that there is no conflict in the display. Only the following indications are mutually exclusive:

a. Green and Red Dots.
b. Breakpoint, Command Fault, Halt IPE, and Type

Program Address Display

This display indicates the location of the program being executed. Five octal positions and a left/right indicator show the address of the current instruction. The low-order digit has the following special meanings:

a. Normal display: The address is the location of the next instruction to be executed.
b. Digit displayed with a subscript -1: The address is one greater than the location of the next instruction to be executed.

T-Register

The T-Register is a program-addressable register controlled by forty-eight switches, each of which represents a bit setting of "zero" when blue and "one" when yellow. These are alternating switches which reverse the state of the associated bit when pressed. A Clear button located to the right of each half of the register sets all bit positions in the respective half-register to the "zero" (blue) state.

Memory Address

The Memory Address Register is controlled by fifteen switches to select any one of 32,768 memory locations for use with the Display, Jump, and Transfer buttons. Each switch represents a binary digit, the color blue meaning "zero", yellow meaning "one". These are alternating switches which reverse the state of the associated bit when pressed. A Clear button located to the left of the register sets all of the positions to the "zero" (blue) state.

Memory Display

This sixteen-digit octal display indicates the contents of the location selected by the Memory Address Register. The contents of the location are displayed each time the memory location is written into by the Store Unit.

Display Button

This button causes the contents of the location specified by the Memory Address Register to be displayed in the Memory Display. It is only effective when the computer is in the Step Mode.

Jump Button

This button causes control to be transferred to the location specified by the Memory Address Register. It is not effective when the computer is running and does not affect the contents of the JA Register.

Transfer Button

This button causes the contents of the T-Register to be transferred to the location specified by the Memory Address Register. It is only effective when the computer is in the Step Mode.

Interrupt Button

This button causes a bit to be set in the Auto-Control Unit. It has no effect when the Auto-Control Unit is not in the system.

Pre-Clear Button

This button clears the matrix on the tape system, returns the carriage on the console typewriter, and restores the central processor to starting conditions.

Run-Step Button

This button places the computer in either the Run or Step mode as follows:

Run: Instructions are executed in the normal manner after being started by the Advance, Load, or Jump buttons.

Step: Instructions are executed singly, one each time the Advance button is pressed. (When in the Step Mode, interrupt is inhibited).

Advance Button

This button causes one or more instructions (as designated by the Run-Step button) to be executed. This button is de-activated when the computer is running.

Load Button

This button causes a pre-clear of the computer (see above), the tape on logical channel zero to be rewound, and one block to be read (mode one) from the tape into location zero; when the read is complete, control is transferred to location zero and one instruction (zero left) is executed. Pressing the normally blue button changes it to yellow; upon successful completion of the sequence, the indicator returns to the blue state. This button is de-activated when the computer is running. In the Step Mode, the computer stops after the first instruction in location zero has been executed. If the computer is in the Run Mode but not running, (i.e., the mode has been changed from Step to Run but the Advance Bar or Jump Button has not been depressed) normal instruction sequence continues upon completion of the load operation. If for any reason the load sequence is not successfully completed the yellow indicator remains on and only the Load and Pre-Clear buttons are active. The Load button retries the sequence and the Pre-Clear button aborts the "load mode" and activates the Jump and Advance Buttons.

Audible Tone

A speaker located in the vicinity of the console emits an audible tone to indicate characteristics of the program being run. A switch located next to the speaker allows the tone to be turned off.

Breakpoint Switch

This three-position switch causes a JBT Instruction to jump, halt, or act as a NOP when executed.

All switches and indicators not specified above are colored blue in the normal (inactive) state and yellow in the active condition. In-line displays provide a white digit on a black background except where noted above.

The switches and displays are arranged for ease of readability in the following modes:

a. Octal (groups of three binary digits) - Memory Address and T-Register
b. Instruction Format - T-Register
c. Groups of three octal digits - Program Address and Memory Display.

INTERRUPT ACTIONS

The following two charts describe the complex of actions which occur upon detection of an error condition or a halt instruction. Chart I presents the interrrupt operation, Chart II the stop operation. For interrupt (Chart I), the Auto-Control Unit must be in the system and not inhibited and the computer must be in Run mode. Chart II applies when the Auto-Control Unit is not in the system (or is inhibited) or the computer is in Step mode.

CHART I - INTERRUPT

(Auto-Control in system and not inhibited, computer in Run mode)

	Computer Actions Before Permitting Interrupt	Conditions Required for Notifying Auto-Control	Normal Return Address from Interrupt Routine	Notes
IPE Instruction Parity Error	1. Halts before executing faulty instructions; awaits interrupt. 2. Indicates the error. 3. Ends repeat mode upon receipt of interrupt signal.	1. Auto-Control is not cycling for external reasons. 2. Computer reaches a point at which interrupt is permitted.	Address of faulty instruction	1. The Advance Bar and Console Jump Switch are not effective in Run mode. 2. Receipt of interrupt signal extinguishes the indicator.
OPE Operand Parity Error	1. Halts after faulty operand has been processed in AU; does not store results of operation. 2. Ignores any exponent fault. 3. Halts before executing next instruction; awaits interrupt. 4. Indicates the error. 5. Ends repeat mode upon receipt of interrupt signal.	Auto-Control is notified as soon as the error is detected, even if cycling.	Address in PA* when error was detected. (Usually one following the address of the instruction with a faulty operand).	See above
SPE-IOPE Store or Input-Output Parity Error	1. Halts before executing next instruction; awaits interrupt 2. Indicates the error; 3. Ends repeat mode upon receipt of interrupt signal.	Memory control is responsible for notifying Auto-Control	Address in PA* when error notification was received from memory control	See above
Command Fault	1. Halts before executing next instruction; awaits interrupt. 2. Indicates the error . 3. Ends repeat mode upon receipt of interrupt signal	1. Auto-Control is not cycling for external reasons. 2. Computer reaches a point at which interrupt is permitted.	Address of faulty instruction.	See above
HALT Instruction	1. If Auto-Control is cycling, allows interrupt before executing Halt instruction. 2. If Auto-Control is not cycling: a) Executes Halt instruction, b) Notifies Auto-Control, c) Indicates the halt, d) Awaits interrupt. 3. Ends repeat mode upon receipt of interrupt signal.	Computer reaches a point at which interrupt is permitted.	Address of the instruction following the Halt.	See above
JBT Instruction and Breakpoint Switch Set to Halt	1. Halts before executing the instruction; awaits interrupt. 2. Indicates the condition. 3. Ends repeat mode upon receipt of interrupt signal.	Auto-Control not cycling.	Address of the JBT instruction.	See above

CHART II - STOP

(Auto-Control inhibited or not in the system, or computer in Step mode)

	Computer Actions Before Stopping	Advance Bar Pressed	Console Jump Switch Pressed	Notes
IPE Instruction Parity Error	1. Halts before executing faulty instruction; awaits operator action. 2. Indicates the error.	No action	1. Jumps to address specified by Memory Address switch, without changing JA. 2. Extinguishes the indicator	Step mode inhibits computer recognition of an interrupt request; the condition is recorded in Auto-Control.
OPE Operand Parity Error	1. Halts after faulty operand has been processed in AU; does not store results of operation. 2. Ignores any exponent fault. 3. If next instruction is Halt, executes it and lights Halt indicator. 4. Indicates the OPE error.	1. Executes the next instruction in either Run or Step mode. 2. Does not end repeat mode. 3. Extinguishes the indicators.	See above	See above
SPE-IOPE Store or Input-Output Parity Error	1. Halts before executing next instruction; awaits operator action. 2. Indicates the error.	See above	See above	See above
Command Fault	1. Halts before executing faulty instruction; awaits operator action. 2. Indicates the error.	No action	See above	See above
HALT Instruction	1. Executes Halt instruction. 2. Indicates the halt.	See OPE	See above	See above
JBT Instruction and Breakpoint Switch Set to Halt	1. Halts before executing the instruction; awaits operator action. 2. Indicates the condition.	See OPE	See above	See above

Philco 2000
Instruction Manual

THE REAL-TIME SYSTEM

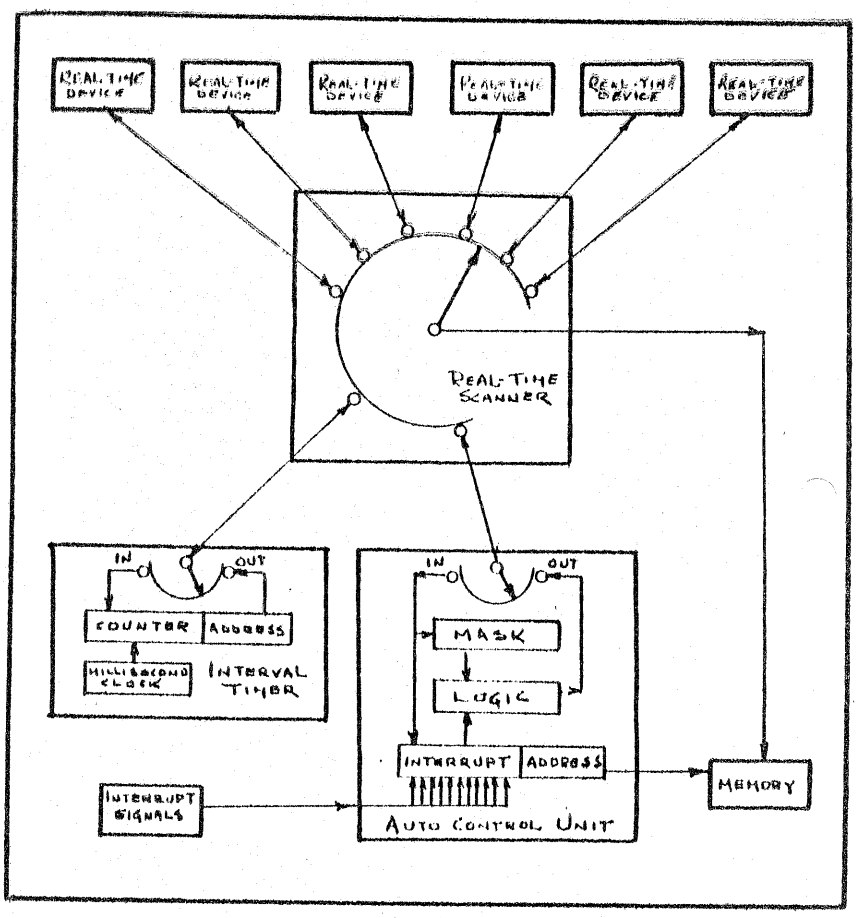
INDEX

PHILCO 2000 INFORMATION MANUAL

Real-Time System

GENERAL INFORMATION

REAL-TIME SYSTEM



Real-time computer operations are characterized by three basic conditions:

a physical operation must always be present in a real-time system;
the central computer must process within the time cycle required by the physical operation;
the central computer re-acts to the operation and exercises control on a feedback principle.

In commerce, for example, bank deposit accounting is a real-time process. The bank teller's deposit recording device provides the real-time input to the central computer, which in turn updates the depositor's account and transmits the new balance to the teller. This process occurs in real-time and the depositor is advised of his balance at that point in time. In industry, a perpetual inventory can be maintained by real-time devices at issue and receiving locations, with the central computer calculating stock status and transmitting all of the instructions to maintain the inventory operation. In military applications, missile and aircraft tracking is an example of real-time processing.

The PHILCO 2000 Real-Time System, when associated with the real-time devices unique to a given processing problem, provides the hardware to answer specific needs. The System consists of three components: the Real-Time Scanner, the Auto-Control Unit, and the Interval Timer. These components and other real-time units can be assembled to form various real-time sub systems.

The Real-Time Scanner is an intermediate device which provides an information channel to the computer and transfers information between the central computer and a real-time unit. The Model 304 Real-Time Scanner multiplexes and checks up to four units; the Model 308, up to eight units. The Scanner interrogates in turn each unit connected to it. Transmissions ordered by the program are initiated and controlled by the scanner, which multiplexes the transmissions between each of its active real-time units and memory. This multiplexing of devices continues automatically until all orders issued to the scanner by the program have been processed.

The Auto-Control Unit is a real-time device connected directly to the real-time scanner to provide a means of interrupting a program whenever specified conditions appear in either the central computer or the input-output system. This interrupt feature is completely under program control. As a result, the programmer determines the system conditions which permit interruption of the computer and the action to be taken when these conditions are noted. In addition to providing a programmer with maximum control over the computing system, the Auto-Control Unit makes efficient use of program time by automatically monitoring computer operations.

The Interval Timer Unit works through the real-time scanner to time, in milliseconds, intervals of up to nine hours in duration. The timer is an electronic "alarm clock" which announces that a predetermined amount of time has alapsed. This alarm clock enables the programmer to interrupt a program, to stop an operation at a specific time, and to interlace programs so that one program follows another at a specific time. To interrupt a program, the timer signals the Auto-Control Unit that the predetermined amount of time has elapsed. The Auto-Control Unit then transmits a signal to the computer and the program determines the action to be taken. If the system does not use an Auto-Control Unit, a time reference read-out can be obtained.

FUNCTIONAL DESCRIPTION

REAL-TIME SYSTEM

REAL-TIME SYSTEM

FUNCTIONAL CHARACTERISTICS

Scanner

Transmission Mode	-	Parallel 48-Bit Information Word
Transmission Rate	-	640,000 characters per second
Channels	-	4 or 8
Scanning Rate	-	0.2 microseconds channel-to-channel

Auto-Control Unit

Automatic Interrupt - Maximum 48 interrupts

Interval Timer

Timer - Milliseconds - Maximum 9 hours

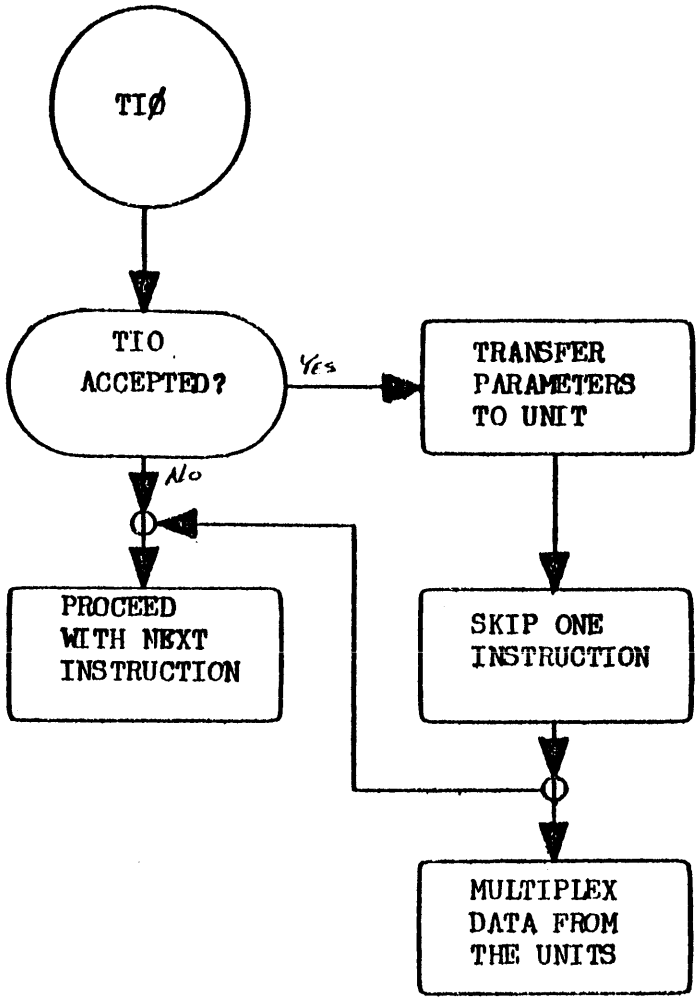
The Real-Time Channel and the Real-Time Scanner are the nucleus of the Real-Time System. A 48-bit information word is transmitted in parallel over the Real-Time Channel; transmission is bi-directional. The terminal points of the channel are the computer memory and the Real-Time Scanner.

FLEXIBILITY

The flexibility of the PHILCO 2000 is illustrated by the design of the Real-Time Scanner. Although there are presently only two real-time devices in the PHILCO 2000 system (the Auto-Control Unit and the Interval Timer) facilities for accommodating more such devices are built into the scanner. This type of open-ended design provides a direct means of communication between the PHILCO 2000 and any other device which has suitable controls. Thus if need arises for the incorporation of special purpose devices such as on-line data links, inquiry stations, or visual readouts, the system has the ability to handle the requirements without re-design.

REAL-TIME SCANNER

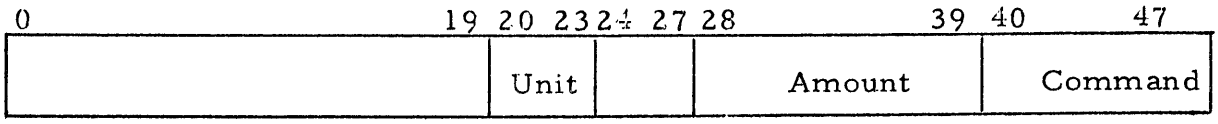
The Real-Time Scanner is a multiplexing device which accommodates four (Model 304) or eight (Model 308) real-time input-output units. Multiplexing is accomplished by continuously



Flow Chart of Real-Time Scanner Only

interrogating each real-time unit in turn to check status, either information ready or inactive. If a unit is inactive, the next real-time unit is interrogated. Scanning continues at a 0.2-microsecond point-to-point rate until an information-ready status of one of the real-time units is encountered. At this time a word is transmitted to or from memory and the scanning continues.

Transmissions to and from the Real-Time System are initiated by the execution of a TIO (Transfer to the I-O Section) instruction when the D Register contains a real time order as shown below. When an order is accepted by the Real Time System, it is processed to completion independently of the program.



Format of a Real-Time Input-Output Order

UNIT is the number of Real-Time Unit.

AMOUNT is the amount of information to be transferred.

COMMAND may be

Quaternary 1101 (RDRT) Real Time System → Memory
Quaternary 0111 (WRRT) Memory → Real Time System

The function of the Real-Time Scanner is to accept the parameters of a Real-Time I-O order, transmit the necessary information to the designated unit, and transfer information between the memory and that unit.

The sequence is as follows: A Real-Time Input-Output order is placed in the D register and the TIO is then issued in the Program Register. If the TIO is accepted*, the parameters of the order are sent to the Real-Time Scanner. The Real-Time Scanner then transfers the parameters to the unit specified by the UNIT field. The parameters are as follows:

* The TIO is always accepted unless the unit specified is unavailable or busy.

a. The command, which sets the unit in either input or output mode of operation.

b. The amount of data to be transferred.

c. The address of the first memory location to be accessed (obtained from the V field of the TIO instruction).

The Real-Time Scanner then interrogates in turn each unit connected to it. When the specified unit is ready to receive or transmit information, one word is transferred between the memory and the designated unit. This action continues until all Real-Time orders are completed.

SKIP INSTRUCTIONS

Four special skip instructions are used with the Real-Time Scanner:

a. SKCRTI, a skip check instruction which checks the ability of a unit to accept an RDRT order.

b. SKCRTO, a skip check instruction which checks the ability of a unit to accept a WRRT order.

c. SKFRTI, a skip fault instruction which checks the operation of an RDRT order.

d. SKFRTO, a skip fault instruction which checks the operation of a WRRT order.

The format of the Real-Time skip instructions is shown below.

		UNIT						COMPARISON QUANTITY									COMMAND							
	S V_{15}	V_{14}	V_{13}	V_{12}	V_{11}	V_{10}	V_9	V_8	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0								
SKCRTI	0	x	x	x	1	1	0	0	0	0	0	x	x	x	0	0	0	0	0	0	0	1	1	0
SKCRTO	0	x	x	x	0	1	0	0	0	0	0	x	x	x	0	0	0	0	0	0	0	1	1	0
SKFRTI	0	x	x	x	1	1	0	0	0	0	0	x	x	x	0	0	1	0	0	0	0	1	1	0
SKFRTO	0	x	x	x	0	1	0	0	0	0	0	x	x	x	0	0	1	0	0	0	0	1	1	0

Format of Real-Time Skip Instructions

V_{15} is always equal to zero, and V_{11} to V_{14} specify the unit being checked (V_{11} equals "1" if an input order is being checked and equals "0" if an output order is being checked). V_9 and V_{10} equal "1" and "0" respectively, and indicate that the skip instruction applies to a device connected to the Real-Time Scanner.

The x's in the comparison quantity indicate the specific bits which are checked during a skip instruction. The three bits in the comparison quantity of the SKCRTI and SKCRTO instructions indicate that the unit being checked is mechanically available, processing data, or waiting to process data. The specific meaning of each bit is shown below. The meaning of the specific bits in the comparison quantity of the SKFRTI and SKFRTO instructions depend upon the specific unit being tested.

COMPARISON QUANTITY

V_8	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0
0	0	0	0	0	x	x	x	0

V_1 - Waiting to Process Data
V_2 - Processing Data
V_3 - Mechanically Available

Meaning of the Bits in the Comparison Quantity
of the SKCRTI or SKCRTO Instructions

AUTO-CONTROL UNIT

The Auto-Control Unit provides a means of program-controlled automatic interruption of the central computer program. It is directly connected to the Real-Time Scanner.

Interruption requests are initiated by the availability status and/or error conditions of the various input-output devices and the service demands of the real-time devices. Acceptance of an interruption request and consequent action is controlled by the computer program. The Auto-Control Unit can recognize and control forty-eight different interrupt conditions.

The following registers are used in recognizing and acting upon interruptions.

1. Auto Control Register - Auto Control Unit
2. Mask Register - Auto Control Unit
3. Memory Address Register - Auto Control Unit
4. Jump Address Register - Central Computer
5. Program Address Register - Central Computer

The 48-bit Auto-Control Register is physically connected to the devices which can generate interruption requests. Each of the bits is related to a given condition and device for which an interrupt may be required. A zero in the bit position of the Auto-Control Register indicates that the condition is not present; a one, that the condition is present.

The 48-bit Mask Register is loaded under program control with a word of information which is a replica of the Auto-Control Register, except that a one-bit is contained in each relative position for only those device conditions permitted to interrupt at that time.

The central processor Jump Address and Program Address Registers serve to coordinate the interruption with the main program.

The Auto-Control operation starts with the issuance of a TIO instruction. The following register activity takes place:

1) The mask contained in memory at the address of the TIO instruction is loaded into the Mask Register of the Auto-Control Unit.

2) The address portion of the TIO instruction is incremented by one and loaded into the Memory Address Register of the Auto-Control Unit, which is now triggered to accept an interrupt.

The Auto-Control system is now set to interrupt; processing continues until a condition for which an interrupt is required presents itself. At this time, the following activity occurs:

1. A one-bit appears in the relative position of Auto-Control Register.

2. If the Mask Register contains a corresponding bit, the contents of the Auto-Control Register are transferred through the mask to the address contained in the Auto-Control Memory Address Register.

3. The interrupting bits of Auto-Control Register are cleared to zero.

4. A signal requesting interrupt is sent to the computer.

5. The computer is interrupted as the pair of instructions is brought up (unless in Repeat or Step modes).

6. Further interrupts are prevented until the Auto-Control Mask Register is reloaded.

The central computer has now accepted an interruption request and transfered control to the interrupt processing program for appropriate action and subsequent return of control to the main program. The following register activity takes place at this time.

1. The contents of the Jump Address Register are transferred to the memory location at A/C MAR +1.

2. The contents of the Program Address Register are transferred to the Jump Address Register to preserve re-entry data.

An array of memory at the location of the interrupt program subroutine appears as follows:

TIO Address	=	Location of mask control
MAR → TIO Address +1	=	Mask indicating interrupts
MAR + 1 → TIO Address +2	=	Contents of JA at interruption
MAR + 2 → TIO Address +3	=	First instruction of interrupt subroutine
		↕
	=	End of subroutine
TMD	=	Insert in D the output order to

		Auto-Control that will reload mask in the Auto-Control Mask Register and set up for next interruption
TIO	=	Execute I/O order in D; the address of the TIO specifies the location of the mask in memory
JL/R	=	Return to main program by a JL or JR instruction

The following table illustrates the types of program interrupt requests which can be initiated by the various equipments of the PHILCO 2000 Input-Output Subsystem. Other external devices connected to the system can also set special bits in the Auto-Control Interrupt Register to demand program interrupts.

NO.	TYPE OF INDICATION	MEANING	SUGGESTED PROGRAM ACTION
1	Assembler Complete	Indicates that an assembler is released	1. Update Input-Output orders list. 2. Set the necessary "flags" to indicate which operations have been completed in order that the program can take appropriate action. 3. Issue new Input-Output orders.
2	Input-Output Errors	Indicates that an error has been detected by the IOP during input transmission from the UBC or during any magnetic tape transmission	1. Tag entry as questionable. 2. Determine exact nature of error and take appropriate action.

NO.	TYPE OF INDICATION	MEANING	SUGGESTED PROGRAM ACTION
3	Buffer Error	Indicates that an error has been detected by the UBC during output transmission between the UBC and an input-output system	1. Determine type of input-output device in use. 2. Determine type of error. 3. Take appropriate corrective action.
4	Buffer Complete	Indicates successful completion of data transmission between the UBC and an input-output device	1. Update Input-Output orders list. 2. Add UBC to list of available input-output units.
5	Console Interrupt	Indicates that the Console Interrupt pushbutton on the Central Computer console has been pressed	1. Interrogate Toggle Register or Interrogate Console Typewriter. 2. Take specified action.
6	Card Reader Ready	Indicates that the START pushbutton on the Card Reader has been pressed	1. Update Input-Output orders list. 2. Add Card Reader to list of available units. 3. Issue punched-card Input-Output order.
7	Paper Tape Ready	Indicates that the READY pushbutton on the off-line Paper Tape Reader has been pressed	1. Update Input-Output orders list. 2. Add Paper Tape System to list of available units. 3. Issue Paper Tape order.

INTERVAL TIMER UNIT

The Interval Timer is a real-time device directly connected to the Real-Time Scanner. This electronic chronoscope adds a time reference to the computing system, counting up to nine hours in units of milliseconds.

The operation starts with a TIO output instruction referencing the Interval Timer. The contents of the memory location addressed (low order 25 bits) are transferred to the Interval Timer Register to represent the interval of time needed for control purposes. Clocking is accomplished by decrementing the Interval Timer Register by millisecond.

Time data can be obtained from the Interval Timer in two ways.

1) A TIO input order referencing the Interval Timer causes the contents of the Interval Timer Register to be transferred to the low-order 25 bits of the memory location specified by the TIO instruction. The 23 high-order bits are set to zero. The register reading is then available for appropriate program action.

2) If there is an Auto-Control Unit in the system, a bit will be set in the Auto-Control Register when the timer is decremented to zero.

If a TIO output order referencing the Interval Timer is accepted, the low-order 25 bits of the memory location specified by the TIO instruction address are set into the timer and the decrementing of the timer is initiated. For continued use, the Interval Timer must be reset with a TIO output instruction after each alarm signal.