



Control Data[®]
7600 Computer System

Preliminary System Description



September, 1969
©1969, Control Data Corporation
Printed in the United States of America
This edition obsoletes all previous editions

7600 SYSTEM DESCRIPTION (Preliminary)

Introduction

The 7600 system is the result of a development program to provide computing capacity substantially beyond that of the 6600 systems. Central processor computation exceeds the corresponding computation in the 6600 system by a factor of four. The 7600 system is designed to be machine code upward compatible with the 6400/6500/6600 systems in the area of central processor routines. The 7600 system input-output provisions have been generalized and greatly expanded over those proved in the 6400/6500/6600 systems. Input-output data rates are not expected to average substantially higher on a per channel basis than the rates in the 6600 systems. A much larger volume of input-output data is handled in the 7600 system by a much larger number of input-output channels.

The 7600 system contains a central processing unit (CPU) and a number of peripheral processing units (PPU). Some of the PPU are physically located with the CPU and others may be remotely located. The PPU communicate with the CPU over high speed data links with the data buffered at the CPU end of the data link. The PPU provide a communication and message switching function between the CPU and individual peripheral equipment controllers. Each PPU has a number of high speed data links to individual peripheral equipment controllers in addition to the data link to the CPU. The PPU time shares the data link to the CPU among the peripheral equipment controllers on a record by record basis.

The 7600 system is designed to accommodate multiple operating stations. New peripheral equipment configurations are being developed which will operate from programmable equipment controllers and are specifically intended for this application. These controllers will be able to communicate with the PPU on a record by record basis at higher rates than the existing 6400/6500/6600 equipment.

7600 system parameters

CPU computation section

60-bit internal word
binary computation in fixed point and floating point format
nine independent arithmetic units
twelve word instruction stack
synchronous internal logic with 27.5 nanosecond clock period

CPU small core memory

65,536 words of coincident current memory (60 bit + 5 parity)
32 independent banks
2048 words per bank
275 nanosecond read/write cycle time
27.5 nanosecond per word maximum transfer rate

CPU large core memory

512,000 words of linear select memory (60 bit + 4 parity)
8 independent banks
64,000 words per bank
1760 nanosecond read/write cycle time
8 words read simultaneously each reference
27.5 nanosecond per word maximum transfer rate

CPU input-output section

15 independent channels (asynchronous)
each channel full duplex (12 SCM buffer areas of 64, 128 or 256 words allocated per channel)
55 nanosecond per 60-bit word transfer rate

PPU computation section

12-bit internal word
binary computation in fixed point
synchronous internal logic with 27.5 nanosecond clock period

PPU core memory

4096 words of coincident current memory (12 bit + 1 parity)
two independent banks
2048 words per bank
275 nanosecond read/write cycle time

PPU input-output section

8 independent channels (asynchronous)
each channel full duplex (12 bit)

System communication

The 7600 system is divided into a number of major sections which are interconnected as shown in Figure 1. All input data enters the system at a programmable peripheral equipment controller. All output data leaves the system at a programmable peripheral equipment controller. The PPU serve to gather input data from these peripheral equipment controllers for delivery to the CPU for processing, and distribute processed data to these equipment controllers for output devices. The communication between PPU and equipment controller is generally limited by the rate at which the equipment controller can deliver or accept data. Most equipment controllers in the 7600 system will contain a core memory buffer capable of holding one record of data for the attached input or output devices. The communication between PPU and equipment controller is then a one record burst of data followed by a relatively long period of inaction. The PPU is then able to time share its channel to the CPU among a number of equipment controllers without storing more than one record of data in its own memory at any one time.

Communication between a PPU and a peripheral equipment controller is over a 12-bit full duplex channel. Each channel has a 12-bit data path from PPU to controller, and a separate 12-bit data path from controller to PPU. These two data paths are independent and may operate simultaneously. Each path has two associated control lines carrying control information in the direction of data flow. These lines carry a "word flag" to indicate passage of each 12-bit word of data, and a "record flag" to indicate the completion of a record of data. Each path has one associated control line carrying control information against the direction of data flow. This line carries a "word resume" signal to indicate receipt of a data word.

Communication between a PPU and the CPU is over a 12-bit full duplex channel identical to that described above. The 12-bit data path from PPU to CPU includes a 60-bit assembly register at the CPU end of the data link. This register assembles five 12-bit words into a 60-bit word for entry into the CPU memory. The 12-bit data path from CPU to PPU includes a 60-bit disassembly register at the CPU end of the data link. This register disassembles a 60-bit word from the CPU memory into five 12-bit words for transmission over the data link.

A maximum of 15 PPU may be directly connected to the CPU. Each CPU channel has assembly and disassembly registers to convert from 60-bit to 12-bit word length. All 15 CPU input-output channels may be in operation at the same time. Data is transmitted to, or from, PPU on a record by record basis. The CPU program is interrupted at the end of a record transmission to exchange control information with the communicating PPU or to initiate transmission of another record. On very long records the CPU program is interrupted at prescribed intervals in the buffer data. The frequency of this interruption is a function of the buffer size and may be preset individually for each CPU channel.

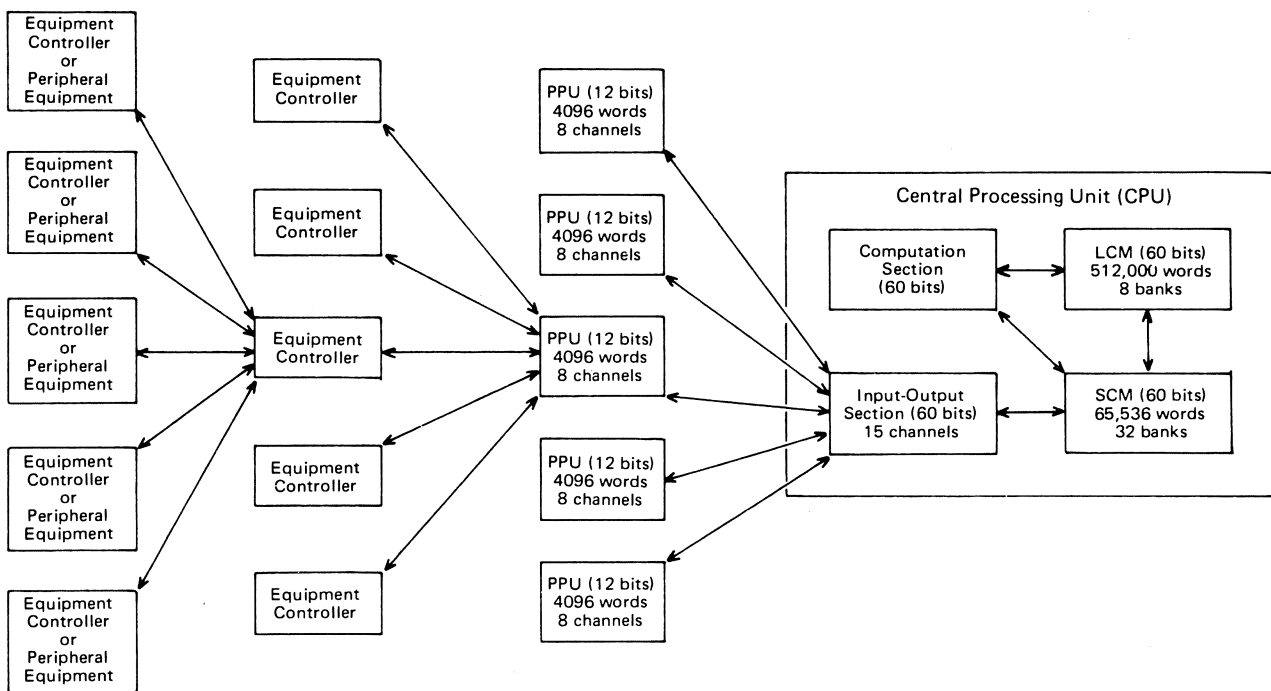


Figure 1 7600 System Communication

Operating system

The 7600 hardware was designed with a particular software approach in mind. This approach is an outgrowth of experience with the Chippewa and SCOPE Operating Systems for the 6600. The 7600 system software will be described in a following section of this manual.

System monitor

The system monitor is a CPU program in the 7600 system. This program is loaded with the operating system on a machine "dead start" and remains in the CPU core memory as long as the operating system is used. A portion of the system monitor resides permanently in the small core memory (SCM) section of the CPU. This portion of the monitor program is called the "resident monitor" program. The bulk of the system monitor resides in the large core memory (LCM) section of the CPU. This portion of the monitor program is called piecemeal into the SCM for execution as overlays on the resident monitor program.

Object program

An object program is defined in this manual to mean any CPU program other than the system monitor program. This term is used to describe generally a job oriented program. An object program may be a machine language program such as a FORTRAN compiler, or it may be a program which results from compiling FORTRAN statements with a compiler.

Central processing unit (CPU)

The CPU is a single integrated data processing unit. It consists of a computation section, small core memory, large core memory, and input-output section. These sections are all contained in one main frame cabinet and operate in a tightly synchronous mode with a clock period of 27.5 nanoseconds. Communication with equipment outside of the main frame cabinet is asynchronous.

CPU core memory

The CPU contains two types of internal core memory. One type, designated as the small core memory (SCM), is a many bank coincident current type memory with a total capacity of 64K words of 60-bit length ($K = 1024$). The other type, designated as the large core memory (LCM) is a linear selection type of memory in which eight 60-bit words are addressed as a single unit. The LCM has a total capacity of 500K words of 60-bit length. These two types of internal memory have significantly different system functions in the CPU.

The SCM is arranged in 32 banks of 2K words each. Each bank is independent of the other 31 banks. Maximum data transfer rate between the SCM as a unit and other parts of the system is one word each clock period. Each SCM bank has a four clock period access time from arrival of the storage address to readout of the 60-bit word. The total read/write cycle time for a SCM bank is ten clock periods. It is thus possible for a maximum of ten SCM banks to be in operation at one time. This maximum occurs during block copy instructions between SCM and LCM in which the addresses for sequential words cause no SCM bank conflicts. In random addressing of the SCM for CPU program data, CPU instructions, and input-output channel data, an average of four SCM banks in operation at one time is more normal.

The SCM performs certain basic functions in system operation which the LCM cannot effectively perform. These functions are essentially ones requiring rapid random access to unrelated fields of data. The first 4K addresses in SCM are reserved for input-output buffer, control areas, and resident monitor program. These areas are addressed by the CPU input-output section as required to service the communication channels to the PPU. CPU object programs do not have access to these areas.

The remainder of the SCM is divided between fields of CPU program code and fields of data for the currently executing program.

The LCM is arranged in eight banks of 64K words each. Each bank is independent of the other seven banks. A storage reference to a LCM bank results in a read/write cycle which takes 64 clock periods. Eight 60-bit words are read simultaneously from a LCM bank whenever a read/write cycle occurs. These words are held in a 480-bit operand register for each LCM bank. Subsequent reference to a word residing in one of these operand registers allows either read or write function without the delay of a bank read/write cycle. Maximum data transfer rate between the LCM as a unit and other parts of the system is one word each clock period. This maximum transfer rate occurs during block copy instructions between LCM and SCM. LCM bank read/write cycles are anticipated in the block copy operation to avoid a bank access delay.

The LCM provides the basic working storage for the CPU. All object programs are assembled here for execution in the SCM. All data files are buffered through LCM for the object programs. Small object programs are generally run to completion in SCM with the complete input file in LCM at the beginning of execution, and the complete output file in LCM at the end of execution.

The low order addresses in LCM are reserved for system tables, monitor program overlays, object time library, and compilers. These areas require approximately 65K of the 500K available stor-

age. The remainder of LCM is divided into two fields, each holding a program. A copy of one of the programs is in SCM in execution. The other program is waiting for execution.

Computation section

The computation section of the CPU contains nine segmented arithmetic units, 24 operating registers, and a 12 word instruction stack. These units work together to execute a CPU program stored in the SCM. Data moves into, and out of, the computation section of the CPU through the operating registers. Data may be directly addressed in either the SCM or the LCM. The general information flow in this section is illustrated in Figure 2.

Instruction word stack

The instruction word stack is a group of twelve 60-bit registers in the CPU computation section which hold program instruction words for execution. The instruction stack information is essentially a moving window in the program code. The stack is filled two words ahead of the program address currently being executed. A small program loop may frequently be entirely contained within the instruction stack. When this happens the loop may be executed repeatedly without further references to SCM.

The current instruction word is contained in a special register in the CPU computation section. This register is designated the CIW register. Program instruction words are read one at a time from the instruction stack and are interpreted in the CIW register for execution. This register controls all of the data transmission paths between the operating registers and the functional units in the computation section.

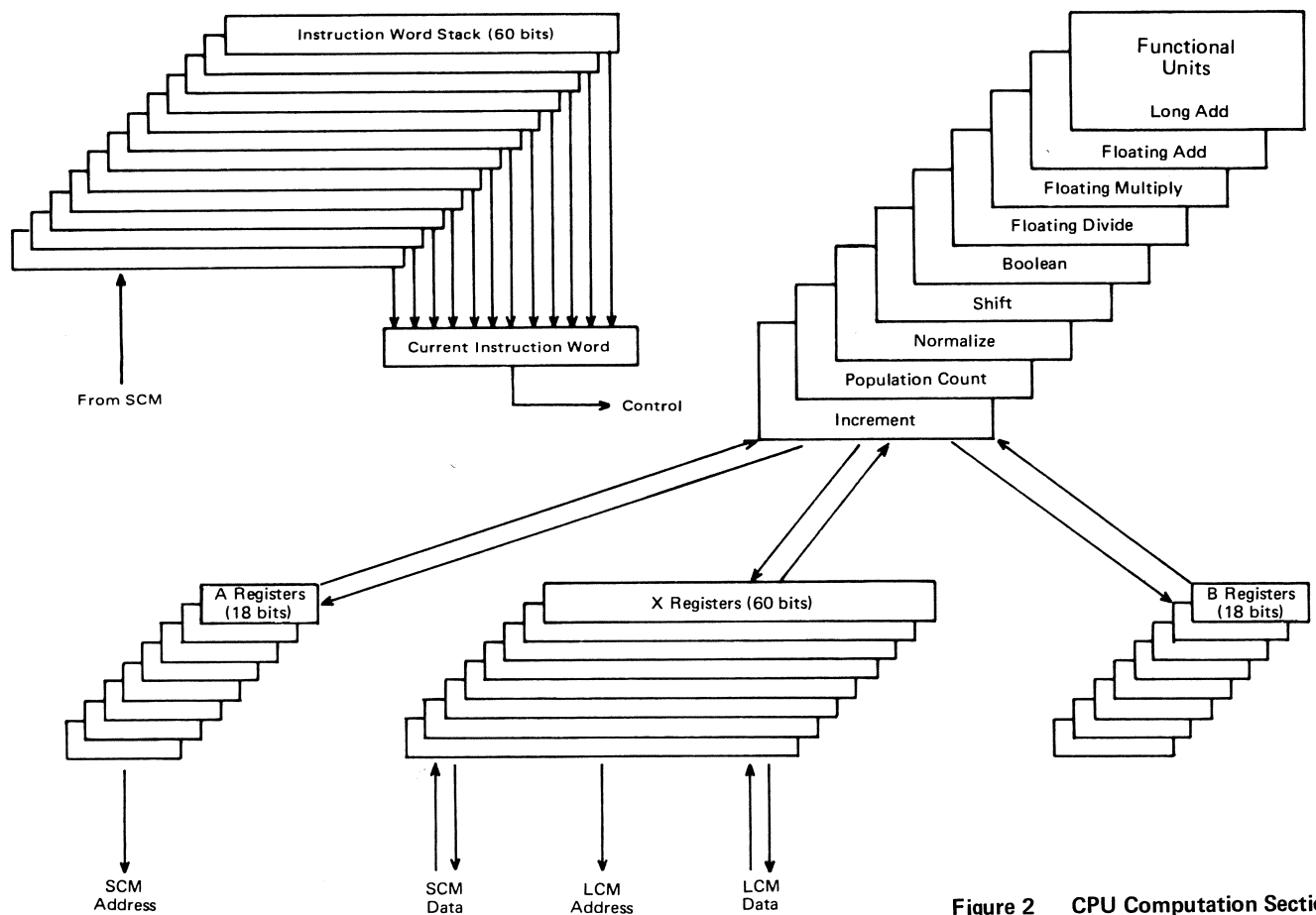


Figure 2 CPU Computation Section

X registers

There are eight 60-bit X registers in the computation section of the CPU. These registers are the principal data handling registers for computation. Data flows from these registers to the SCM and the LCM. Data also flows from SCM and LCM into these registers. All 60-bit operands involved in computation must originate and terminate in these registers.

A registers

There are eight 18-bit A registers in the computation section of the CPU. These registers are essentially SCM operand address registers. These registers are associated one-for-one with the X registers. When an address is entered in an A register the corresponding data in the X register is normally read from, or stored into, SCM at that address.

B registers

There are seven 18-bit B registers in the computation section of the CPU. BO is inferred but does not actually exist. These registers are primarily indexing registers for controlling program execution. Program loop counts may be incremented or decremented in these registers. Program addresses may be modified on the way to an A register by adding or subtracting one or more B register quantities.

Functional units

There are nine functional units in the computation section of the CPU. Each is a specialized arithmetic unit with algorithms for a portion of the CPU instructions. Each unit is independent of the other units, and a number of functional units may be in operation at the same time. There are not "visible" registers in the functional units from a programming standpoint. A functional unit receives one or two operands from operating registers at the beginning of instruction execution and delivers the result to the operating registers when the function has been performed. There is no information retained in a functional unit for reference in subsequent instructions. These units operate essentially in a three address mode, with very limited source and destination addressing.

All functional units with the exception of the floating multiply and divide units have one clock period segmentation. This means that the information arriving at the unit, or moving within the unit, is captured and held in a new set of registers at the end of every clock period. It is therefore possible to start a new set of operands for unrelated computation into a functional unit each clock period even though the unit may require more than one clock period to complete the calculation. This process may be compared to a delay line in which data moves through the unit in segments to arrive at the destination in the proper order but at a later time. All functional units perform their algorithms in a fixed amount of time. No delays are possible once the operands have been delivered to the front of the unit.

The floating multiply unit has two clock period segmentation. Operands may enter the multiply unit in any clock period providing there was no operation initiated in the preceding clock period. There is a one clock period delay in initiating a multiply instruction if another multiply instruction has just been started.

The floating divide unit is the only functional unit in which an iterative algorithm is executed. There is essentially no segmentation possible in this unit although the beginning of a new operation can overlap the completion of the previous operation by two clock periods.

The table below lists the functional units with the number of clock periods required for execution in each unit. The first column indicates the number of clock periods in each segment. The second column indicates the number of clock periods required to execute the function from the time the operands leave the operating registers until the result arrives back at the operating registers. Each functional unit has a fixed execution time which is independent of its possible modes of operation. For example, a double precision multiply requires the same amount of time as a single precision multiply. The list of octal designators for each functional unit identifies which CPU instructions are performed in that unit.

	<i>segment time</i>	<i>execution time</i>
Long add unit (36, 37)	1 clock period	2 clock periods
Floating add unit (30, 31, 32, 33, 34, 35)	1 clock period	4 clock periods
Floating multiply unit (40, 41, 42)	2 clock periods	5 clock periods
Floating divide unit (44, 45)	18 clock periods	20 clock periods
Boolean unit (10 through 17, 26, 27)	1 clock period	2 clock periods
Shift unit (20, 21, 22, 23, 43)	1 clock period	2 clock periods
Normalize unit (24, 25)	1 clock period	3 clock periods
Population count unit (47)	1 clock period	2 clock periods
Increment unit (50 through 77)	1 clock period	2 clock periods
Branch		2 clock periods (no branch)
		3 clock periods (in stack)
		11 clock periods (outside stack)

Storage field protection

Each object program at execution time has a designated field of SCM and a designated field of LCM in which it may address data. These fields are specified by the monitor program at the time the object program is initiated. Each field may begin at an arbitrary address in storage and continue for an arbitrary length. All addresses in an object program field must be contiguous.

The storage bounds for an object program are contained in four hardware registers in the CPU. These registers are:

(RAS) Reference address for small core memory
This is an 18-bit register which defines the absolute SCM address which is the first address in the SCM field.

(FLS) Field length for small core memory
This is an 18-bit register which defines the length of the SCM field.

(RAL) Reference address for large core memory
This is a 24-bit register which defines the absolute LCM address which is the first address in the LCM field.

(FLL) Field length for large core memory
This is a 24-bit register which defines the length of the LCM field.

All addresses for SCM or LCM contained in the object program code are relative to the reference address which begins the defined field. It is therefore not possible for an object program to read or alter any storage locations with a lower absolute address than the reference address. Each object program reference to storage is checked against the appropriate field length to determine if the address is within the bounds assigned. A storage reference beyond the assigned field length is prevented from altering the storage content and creates an error condition which terminates the program execution.

Program branching

Program branching presents some special situations because of the instruction stack. The current program address is maintained at all times in a program address register (P). This register contains the relative address in SCM for the word currently in the CIW register. When program instruction words are read sequentially from the instruction stack into the CIW register, the P register is advanced one count for each word. The instruction stack contains 12 words of instruction code in a group of registers called the instruction word stack (IWS). Associated with each word in the instruction word stack is an 18-bit address. The 12 addresses in the instruction stack are contained in twelve 18-bit registers called the instruction address stack (IAS). The addresses in the IAS move with the words in the IWS so that the one-for-one relationship between address and program instruction word is maintained as the words move through the instruction stack.

When a program branch point is reached and the current program sequence is terminated with a jump to a new program address, the new program address is entered in the P register. This new program address is then compared with the 12 addresses in the IAS to determine if the jump is within the instruction stack. If a coincidence is found between the new address in the P register and one of the 12 addresses in the IAS, the associated word in the IWS is read immediately into the CIW register and the jump instruction has been executed. If no coincidence is found between (P) and the addresses in the IAS, the jump is "out of stack." In this case a new sequence of instructions must be read from SCM at the new program address (P) and entered in the instruction stack. Completion of the jump instruction is delayed in this case by the amount of time required for the first word to be read from SCM.

The old instruction words in the IWS are not cleared when a jump out of stack occurs. The old words are simply shifted along in the IWS as the new program sequence enters the instruction stack. It is therefore possible to have several sequences of non-contiguous program code in the instruction stack at one time. The program execution may jump back and forth between these program sequences without leaving the instruction stack as long as the current program address (P) does not come within two words of the end of the program sequences held in the instruction stack.

Exchange jump

The CPU exchange jump is a mechanism for switching CPU execution between object program and monitor program. An object program which requires monitor action for a library call, input-output request, or error treatment performs an exchange jump to terminate its own execution and begin the monitor program. Similarly the monitor program performs an exchange jump to initiate execution of a particular object program.

The execution of an exchange jump involves the simultaneous storing of all pertinent information in the CPU operating registers and control registers into SCM, and the reading of new information from SCM into these same registers. This block of data is called an

"exchange package." The execution of an exchange jump then involves the storing of the exchange package for the terminating program and the reading of the exchange package for the initiating program. The information contained in an exchange package is shown in Figure 3.

An "execution interval" for an exchange package is defined in this manual to mean a period of time during which the particular exchange package resides in the CPU hardware registers. The execution interval begins with an exchange jump which reads the exchange package from SCM and enters these parameters in the CPU registers. The execution interval ends with another exchange jump which stores the exchange package back into SCM. The complete execution of an object program may then be composed of a number of execution intervals for the object program exchange package interspersed with monitor activity.

An exchange package contains all of the information necessary to resume the execution of a terminating program. The contents of the operating registers A, B, and X are contained in the exchange package along with the current program address P. The four storage bounds registers RAS, FLS, RAL, and FLL are represented. In addition the following four registers of special information are included in the exchange package.

(NEA) Normal exit address — This is a SCM absolute address for an object program exchange exit instruction.

(EEA) Error exit address — This is a SCM absolute address for an exchange jump or error termination.

(BPA) Breakpoint address — This is a SCM relative address for breakpointing an object program.

(PSD) Program status designation — This is a register of control information.

Program breakpoint

An object program may be executed in small sections during a debugging phase by using the breakpoint address register (BPA). This is a hardware register in the computation section of the CPU which is loaded from the object program exchange package. A coincidence test is made between (BPA) and the program address register (P) as each program instruction word is read from the IWS to the CIW register. When a coincidence occurs the program execution is terminated with an exchange jump to the error exit address (EEA).

SCM location	n	P	A0	BPA
	n + 1	RAS	A1	B1
	n + 2	FLS	A2	B2
	n + 3	PSD	A3	B3
	n + 4	RAL	A4	B4
	n + 5	FLL	A5	B5
	n + 6	NEA	A6	B6
	n + 7	EEA	A7	B7
	n + 8	X0		
	n + 9	X1		
	n + 10	X2		
	n + 11	X3		
	n + 12	X4		
	n + 13	X5		
	n + 14	X6		
	n + 15	X7		

Figure 3 CPU Exchange Package

The monitor program controls the breakpoint address for debugging an object program by altering the exchange package for the object program before each execution interval. The monitor program receives instructions for breakpoint control from an operator console or from control cards in a job stack. It is possible to step through a program one instruction word at a time using an operator console to monitor the register values at each step.

Error exits

Execution of an object program may be terminated by an exchange jump to the error exit address (EEA) under certain conditions. Some of these conditions may be selected by mode declarations through the monitor program, and some are unconditional. In general, errors due to arithmetic overflow, underflow, or indefinite results during computation may be allowed to proceed through the calculation, or may cause an error exit, depending on mode selection. Errors due to hardware failure or program addressing out of an assigned field in storage cause unconditional error exits. In any error exit case the monitor program has the ability to continue the object program where the error can be corrected or ignored.

The error condition flags and mode selection flags are all contained in an 18-bit program status designation (PSD) register. This register is loaded from the exchange package for each object program. The mode selections are made in the exchange package prior to the execution interval by the monitor program. If an error condition occurs during the execution interval the monitor program can determine the type of error by analyzing the terminating exchange package parameters. Each bit in the PSD register has significance either as a mode selection or an error condition flag.

CPU input-output section

The CPU input-output section includes the mechanism to buffer data to (or from) the directly connected PPU. Each PPU communicates with the CPU over a 12-bit full duplex channel. Each channel has assembly and disassembly registers to convert the 12-bit channel data to 60-bit CPU words. The function of the CPU input-output section is to deliver these 60-bit words to the SCM for incoming data, read 60-bit words from the SCM for outgoing data, and interrupt the CPU program for monitor action on the buffer data as required.

The input-output section is able to process a maximum of one 60-bit word each two clock periods. The effective processing rate is somewhat lower than this because of bank storage conflicts in SCM. Whenever a bank conflict occurs on an input-output section request, the communication path to the SCM is held up until the conflict is resolved. Channel requests for a SCM word reference are processed on a priority basis whenever the I/O section is not able to keep up with the channel requests. The priority is assigned in order by channel number, with the lowest order channels having the highest priority.

There are a total of 15 channels in the I/O section of the CPU. These channels are numbered in octal beginning with 01 and ending with 17. Each channel has a SCM buffer area for incoming data and a separate SCM buffer for outgoing data. In addition each channel has an exchange package for incoming data and an exchange package for outgoing data. Each buffer area is divided into two fields, a lower field and an upper field. Data is entered (or removed) from the buffer area in a circular mode. The last word in the lower field is followed by the first word in the upper field. The last word in the upper field is followed by the first word in the lower field. Whenever a buffer area has been filled (or emptied) to the point where a field boundary is crossed, the CPU is interrupted to the associated exchange package to process the buffer data. The channel continues to fill (or empty) the other buffer field while the CPU is processing this buffer data.

The I/O section exchange package areas are permanently assigned in the lowest order addresses of SCM. These areas are arranged as shown in Figure 4. The I/O section buffer areas are assigned in the next higher order address positions of SCM. These areas may be changed both in size and order (wiring change) to accommodate various types of channel volume. A typical arrangement for the buffer areas is shown in Figure 5. Total I/O section space in SCM cannot exceed absolute address 10,000 octal.

Real time clock

CPU programs may be timed precisely by using the CPU clock period counter. This counter is essentially a real time clock which is advanced one count each clock period of 27.5 nanoseconds. Since the clock is advanced synchronously with the program execution, the program may be timed to an exact number of CPU clock periods.

The CPU clock period counter contains a 17-bit register which can be read by a CPU program with an 016 instruction. This register contains the lowest order 17 bits of the real time count. An overflow of the highest order bit in this register sets a real time interrupt flag. This flag reads as an 18th bit on an 016 instruction. In addition, this flag causes an exchange jump to the real time exchange package at absolute address 20 in SCM.

Real time interrupt

The CPU clock period counter causes an interrupt of the CPU program every 3.6 milliseconds (approx.). This corresponds to the modulus of the 17-bit register in the clock period counter. This interrupt causes an exchange jump to absolute address 20 in SCM. The real time exchange package at this SCM address executes a CPU program in monitor mode which advances the count in a 60-bit SCM storage location to continue the real time clock function of the clock period counter. This program also tests time limit for the currently active object program. The real time interrupt flag is cleared at the end of the execution interval for real time exchange package. This flag is read as an 18th bit on an 016

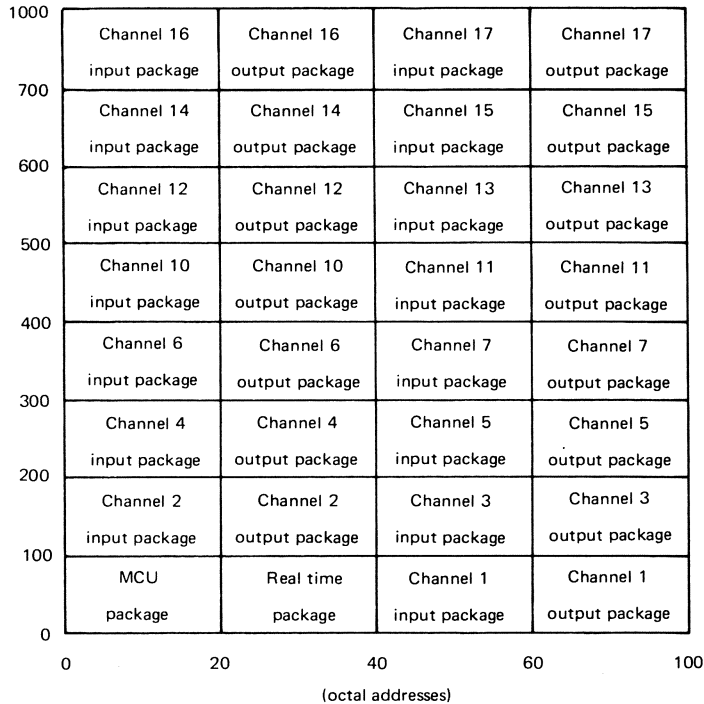


Figure 4 I/O Section Exchange Packages in SCM

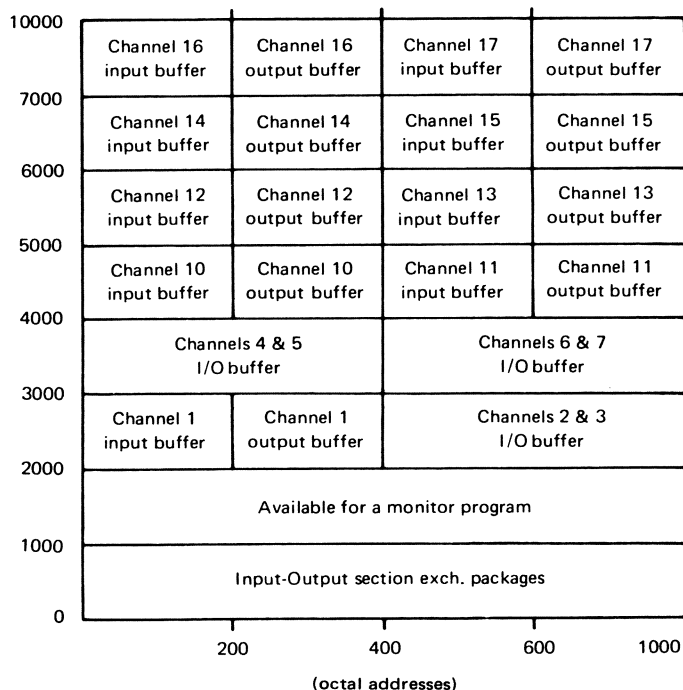


Figure 5 I/O Section Buffer Areas in SCM

instruction in order to avoid erroneous timing indication where the reading of the clock period counter coincided with the setting of the interrupt flag. This bit is an indication that the counter has passed its modulus but the interrupt to advance the SCM word has not yet occurred.

External interrupt

The CPU computation may be interrupted from an external source through the directly connected PPU. Each such PPU has the ability to interrupt the execution of an object program and call the system monitor by sending a control message over the associated channel to the CPU. The interrupt occurs whenever a record flag arrives at the CPU input-output section on an incoming data link. Interpretation of message content is a function of the system monitor program.

System dead start

The system is initially started through the maintenance control unit (MCU). This mechanism is used whenever power is turned on after an idle period or when the system is restarted after hardware failure. The MCU is essentially a PPU with specially adapted input-output channels. This unit is used exclusively for maintenance functions.

The dead start sequence begins with a deck of binary cards for the MCU program. These cards are loaded through the MCU card reader and activate the MCU. The MCU program then dead starts the CPU and all other PPU in sequence. A bootstrap program is entered directly into SCM from the MCU. The CPU program is then initiated by the MCU through the MCU exchange package at absolute address zero in SCM. The MCU program also transmits a resident PPU program to all directly connected PPU to initiate their activity. The system is then loaded completely from a system library device associated with one of the operating stations.

System storage allocation

The operating system software consists of the CPU monitor program, with its overlays, plus the PPU programs to drive the peripheral equipment. The operating system forms a software framework in SCM and LCM to hold the object programs for execution. Organization of CPU storage is illustrated in Figure 6. The I/O section storage areas are fixed by hardware addressing in SCM. The remaining areas are strictly software organization.

The two-level 7600 central memory consists of 65K 60-bit words in the small core memory (SCM) from which the central processor executes, and 512K 60-bit words of linear select large core memory (LCM) which serves as high speed intermediate storage.

SCM is shared by exchange packages, I/O buffers, resident portions of the operating system, and the object program as shown in Figure 6. Of the 512 words that comprise the resident portion of the operating system, 64 words are dynamically overlayed with operating system routines. No overlay is longer than 64 words and can be loaded from LCM in 4 microseconds.

SCOPE 70 LCM resident occupies approximately 65K, including tables, buffers, the FORTRAN compiler, FORTRAN object-time routines, and the system library. The remainder of LCM is used for two job fields: an upper job field and a lower job field. The lower field begins with the first available address and extends upward. The upper field begins with the last available core address and extends downward. Each job field includes:

1. A 4K buffer for the job INPUT file
2. An 8K buffer for the job OUTPUT file
3. An SCM field (length scheduled by user)
4. An LCM field (length scheduled by user)
5. An 8K buffer for each file the user attaches to the job.

The SCM fields in LCM are occupied by the two user programs that are alternately swapped into SCM. The LCM field for the job is accessible by the user program when it is in execution. Although data cannot be preloaded into the LCM field when the job is loaded into the SCM field, a program or a subsequent run of a job can use the field as an unlabeled common area.

SCOPE manages up to three logical mass storage units. Each of these units may consist of multiple physical units and contains up to 512 tracks numbered 0000 to 7777₈. SCOPE maintains track reservation tables in LCM for each unit. Tracks on non-existent units and bad tracks are marked unavailable at system dead start. Other tracks are marked unavailable at allocation and available at release.

For optimum storage usage, all three units are present and have the following functions:

- Unit 1 Contains short files received from I/O stations
- Unit 2 Contains the system library, public files, and overflow from units 1 and 3
- Unit 3 Contains job rollout files

Thus, only unit 2 is required. In the absence of units 1 and 3, their track reservation tables are zeroed indicating their unavailability, and all files are placed on unit 2.

System operation

Job execution proceeds through the system in three phases. In the first phase cards are read at an operating station and an input file is generated on a disk pack, tape unit, or disk file. This input file may physically reside at the operating station or it may reside in a central disk file.

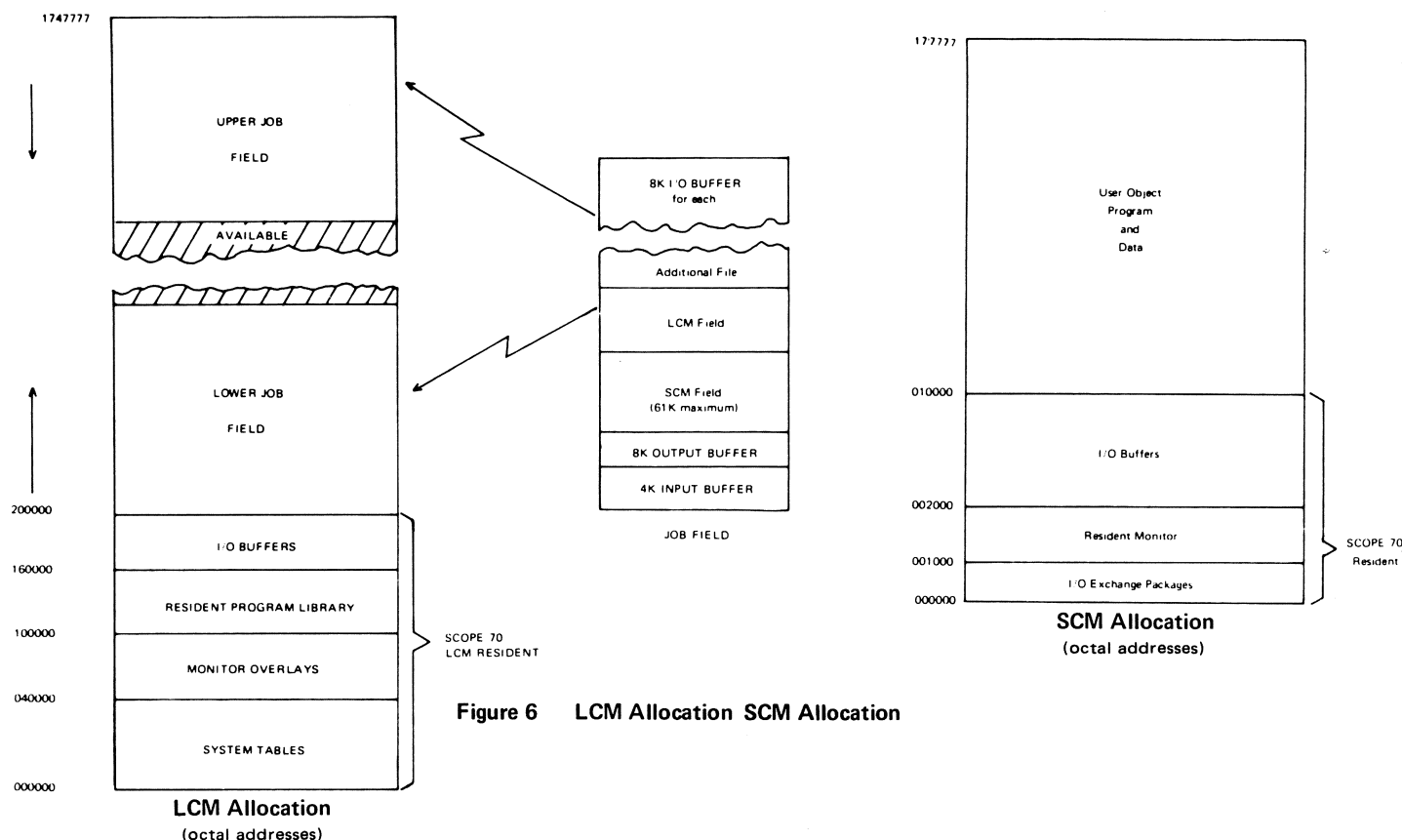


Figure 6 LCM Allocation SCM Allocation

In the second phase the input file is copied into a job field in LCM. If the input file is small the entire file may then reside in LCM. If the input file is large the first portion of the file is copied into a buffer area in LCM. The control cards in the input file are then interpreted by the monitor program and the necessary compiler or library routines are read from outside the LCM if necessary for program execution. When the job is ready for execution the program code is transferred to SCM. If the program and associated data are too large to fit entirely in SCM a portion of the data must be retained in LCM and directly addressed there. This must be done by declaration at compile time in order to designate certain arrays of data to reside in LCM for execution.

Only one program at a time is executed in SCM. The entire SCM object program field may thus be used for each program. Data is read from the input file in LCM and results are stored in an output file in LCM. If the amount of input and output data is small the job may be run to completion in one execution interval. If job execution is delayed by buffer size or by intermediate file references the program code is returned to LCM and another job uses the SCM while buffer data is transferred to (or from) LCM. The second phase is completed when the output data has been delivered to the output file buffer in LCM and this buffer has been emptied onto a system mass storage unit.

The third phase consists of copying the output file from the mass storage to an operating station. During this phase the LCM job field has been released for another job.

System peripherals

The 7638 Disk Subsystem consists of two controllers and one 817 Disk File (two storage units). The 817 Disk File contains two double-ended disk drive spindles that rotate asynchronously at 1,800 rpm, less inductive slip. Each of the four half-spindles

contains 18 disks (32 recording surfaces). There are two hydraulic positioning access assemblies, each one servicing a pair of half-spindles. Each of the two access assemblies is considered to be one file unit. Each access assembly contains two separate horizontally opposed groups of 16 head arms, one group for each half-spindle. Mounted on the end of each head arm are two head pads, each of which contains one read/write magnetic head. Thus, there is one head for each of the 32 surfaces on a half-spindle.

Each unit is divided into four groups of 16 heads each (two groups on each stack). Each access assembly can be moved to any of the 1024 data positions on each stack or to a non-data (retract) position via a position function. There are 512 positions on each of the two stacks. The positioning times are:

Adjacent position	30 msec
Random average position	85 msec
Maximum position	145 msec
Maximum latency	35.5 msec

Disk File Characteristics (approximate figures)

Disk file data rate (average over revolution)	34.3 x 10 ⁶ bits/sec
Number of tracks/disk	512
Sectors per track	40
Bits/head/sector	2,166 bits
Preamble	165 bits
Data	1,935 bits
Post-amble	66 bits
Index	290 bits
Total Available Storage/System	4.78 x 10 ⁹ bits

Each control unit can be operated by two 7600 PPU on a time-shared basis. Each PPU uses two input-output channels (one for data and one for control) to operate the controller. Each read/write control unit operates one disk file storage unit. The two PPU channels are identical, but the signals are defined differently. The control channel transmits output function codes and receives input status codes. After the necessary functions are issued on the control channel, the data transfer takes place on the data channel. The read/write control units do not contain any reservation logic for the two PPU interfaces. Thus, the PPU must reserve control units with software. The software ensures that only one channel of a read/write control unit is operating at one time. The only exception is that status can be read at any time. A 16-bit cyclic parity byte is generated by the controller for the data field of each record written on the disk. The parity byte is written at the end of the record. During a read operation, the parity byte is read and compared to the parity byte generated for the record being read. If the parity does not compare the parity error status bit sets.

Data transfers can be accomplished by writing on alternate sectors. Two PPU can operate one storage unit in a mode where PPU A operates on the even-numbered sectors and PPU B operates on odd-numbered sectors. Thus the transfer rate of one block is twice as fast.

Other 7600 peripherals include card readers and punches, tape transports, line printers, removable disk storage drives, and CRT displays. These will communicate with the system PPU through programmable equipment controllers.

Operating specifications will be comparable or identical to those of 3000 and 6000 Series peripherals.

System configurations

Typically, a 7600 system utilizes a small number of peripheral equipments used primarily for system functions, and a larger number of peripheral equipments grouped as operating stations and used primarily for user functions.

The system-oriented peripheral equipments include a card reader and CRT display reserved for the maintenance function and used in conjunction with the MCU. Additionally, one or more Disk Subsystems and a variable number of PPU are utilized for system-oriented functions; of these, one Disk Subsystem configured as two independent units and four PPU constitute the minimum recommended configuration. The configuration of the operating stations will vary, and is subject to continued refinement as system operating experience accumulates. In general, both general purpose and special purpose stations may be utilized; the latter typified by a tape input-output station operated in conjunction with a centralized tape vault, and the former by the configuration of Figure 7. Operating stations may be located remote from the central complex through use of communications links and interfaces, and when employing programmable equipment controllers, may be utilized for a variety of CPU-independent processing.

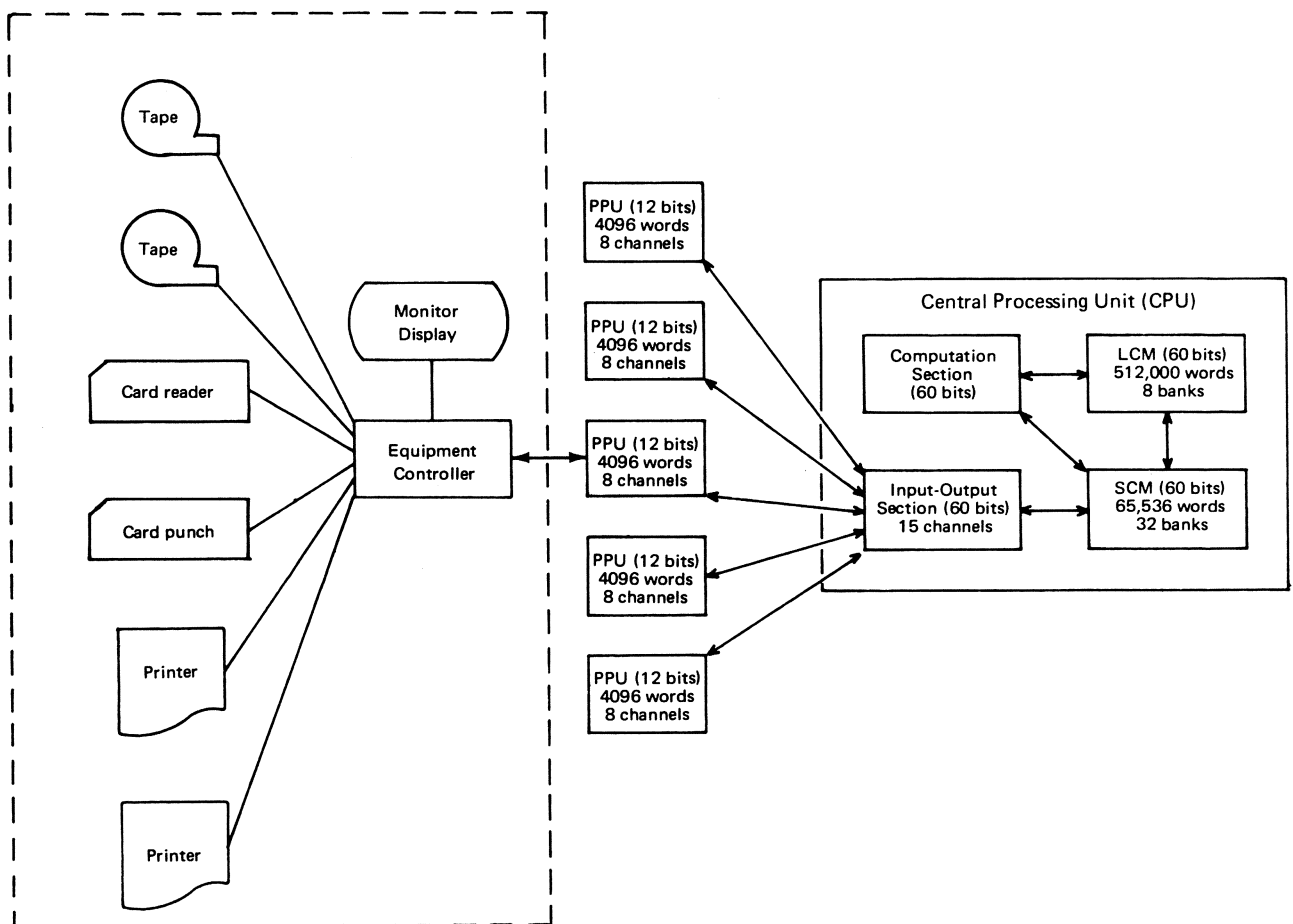


Figure 7 Typical 7600 Operating Station

System software

In general, 7600 software releases will provide source language compatibility with their then current 6000 Series counterparts. Exceptions will be made only in the interest of improved system performance or compliance with evolving language standards.

Product Set releases will provide an increasing variety of user facilities. Facilities will include comprehensive sub-systems for information management, on-line diagnostics, and job or system resource accounting. Additionally, the system will emphasize reliability and include facility for restart and recovery from a variety of hardware and user errors. Specifications and schedules will be published separately, but will be consistent with the following general descriptions and will be oriented toward the use of programable equipment controllers.

The software support for the 7600 System is planned to provide a bridge for early 6000 customers and a viable conversion to a comprehensive system which exploits the powerful architecture of the 7600 System. There will be two basic operating systems — SCOPE 70 and SCOPE 76.

The SCOPE 70 System is intended to meet early customer requirements and provide the base for further system development.

The SCOPE 76 System is intended to meet a comprehensive operating system design to support the broad range of applications for the 7600 System. It will be developed in two phases and referred to as SCOPE 76 Release 1 and Release 2. The Release 1 system will be a selected subset of the comprehensive design and Release 2 will provide the capabilities of the complete design.

The product set associated with SCOPE 70 is:

- FORTRAN
- COMPASS
- LOADER
- 7611 Station
- MODIFY

The product set associated with SCOPE 76 Release 1 is:

- All products under SCOPE 70
- 7611 Station
- Wide Band EXPORT/IMPORT
- COBOL
- SORT/MERGE
- Table Generator
- Report Generator

The product set associated with SCOPE 76 Release 2 is:

- All products under SCOPE 76 Release 1
- MARS VII
- Voice Grade EXPORT/IMPORT
- Communications Station

SCOPE 70 product set

SCOPE 70: SCOPE 70 will be a limited operating system enabling customers with early deliveries to use the 7600. It will be a file oriented system with mono-programming in SCM and all input, output and tape files will be staged on the system disk file. Job initiation will be on a priority basis while job execution will be on a time-slicing basis. Roll-out of jobs resident in SCM may occur based upon the need for I/O or the existence of a high priority job in the input queue. All communications with the operator station will be initiated at the station either manually from the console or on an automatic basis. There will be system dayfile maintained for time accounting purposes.

A recovery capability will be implemented which will consist of restarting jobs in execution, recovering print and punch files currently being processed, and recovering the input and output queues and common files. This recovery presumes that certain system resident areas in SCM and LCM (like File Name Table, File Status Table, Track Reservation Table, etc.) are not destroyed. The only non-7000 tape formats supported will be 6000 SCOPE Version 3 external tapes (x-mode).

COMPASS: COMPASS Assembler provides users with a versatile and extensive symbolic language for generation of object code. COMPASS is designed to facilitate efficient utilization of all computer resources while giving the user maximum flexibility in program construction.

The COMPASS language allows all hardware functions of the 7600 Central and Peripheral Processors to be expressed symbolically. In addition, many pseudo instructions are included which enable the programmer to control the assembly process itself much the same as machine language instructions control the computer. This ability to control assemblies places 7600 COMPASS at a level of sophistication much higher than that of the conventional assembler.

The source program can also include definitions of macro and micro instructions and references to them. The definition of a macro is a group of source statements that are assembled each time the macro is referenced. An extension to the macro facility permits user definition of instructions having machine language formats. A micro is a named character string that the assembler substitutes for a reference to the name.

COMPASS executes under control of the SCOPE 70 Operating System, and will operate on all configurations supported by SCOPE 70. COMPASS is called from the system library by control card parameters on the card specify files to be used by the assembler. COMPASS executes as a Central Processor program and produces a binary file used by the LOADER to establish an executable program in SCM. COMPASS also produces a coded file which is normally printed by SCOPE 70. This file contains a listing of each source statement and object code generated if any. Pseudo instructions provide the programmer with the option of obtaining a variety of detailed or brief listings with or without special titles on each page. Errors detected by the assembler are noted on the listing. A cross reference table will also be produced if requested.

Features included in COMPASS:

- free-field source statement format
- selective assembly of code sequences
- control of local and common blocks
- address arithmetic
- preloaded data
- library routine calls
- data definition
- symbol equation and redefinition
- macro coding
- micro coding
- code repetition
- thorough language diagnostics
- extensive listing control
- remote assembly
- concatenation

FORTRAN: FORTRAN is an improved language for the expression of procedures in carrying out scientific computations. It includes the features implicit in the latest revisions of the USASI FORTRAN language, and in addition includes some features developed exclusively by Control Data Corporation. The most important of these features is the facility for controlling the

buffering of input and output with checking in the same sense that it can be controlled by the machine language coder. "ENCODE" and "DECODE" statements are provided to control the conversion of core storage data according to a format statement. NAMELIST (a powerful, flexible I/O feature) is also incorporated.

Three new source statements to exploit the unique architecture of the 7600 have been added to the compiler. They are: LARGE to allocate space in LCM, SMALLIN to produce in-line code to transfer a block of data from LCM to SCM, and SMALLOUT to produce in-line code to transfer a block of data from SCM to LCM.

The FORTRAN compiler operates as an independent program under control of the SCOPE 70 Operating System and can be called to use only the storage required for the compilations to be in progress concurrently using the SCOPE 70 Operating System multiprogramming features. FORTRAN accepts main programs and subprograms written either in FORTRAN source language or COMPASS assembly language, as well as binary decks of previously compiled subprograms, which are linked together and run as a single program. A program compiled under the FORTRAN will be preserved in disk storage as a binary file which may then be called and executed repeatedly.

SCOPE 70 contains a powerful relocatable loader, which adds great capability to FORTRAN. Subprograms may be assembled or compiled independently and then brought together prior to execution. Subprograms may also be written and combined in the form of overlays, which permit loading for execution with minimal control. Overlays allow the user to subdivide a task and write portions of the code on a file in absolute form. These overlays can then be loaded at execution time without necessitating a relocatable loading operation. The resident loader for overlays is therefore substantially reduced in size and may be more easily retained with the job for subsequent loading. The utilization of these features is governed by control cards, loader, requests, and a standard subprogram format.

7611 STATION: The 7611 Station is capable of driving the following fixed peripheral hardware configuration for SCOPE 70 Operating System. In addition to the programmed controller, the station components are:

- 1 - Disk Drive
- 1 - Console Display
- 1 - Dead Start via 407-1

Station Peripherals

- 1 - 407-1 Card Reader - 1200 CPM
- 1 - 417-1 Card Punch - 250 CPM
- 2 - 517 Line Printer - 1200 LPM
- 2 - 617 Magnetic Tape Units

Features and capabilities of the 7611

1. The Station will have the capability to request and display status of jobs submitted to central for execution.
2. The Station will impose a maximum physical record size of 5120 characters. The 7000 tape format uses 10 characters for control purposes. This provides 5110 characters to the user.
3. Station control information may be submitted either from the card reader or the display console.
4. The Station will perform error detection and provide limited restart capability.
5. The Station will direct drive the magnetic tape unit, card reader, train printer, disk drive, and the display console.
6. The Station will allow for 150/inch second magnetic tape units at 200, 556, and 800 BPI density.

7. Magnetic tape files will be written in either binary or external BCD.
8. Card Read/Card Punch information may be presented in either binary or Hollerith code (Control Data Scientific Set).

SCOPE 76 product set

SCOPE 76: SCOPE 76 will implement a comprehensive system design for the 7600. Data Manager functions will be embedded within this system and the associated products will make full use of the extended capabilities. Time-critical response times while running standard batch processing will become possible under the scheme for multi-programming in SCM.

A rigorous error message format to be used extensively for both hardware and software errors will be incorporated with the operating system as well as all of the associated products. Based on an installation parameter, all or certain of these messages will be logged in sequence by time of day on a system file which also contains the regular dayfile entries. Various jobs can then process this file to obtain information concerning what has occurred in the system regarding any piece of equipment or software product. These jobs may be user initiated as well as diagnostically initiated from the MCU.

Recovery capabilities will be greatly expanded over SCOPE 70. Job initiation will occur on a priority basis while execution will be performed using time slicing. In addition, multi-programming will be performed in SCM. The dayfile accounting facilities will be similar to that implemented for the 6000 Series SCOPE Operating System and a comprehensive edit and library building routine will be incorporated.

DATA MANAGER: The DATA MANAGER will provide centralized organizing, identifying, storing, retrieving and cataloging of data processed by the 7600 System. It will provide the interface between all programs and the operating system for I/O and related operations.

COBOL: COBOL satisfies full USASI requirements and in addition contains the majority of features from the high levels of the following modules:

NUCLEUS
TABLE HANDLING
SEQUENTIAL ACCESS
RANDOM ACCESS
SORT
REPORT WRITER
SEGMENTATION
LIBRARY

The product runs under control of SCOPE 76 and utilizes the DATA MANAGER as an input/output vehicle.

SORT/MERGE: The SORT/MERGE program is a flexible set of sort and merge routines capable of running in a multiprogramming environment. It can be used by the COBOL SORT VERB, or a stand-alone control card system or by programs assembled in COMPASS.

TABLE GENERATOR: The TABLE GENERATOR is a specialized, batch-oriented, information retrieval program capable of tabulating large amounts of numeric input data.

The TABLE GENERATOR retrieves data from sequentially organized input data into tables on an output file. The user describes the characteristics of the input and output files on parameter control cards via a simple TABLE GENERATOR language. The program extracts and sorts only the data specified by the user and is capable of producing up to two hundred tables from a single pass of an input data file comprising fixed or variable length records.

A table, within the context of the TABLE GENERATOR concept, may be regarded as a multi-dimensioned array. Each cell in a table is characterized by its position within the table (INDICATIVE) and by its value (TALLY). The INDICATIVE is the sort-merge key. The TALLY is the count of the occurrences of data identified by the INDICATIVE. The INDICATIVE and the TALLY are each one computer word in length. An index option allows the user to circumvent the insertion of zero TALLY values in an output table which has relatively few non-zero TALLY values.

The TABLE GENERATOR is well-suited to creating statistical tables, but has applications to a wide variety of problems. A specific application is the production of large, complex statistical tables from large files of census data.

REPORT GENERATOR: The REPORT GENERATOR permits the user to describe a report pictorially. It provides the user with features to describe the physical layout, the position and format of individual data items, and the logical grouping of the report.

The product provides a convenient method for describing the

characteristics of an input file and reports which may be generated from such a file. Features provided include:

1. Formatting and editing of report lines.
2. Incrementing and listing line counters for page or overflow conditions, and producing headings and footings as required.
3. Checking for control breaks and producing the required report lines.
4. Generating final report lines if end-of-file encountered on an input file.

WIDE BAND EXPORT/IMPORT: The WIDE BAND EXPORT/IMPORT product provides stations access to the 7600 central system via wide band communications link. This product will provide the capability for remote stations to transmit tasks to and receive output from the 7600 central system at high speeds. The features of the product are:

- Job submission and input/output formats identical to those used on local stations.
- Operator control over jobs identical to that of a local station operator through the display console.
- USASCII standard communication formats across communication links.
- Accommodates 40.8K bit or 50.0K bit per second lines.
- Full transmission error checking.

Additional products

Additional software products will be announced and specified at a later date. These will include interactive terminal access facilities, mathematical and simulation languages, PERT, APT, and other technical applications packages.

COMMENT SHEET

MANUAL TITLE CONTROL DATA 7600 Computer System

Preliminary System Description

PUBLICATION NO. 60258400 REVISION

FROM: NAME:

BUSINESS
ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

PRINTED IN U.S.A.

AA 3419

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

8100 34TH AVENUE SOUTH

MINNEAPOLIS, MINNESOTA 55440

**ATTN: TECHNICAL PUBLICATIONS DEPT.
PLANT TWO**

FOLD

FOLD

CUT ALONG LINE