Systems

IBM System/370 Principles of Operation

The IBM System/370 is a data processing system that is based on the IBM System/360 but that extends the capabilities of that system. This manual describes extensions to the functional design of the System/360 that are incorporated in models of the System/370.

The reader is assumed to have an understanding of the System/360, as described in the *IBM System/360 Principles of Operation,* GA22-6821. The *System/370 Princples of Operation* should be used in conjunction with the *System/360 Principles of Operation.*

For information about the characteristics, functions, and features of a specific System/370 model, use the functional characteristics manual for that model.

IBM

Preface

Just as the *IBM System/360 Principles of Operation* is the machine reference manual for the IBM System/360, this publication is the machine reference manual for the IBM System/370. It deals, however, only with those functions that are additional to or different from those of the System/360, and the reader therefore must rely directly on the *IBM System/360 Principles of Operation*, GA22-6821

Note that this publication is not written as an introduction or as a textbook but as a reference document. Other manuals are available that serve those other functions. It assumes the reader has a basic knowledge of data processing systems and, specifically, the IBM System/360. Such basic information can be found, for example, in the *Introduction to IBM Data Processing Systems,* GC20-1684, and in the *IBM System/360 System Summary,* GA22-6810.

The information in this manual is aimed most directly at those readers concerned with the preparation of programs to be run on the System/370, where in such preparation a knowledge of the machine language and system functions is pertinent. At the same time, the book contains information that may be of particular interest to system designers.

First Edition (June 1970)

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

This manual has been prepared by the IBM Systems Development Division, Product Publications, Dept. B98, PO Box 390, Poughkeepsie, N.Y. 12602. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be sent to the above address.

Contents

REMOVAL OF USASCII MODE

System/360 provides for the USASCII-8 code by a mode under control of PSW bit 12. When bit 12 of the PSW is one, the codes preferred for the USASCII-8 are generated for decimal results. When PSW bit 12 is zero, the codes preferred for EBCDIC are generated.

In System/370, the USASCII mode and the associated meaning of PSW bit 12 are removed. All instructions whose execution in System/360 depends on the setting of PSW bit 12 are executed in the System/370 EBCDIC mode.

When bit 12 is a one, a program interruption for specification exception occurs.

HANDLING AN INVALID DECIMAL SIGN

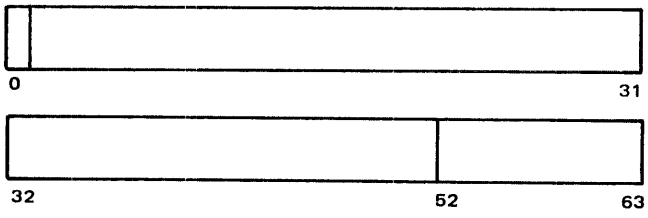
In System/360 an invalid decimal operand causes the operation to be terminated. In System/370, the operation is suppressed, instead of terminated, when an invalid sign is detected. The action applies to all instructions that check the validity of decimal operands: ADD DECIMAL, SUBTRACT DECIMAL, ZERO AND ADD, COMPARE DECIMAL, MULTIPLY DECIMAL, DIVIDE DECIMAL, and CONVERT TO BINARY. It includes also the new instruction SHIFT AND ROUND DECIMAL. The action is:

Sign Code	*Digit Code*	*Instruction Execution*
Valid	All valid	Completed
Valid	One or more invalid	Terminated
Invalid	All valid	Suppressed
Invalid	One or more invalid	Suppressed

Time-of-Day Clock

The time-of-day clock provides a consistent measure of elapsed time suitable for time-of-day indication. The cycle of the clock is approximately 143 years.

The time-of-day clock is a binary counter with this format:



The bit positions of the clock are numbered 0 to 63, corresponding to the bit positions of a fixed-point number of double precision. Time is measured by incrementing the value of the clock, following the rules for fixed-point arithmetic.

The resolution of the time-of-day clock is one microsecond, and the clock is incremented by adding a one in bit position 51 every microsecond.

The program is not alerted to the overflow as the clock value changes from a large positive value to a large negative value because of a carry from bit position 1 into bit position 0. No interruption condition is generated as a result of the overflow. When incrementing the clock causes a carry to be propagated out of bit position 0, the carry is ignored, and counting continues from zero on.

The clock can be inspected by using the STORE CLOCK instruction, which causes the current clock value to be stored in main storage. The clock can be set to a specific value by SET CLOCK, which causes the current clock value to be replaced by the operand designated by the instruction. The SET CLOCK instruction is executed only when the clock security switch on the operator intervention panel is set to permit changing the value of the clock.

The operation of the clock is not affected or inhibited by any normal activity or event in the system. The clock runs when the CPU is in the wait or stopped state or in the instruction-step, single-cycle, or test mode. Its operation is not affected by system reset or the IPL procedure. Depending on the implementation, the clock may or may not run with CPU power off.

Three states are distinguished for an operational clock: set, not-set, and error. The state determines the condition code set by STORE CLOCK.

When the power for the clock is turned on, the value of the clock is set to zero, and the clock enters the not-set state.

The clock enters the set state when SET CLOCK causes the clock's contents to be set, that is, when SET CLOCK is executed without encountering any exceptions and with the clock operational and the clock-security switch in the enabled position. The clock can be placed in the set state from either the not-set or the error state.

The clock enters the error state whenever the clock stops or misses a time increment, such as may occur when the power supply is temporarily disconnected, or when a malfunction is detected that is likely to have affected the validity of the clock's value.

The operation of the clock is interlocked such that successive executions of STORE CLOCK do not provide the same clock value unless the clock validity indicator is off or the clock value has been changed in between by execution of SET CLOCK.

The control registers provide for loading and storing control information.

Two instructions, LOAD CONTROL and STORE CONTROL, are provided. LOAD CONTROL furnishes a means of loading control information from main storage into control registers; STORE CONTROL permits information to be transmitted from control registers to main storage. These instructions operate in a manner similar to LOAD MULTIPLE and STORE MULTIPLE.

The structure provides for sixteen 32-bit registers for control purposes. These registers are not part of addressable storage.

One or more specific bit positions in control registers are assigned to each function requiring register space. A model may, however, furnish more control register bit positions than required by the installed function. For example, a model may provide register positions for more channel mask bits than the number of channels installed. It depends on the model and on the installed functions whether additional unassigned control register bit positions are furnished.

Only the general structure of the control register function is described here; a description of the assignment and meaning of register positions appears with the description of the function with which the register position is associated. Figure 1 lists the functions to which each control register field belongs.

Word	Bits	Name of Field	Function	Value after Reset
0	0	Block-Multiplex Mode	Block-Multiplexing Control	0
0	24	Timer Mask	Extended External Masking	1
0	25	Interrupt Key Mask	Extended External Masking	1
0	26	External Signal Masks	Extended External Masking	1
2	0-31	Channel Masks	Extended I/O Masking	1
14	0	Hard Stop Mode	Machine-Check Handling	1
14	1	Synchronous MCEL* Mask	Machine-Check Handling	1
14	2	I/O Extended Logout	Machine-Check Handling	0
14	4	Recovery Report Mask	Machine-Check Handling	0
14	5	Configuration Report Mask	Machine-Check Handling	0
14	6	External Damage Report Mask	Machine-Check Handling	1
14	7	Warning Mask	Machine-Check Handling	0
14	8	Asynchronous MCEL Mask	Machine-Check Handling	0
14	9	Asynchronous Fixed Log Mask	Machine-Check Handling	0
15	8-31	MCEL Pointer	Machine-Check Handling	512

Note: Control register positions not listed are unassigned.
* MCEL: Machine-check extended logout.

Figure 1. Allocation of Control Register Fields

EXTENDED EXTERNAL MASKING

Extended external masking provides for selective masking of interruptions due to timer, key, and external signals 2–7. It consists of three subclass mask bits in control register 0, defined as:

Timer Mask (T): Bit 24 of control register 0 controls whether the CPU is enabled for an interruption due to the timer. An external interruption due to the timer value becoming negative can occur only when both the external mask bit in the PSW and the timer mask bit in control register 0 are one. The bit is set to one by system reset.

Key Mask (K): Bit 25 of control register 0 controls whether the CPU is enabled for interruption due to a signal from the key. An external interruption due to pressing the interrupt key can occur only when both the external mask bit in the PSW and the key mask bit in control register 0 are one. The bit is set to one by system reset.

Signal Mask (S): Bit 26 of control register 0 controls whether the CPU is enabled for interruption by external signals 2–7. An external interruption due to the presence of external signals 2–7 can occur only when the external mask bit in the PSW and the signal mask bit in control register 0 are one. The bit is set to one by system reset.

EXTENDED I/O MASKING

Extended I/O masking provides for selective masking of each channel. It provides a set of bits in control register 2 starting at bit position 0 and extending for as many contiguous bit positions as the number of channels supplied. The assignment is such that a bit is assigned to the channel with an address equal to the position of the bit in control register 2.

The channel mask bits in control register 2 for all available channels are set to one by system reset. The state of channel mask bits for unavailable channels is unpredictable. Interruptions from channels 6 and up are controlled by the I/O mask bit (PSW bit 6) in conjunction with the corresponding channel mask bit: the channel can cause an interruption only when the I/O mask is one and the corresponding channel mask is one. Interruptions from channels 0–5 are controlled by channel masks 0–5 in the PSW. Bits 0–5 in control register 2 do not participate in controlling I/O interruptions; they are, however, preserved in the control register.

The machine-check interruption furnishes a means of reporting equipment malfunction and certain external disturbances, and it supplies the program with information about the location and the nature of the cause. In some cases, depending on the nature of the malfunction, the system may either take correcting action or circumvent the failing components.

MACHINE-CHECK CONDITIONS

Equipment malfunctions and other conditions responsible for machine-check interruptions are called machine-check conditions. Machine-check conditions are subdivided into several subclasses.

An equipment malfunction resulting in the loss of integrity of a process in the system is called a damage condition. Damage conditions are divided into five subclasses identifying the process affected: system damage, instruction-processing damage, timer damage, time-of-day clock damage, and external damage. Malfunctions that cannot be isolated to a specific process are indicated as system damage.

An equipment malfunction that is successfully corrected or circumvented without loss of system integrity is called a recovery condition. Depending on the model and the type of malfunction, some recovery conditions may be discarded. Recovery conditions are grouped into one subclass called system recovery.

Machine-check conditions not directly related to equipment malfunctions are called alert conditions. The alert conditions contain two subclasses: automatic configuration and warning.

System damage and instruction-processing damage conditions may result in termination of the current instruction or cause interruptions to be lost, and are referred to as hard machine-check conditions. All other subclasses of machine-check conditions are called soft machine-check conditions.

HANDLING OF CONDITIONS

In normal operation, the system is assumed to have correct parity in storage and registers. A machine-check condition is generated whenever instructions or data with invalid parity are fetched and processed or whenever invalid parity on the protection key or the key in storage makes it impossible to establish reliably whether protection applies to the particular type of storage reference. When instructions or data with invalid parity are fetched but are not used, the program is not necessarily alerted. Similarly, a machine-check condition is not necessarily indicated when the key in storage on a fetch-type reference has bad parity but protection does not apply because the system does not have fetch protection installed.

Once valid information has been placed in storage and in registers, a machine-check condition can be caused only by machine malfunction and never by logically invalid data or instructions. Specification of an unavailable system component, such as a storage unit, channel, or I/O device, does not cause indication of equipment malfunctioning. Instead, such an error causes the appropriate program or I/O interruption condition or sets the condition code.

Depending on the model and type of error, a malfunction detected during an I/O operation may cause a machine-check condition or may result in the placing of identifying information in the I/O status information associated with the operation, or both. When a CCW or data with invalid parity are fetched from storage but are not used in the I/O operation, it depends on the model and the type of error whether equipment malfunctioning is indicated. When an equipment malfunction occurring in an I/O operation is not treated as a machine-check condition, indication of the error to the program and the logging of diagnostic information are not subject to the machine-check mask, PSW bit 13.

CPU RECOVERY ACTION

Two basic techniques, bit correction and retry, are used for recovery from machine-detected malfunctions.

Bit-Correction Capability

Within certain units of the system, a bit-correction capability is provided by either appending additional check bits to a group of bytes or by converting the check bits of a group of bytes into an arrangement which provides for error checking and correction (ECC). The group of bytes associated with a single ECC code is called an ECC block. The number of bytes in an ECC block, and the manner in which the conversion or appending is accomplished depend on the type of unit involved and may vary among models.

When ECC is provided, a reference which results in a single-bit error in the ECC block is automatically corrected.

The automatic correction of a single-bit error causes: (1) the transmittal of correct information to the requesting area, and (2) possibly an attempt to correctly restore the information in the location from which the data was obtained. When the cause of the error does not affect the restore operation, the contents of the location will be correct. If the fault causes the restore operation to return the data with a single-bit error, a subsequent readout of that location will correct the error only if no other bit error is encountered. Thus, a persistent error, as well as a one-time occurrence, is corrected when only one bit is affected.

When multiple-bit errors are detected the information is not corrected. The condition of the check bits on the information transmitted to the requesting area depends on the model. Correction of multiple-bit errors requires special procedures and is described in "Storage Validation."

CPU Retry

In models with machine-retry capability, information about the state of the machine is saved periodically. The logical point in the processing to which this saving of information pertains is called a hardware checkpoint. When an error occurs, recovery is attempted by returning the machine state to that existing at the latest hardware checkpoint and proceeding from that point. The interval between checkpoints is model-dependent. In some cases, several checkpoints are established within a single instruction; in others, checkpoints are established only at the beginning of instructions, or even less frequently.

If, after a restart at the checkpoint, the machine error does not reoccur or is circumvented, a recovery condition is recognized.

When, after repeated restarts at the checkpoint, the machine cannot proceed beyond the point of error, a hard machine-check condition is recognized.

Point of Interruption

Due to the checkpoint capability the interruption resulting from a hard machine-check may occur at a point in the recovery cycle which is prior to the error. Additionally, due to the nature of the malfunction, it is possible that not all of the status reflects the same instant. For example, the general registers modified by a BXLE instruction may have been updated, but the instruction address in the current PSW still points to the beginning of the instruction; or, during the execution of a LPSW instruction, only a portion of the current PSW may have been replaced. As a result, the model may have some choice as to which point in the recovery cycle the interruption is taken, and in some cases, the status which can be indicated as valid depends on the point chosen.

The point during processing at which an error occurs is called the point of error. The point in the processing that is used as a reference point by the machine to determine and indicate the validity of the status stored when the machine-check interruption occurs is called the point of interruption.

Only certain points in the processing may be used as a point of interruption. For soft machine-check conditions, the point of interruption does not precede the point of error; the point of interruption must be after one instruction is completed, including the associated program or SVC interruption, if applicable, and before the next instruction is begun. It may also be at one of the interruption points of an interruptible instruction. In some cases, the point of interruption may be several instructions past the point of error.

Hard machine-check conditions that are delayed (disallowed and presented later when allowed) can occur only at the same points of interruption as soft machine-check conditions. When hard machine-check conditions are not delayed, the point of interruption may occur at two additional places:

1. The interruption point immediately preceding the point of error. In this case, a special bit, called the backup bit, is set in the interruption code to indicate that the status reflects a state prior to the error. When the backup bit is one and all status is indicated as valid, the machine has successfully returned to a point prior to the error, and no additional damage has occurred. This situation is indicated as damage condition rather than system recovery because the malfunction has not been circumvented and damage would have occurred if the process had continued.
2. The point after the instruction is completed but before the associated program or SVC interruption occurs. In this case, a valid PSW is defined as that which would have been stored in the old PSW for the program or SVC interruption. Even though all status may be indicated as valid, damage has occurred because the associated interruption is lost.

When a hard machine-check condition occurs, the point of interruption that is chosen affects the amount of damage that must be indicated. An attempt is made, when possible, to minimize the damage that must be indicated. In general, the order of preference is in the sequence beginning at the interruption point immediately preceding the point of error. When a point of interruption must be chosen, which is after an associated program or SVC interruption, the damage cannot be isolated to the point prior to the program or SVC interruption, and system damage must be indicated.

When the status information stored as a result of a hard machine-check condition does not all reflect the same logical point, an attempt is made when possible to choose the point of interruption so as to make the instruction address which is stored in the machine-check old PSW valid.

CHECK STOP STATE

When the CPU is in the check stop state, the condition is appropriately indicated by an error light and/or audible signal. The exact indication is model-dependent. The system light is off, and the state of the manual light depends on the model.

The machine enters the check stop state only as the result of machine malfunction. The machine may be removed from the check stop state by system reset or IPL.

When the CPU is in the check stop state, instructions and interruptions are not executed. The timer is not updated, and channel operations may be suspended. The CPU cluster meter does not run, and the clock-out and metering-out lines are down. The stop key and start key are not operative during this state.

MACHINE-CHECK LOGOUT

The storing of model-dependent information into main storage as a result of, or in association with, a machine error is called a machine-check logout. When a machine-check logout occurs during the machine-check interruption it is called synchronous. If a machine-check logout occurs without a machine-check interruption, or if the logout and the interruption are separated by instruction processing or by instruction retry, then the logout is called asynchronous. Machine-check logout can occur in either of two areas. The 96-byte area, starting at location 256, is called the fixed-logout area. In addition, a machine-check extended-logout area (MCEL) is defined. The starting location of the MCEL area is specified by the contents of control register 15. The length of the MCEL area depends on the model.

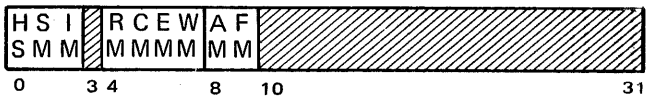
To preserve the initial machine error conditions, some models perform an asynchronous logout before invoking automatic CPU recovery action. In some models the diagnostic information is either stored before recovery or afterwards, but not at both times. On other models both may occur either into the same portion of the logout area or into different portions.

Machine-check logouts are referred to as four different types: synchronous fixed logout (SFL), asynchronous fixed logout (AFL), synchronous machine-check extended logout (SMCEL), and asychronous machine-check extended logout (AMCEL).

MACHINE-CHECK CONTROL REGISTERS

Control register 14 contains mask bits which specify whether certain conditions can cause machine-check interruptions, and it also contains mask bits to control the conditions under which a logout may occur. Bits 3 and 10–31 are reserved.

Control Register 14



Hard Stop

Hard Stop (HS): Bit 0 of control register 14 controls the system action taken when a hard machine-check condition occurs during a hard machine-check interruption or with PSW bit 13 zero. If the hard stop bit is one, the machine enters the check stop state; if the hard stop bit is zero, the machine may attempt to continue or may enter the check stop state, depending on the type of error and the model. The hard stop bit is set to one by system reset. If damage occurs to control register 14 the hard stop bit is assumed to be one.

Extended Logout Masks

Synchronous Machine-Check Extended Logout Mask (SM): Bit 1 of control register 14 controls the logout action during a machine-check interruption. If the bit is one, the machine-check extended logout area may be changed during the interruption; if the bit is zero, the area may be changed only under control of the asynchronous machine-check extended logout mask bit (bit 8 of control register 14). Bit 1 of control register 14 is set to one by system reset.

Input/Output Extended Logout Mask (IM): Bit 2 of control register 14, when one, permits the channel to log out into the I/O extended logout area as part of an I/O interruption. When the I/O extended logout mask is zero, I/O extended logouts cannot occur. This bit is set to zero by system reset.

Asynchronous Machine-Check Logout Mask (AM): Bit 8 of control register 14, in conjunction with PSW bit 13, controls asynchronous change of the machine-check extended logout area. When this bit and PSW bit 13 are both one, the machine may change the machine-check extended logout area at any time. This bit is set to zero by system reset.

Programming Note: The maximum logout information is obtained by setting both the synchronous and asynchronous machine-check extended logout mask bits to one. Both of these bits must be zero to prevent any changes to

the machine-check extended logout area. When AMCEL is allowed, use of the MCEL area for other than logout may produce unpredictable results.

Asynchronous Fixed Logout Mask (FM): Bit 9 of control register 14, when one, permits the fixed logout area to be changed at any time. When this bit is zero, the fixed logout area may be changed only during a machine-check interruption or during an I/O interruption. This bit is set to zero by system reset.

Programming Note: When the AFL mask bit is one, program use of the fixed logout area should be restricted to the fetching of data from this area. Store operations on channel programs reading into the fixed logout area may cause machine checks on undetected errors if the store occurs during CPU retry. *Note that this is an exception to the rule that programming errors do not cause machine-check indications.*

Machine-Check Subclass Masks

Bits 4–7 of control register 14, in conjunction with PSW bit 13, control various machine-check subclass conditions. When PSW bit 13 is one and the subclass mask is one, the associated condition initiates a machine-check interruption. If the subclass mask is zero, the associated condition does not initiate an interruption, but the condition may be presented with another condition which initiates the interruption. All conditions presented are then cleared.

Recovery Report Mask (RM): Bit 4 of control register 14 controls recovery conditions. This bit is set to zero by system reset.

Configuration Report Mask (CM): Bit 5 of control register 14 controls automatic configuration conditions. This bit is set to zero by system reset.

External Damage Report Mask (EM): Bit 6 of control register 14 controls the following machine-check conditions: timer damage, time-of-day clock damage, and external damage. This bit is set to one by system reset.

Warning Mask (WM): Bit 7 of control register 14 controls all warning conditions. This bit is set to zero by system reset.

Summary of Machine-Check Masking

Machine-check masking is summarized in Figures 2, 3, and 4.

Subclass Condition		Mask	Action when CPU Disabled for Subclass Condition	
			Hard Stop Bit = 0	Hard Stop Bit = 1
SD	System Damage	13	P*	Check Stop
PD	Instruction Processing Damage	13	P*	Check Stop
TP	Timer Damage	13 and EM	P*	P*
CD	Time-of-Day Clock Damage	13 and EM	P*	P*
SR	System Recovery	13 and RM	D	D
ED	External Damage	13 and EM	P*	P*
AC	Automatic Configuration	13 and CM	P	P
W	Warning	13 and WM	P	P

Action Code Definition

P Indication held pending.

D Indication may be held pending or may be discarded.

* System integrity is undependable.

Figure 2. Machine-Check Condition Masking

PSW Bit 13	SMCEL Mask	AMCEL Mask	Machine-Check Extended Logout Action
0	X	X	No MCEL
1	0	0	No MCEL
1	1	0	MCEL may occur only during machine-check interruption.*
1	0	1	MCEL may occur at any time.**
1	1	1	MCEL may occur at any time.

AFL Mask	Fixed Logout Action
0	CPU portion of fixed logout area may be changed only during machine-check interruption.*
1	Any portion of fixed logout area may be changed at any time.

Notes:

X Indicates the same action occurs whether the bit is zero or one.

* Logout prior to instruction retry is not permissible in this state even though recovery reports are allowed.

** In some models the AMCEL mask bit is ignored, and no logout occurs in this state.

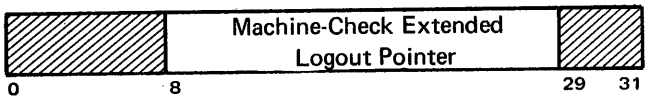
Figure 3. Machine-Check Logout Masking

Bit Description		Control Register 14 Bit Position	State of Bit on System Reset
HS	Hard Stop	0	1
SM	Synchronous MCEL Mask	1	1
IM	IOEL Mask	2	0
RM	Recovery Report Mask	4	0
CM	Configuration Report Mask	5	0
EM	External Damage Report Mask	6	1
WM	Warning Mask	7	0
AM	Asynchronous MCEL Mask	8	0
FM	Asynchronous Fixed Logout Mask	9	0

Figure 4. Machine-Check Control Register Bits

Machine-Check Extended Logout Pointer

Control Register 15



Bits 8–28 of control register 15, with three low-order zeros appended, specify the starting location of the machine-check extended logout area. Bits 0–7 and 29–31 are reserved. The contents of control register 15 are set to 512 (decimal) by system reset.

INTERRUPTION ACTION

A machine-check interruption causes the PSW reflecting the point of interruption to be stored as the old PSW at location 48; extended machine-check interruption information is stored, consisting of the information in all the control registers, general registers, and floating-point registers, a region code, and a failing storage address. Then the machine-check interruption code (MCIC) of eight bytes is stored. A new PSW is fetched from location 112. Additionally, sometime before the storing of the machine-check interruption code, one or several machine-check logouts may have occurred. If the machine-check interruption code cannot be stored successfully or the new PSW cannot be fetched successfully, the CPU enters the check stop state if the hard stop bit is one.

A machine-check interruption due to a soft machine-check condition can occur only when both PSW bit 13 and the associated subclass mask are one. A soft machine-check interruption does not terminate the execution of the current instruction; the interruption is taken after the execution of the current instruction has come to its normal ending and after the associated program or supervisor-call interruption, if any, has been taken. No program or supervisor-call interruptions are eliminated. When the soft machine-check condition occurs during the execution of a system function, such as a timer update, the machine-check interruption takes place after the system function has been completed.

A machine-check interruption due to a hard machine-check condition can occur only when the machine-check mask, PSW bit 13, is one. The interruption terminates the execution of the current instruction and may eliminate the program and supervisor-call interruptions, if any, that would have occurred as a result of the instruction. Proper execution of the steps, including the storing of the old PSW and diagnostic information, depends on the nature of the malfunction. When a hard machine-check condition occurs during the execution of a system function, such as a timer update, the sequence is not necessarily completed.

When PSW bit 13 is zero and a hard machine-check condition is generated, subsequent action depends on the state of the hard stop bit, bit 0 of control register 14. When the hard stop bit is zero, the machine-check condition is held pending, and an attempt is made to complete the execution of the current instruction and to proceed with the next sequential instruction. It should be noted that operation in this mode may result in loss of system integrity. When the hard stop bit is one, processing stops immediately, and the CPU enters the check stop state. Depending on the model and type of error, the CPU may enter the check stop state even when the hard stop bit is set to zero.

Similarly, subsequent action depends on the state of the hard stop bit if, during the execution of the interruption procedure resulting from a previous hard machine-check condition, another hard machine-check condition is detected. If the hard stop bit is one, the CPU enters the check stop state; if the bit is zero, an attempt is made to proceed with the condition held pending for subsequent interruption. If a hard machine-check condition is detected during an interruption due to a soft machine-check condition, the interruption becomes a system damage report.

Hard machine-check conditions held pending while the hard stop bit is zero remain pending and do not cause the CPU to enter the check stop state if the hard stop bit is subsequently set to one.

When a soft machine-check condition is detected with the CPU disabled for the associated machine-check condition, the condition is held pending. When a system-recovery condition is detected during the execution of the interruption procedure caused by a previous soft or hard machine-check condition, the system-recovery condition may be combined with the other conditions, discarded, or held pending. The CPU never enters the check stop state because of a soft machine-check condition.

Only one machine-check condition is held pending for each subclass, regardless of the number of conditions that may have been detected.

Machine-check interruptions can be initiated only by a condition in a subclass for which the CPU is enabled. Conditions in other subclasses which are pending may also

be indicated in the same interruption even though the CPU is not enabled for those subclasses. All conditions that are indicated are then cleared.

Machine-check conditions are handled in the same manner in both the running and wait states. In the wait state a machine-check condition for which the CPU is enabled causes an immediate interruption.

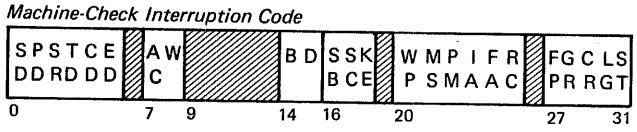
Machine checks that occur while processing takes place in instruction-step mode are handled in the same manner as in process mode; *i.e.*, normal recovery, logout, and machine-check interruptions occur when allowed. Machine checks that occur during a manual operation, such as system reset, set IC, or store, may generate a system-recovery condition. If the condition causes damage which is not corrected or circumvented, the CPU enters the check stop state.

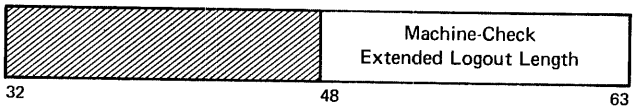
Every reasonable attempt is made to limit the side-effects of any hard machine-check condition and the associated interruption. Normally, I/O and external interruptions, as well as the progress of I/O data transfer and the updating of the timer, remain unaffected. The malfunction, however, may affect these activities, and, if the currently active PSW has bit 13 set to one, the machine-check interruption may terminate the switching of PSWs that is taking place because of another type of interruption.

MACHINE-CHECK INTERRUPTION CODE

The machine-check interruption code (MCIC) is an eight-byte field starting at location 232.

Machine-Check Interruption Code





0-5, 7-8	Subclass
14-15	Time of interruption occurrence
16-18	Storage errors
20-25, 27-31	Validity
6, 9-13, 19, 26, 32-47	Not assigned, stored as zero

Subclass

Bits 0–8 identify the machine-check conditions causing the interruption. At least one bit is stored as a one in the subclass field. When multiple errors occur, several bits may be one.

System Damage (SD): Bit 0, when one, indicates that interruptions may have been lost, or that damage has occurred which cannot be isolated to one or more of the less severe machine-check damage subclasses.

Instruction Processing Damage (PD): Bit 1, when one, indicates that a malfunction has been detected which may have caused incorrect results in the processing of instructions. For damage to be indicated as instruction processing damage, the point of error and the point of interruption must not be separated by an interruption or by a LOAD PSW instruction, and the extent of the damage must fall within one or more of the following categories:

1. The damaged area still contains bad parity.
2. The damaged area lies within the destination operand of the instruction.
3. The damaged area lies within the general registers, floating-point registers, control registers, or PSW.

System Recovery (SR): Bit 2, when one, indicates that errors were detected but have been successfully corrected or circumvented without loss of system integrity. The indication of system recovery does not imply storage logical validity, or that the fields stored as a result of the machine-check interruption are valid.

Timer Damage (TD): Bit 3, when one, indicates that damage has occurred to the timer or to location 80.

Time-of-Day Clock Damage (CD): Bit 4, when one, indicates that damage has occurred to the time-of-day clock.

External Damage (ED): Bit 5, when one, indicates that damage has occurred to a channel, channel controller, switching unit, or other unit external to the CPU, or to a storage unit during operations not directly associated with the CPU. Timer damage and time-of-day clock damage do not turn on the external damage bit.

Automatic Configuration (AC): Bit 7, when one, indicates that operation of a portion of the machine which cannot be noted by the program has been disabled by an equipment malfunction and that reduced performance will result even though logical operation may continue.

Warning (W): Bit 8, when one, indicates that damage is impending in some part of the system (*e.g.*, that power is about to fail, or that a loss of cooling has occurred).

Time of Interruption Occurrence

Bits 14 and 15 of the machine-check interruption code indicate when the interruption occurred in relation to the error.

Backup (B): Bit 14, when one, indicates that the point of interruption is at an equipment checkpoint before the point of error. This bit is set only when retry is unsuccessful. When the backup bit is one, a valid instruction address stored in the machine-check old PSW points to the

instruction in which the error occurred. If the backup bit is zero, a valid instruction address points to an instruction beyond the error.

Delayed (D): Bit 15, when one, indicates that some or all of the information stored as a result of the interruption was delayed in being reported. The delay resulted because the CPU was disabled for that type of interruption at the time the error was detected.

Storage Error Type

Bits 16–18 of the machine-check interruption code are used to indicate errors that occurred in main storage or in a key in storage as a result of internal or external storage requests. The failing-storage address field, when indicated as valid, identifies the area in storage found to be in error. The portion of the system affected by the storage or protection error is indicated in the subclass field of the machine-check interruption code. Storage or protection errors that occur on prefetched or unused data may be indicated by a recovery report or by a damage report, or they may not be indicated, depending on the model.

Storage Error Uncorrected (SE): Bit 16, when one, indicates that a reference to storage caused or resulted in the detection of damaged data that could not be corrected.

Storage Error Corrected (SC): Bit 17, when one, indicates that a reference to storage caused or resulted in the detection of an error which was corrected.

Key in Storage Error Uncorrected (KE): Bit 18, when one, indicates that a reference to a key in storage caused or resulted in the detection of an uncorrectable error in the key in storage. The keys in storage are not checked for errors during storage references when the PSW key or channel key referring to storage is all zeros.

Programming Note: The storage error type bits do not in themselves indicate the occurrence of damage because the error detected may not have affected the result.

Machine-Check Code Validity Bits

Bits 20–31 of the machine-check interruption code are validity bits. With the exception of the storage logical validity bit (bit 30), each bit indicates the validity of a particular field stored during the machine-check interruption. When a validity bit is one, it indicates that the error conditions did not affect the original information and that no error was detected when the data was stored.

PSW AMWP Validity (WP): Bit 20, when one, indicates that bits 12–15 of the machine-check old PSW are valid.

PSW Masks and Key Validity (MS): Bit 21, when one, indicates that all PSW bits other than those in the interruption code, ILC, AMWP, instruction address, condition code, and program mask of the machine-check old PSW are valid.

Program Mask and Condition Code Validity (PM): Bit 22, when one, indicates that the program mask and condition code in the machine-check old PSW are valid.

Instruction Address Validity (IA): Bit 23, when one, indicates that the instruction address in the old PSW accurately reflects the point in the instruction sequence at which the interruption occurred. If the backup bit is one, a valid instruction address points to the instruction in error. If the backup bit is zero, a valid instruction address points to an instruction following the error.

Failing-Storage Address Valid (FA): Bit 24, when one, indicates that the failing-storage address is valid.

Region Code Valid (RC): Bit 25 indicates that a valid region code has been stored.

Floating-Point Registers Valid (FP): Bit 27, when one, indicates that the contents in the floating-point register save area accurately reflect the state of the floating-point registers at the interruption point.

General Registers Valid (GR): Bit 28, when one, indicates that the contents stored in the general register save area accurately reflect the state of the general registers at the interruption point.

Control Registers Valid (CR): Bit 29, when one, indicates that the contents stored in the control register save area accurately reflect the state of the control registers at the interruption point.

Logout Valid (LG): Bit 30, when one, indicates that the CPU extended logout information was correctly stored.

Storage Logical Validity (ST): Bit 31, when one, indicates that the contents of those storage locations that are modified by the instruction processing stream contain the correct information relative to the point of interruption. That is, all stores prior to the point of interruption are completed, and all stores, if any, beyond the point of interruption are suppressed or restored to the original contents. The storage logical validity may be one when a store prior to the point of interruption is suppressed because of an uncorrected storage error.

Programming Notes: The validity bits must be used in addition to the subclass indication and time of occurrence bits to determine the extent of the damage caused by the machine-check condition. The four PSW validity bits, the

three register validity bits, and the storage logical validity bit must all be ones in addition to the following to indicate that no damage has yet occurred to the system:

1. The backup bit is zero and all of the damage subclass bits are zero.
2. The backup bit is one, and the delayed bit is zero, and instruction processing damage is the only damage subclass bit that is one.

Machine-Check Extended Logout Length

Bits 48–63 of the machine-check interruption code indicate the length in bytes of the information stored in the extended logout area starting at the location specified by the machine-check extended logout pointer. When no extended logout has occurred, this field is set to zero.

MACHINE-CHECK EXTENDED INTERRUPTION INFORMATION

The machine-check extended interruption information consists of five fields, which are stored at machine-check interruption time. Each of these fields has a validity bit associated with it in the machine-check interruption code. If for any reason the machine cannot store one of these fields or cannot store the field validly, the associated validity bit is set to zero.

Failing-Storage Address: When a storage-error-uncorrected, storage-error-corrected, or key-in-storage-error-uncorrected has been indicated, the failing storage-address is stored in bits 8–31 of the word at location 248. Bits 0–7 of the word are set to zero. In the case of storage errors, the failing-storage address may point to any address within the ECC block. For key-in-storage-error-uncorrected, the failing-storage address may point to any address within the 2048-byte block of storage associated with the key in storage that is in error. When an error is detected in more than one location before the interruption, the failing-storage address may point to any of the failing locations.

Region Code: The word at location 252 contains model-dependent information that more specifically defines the location of the error. For example, it may contain a model-dependent address of the unit causing an external damage or recovery report.

Register Save Area: On all machine-check interruptions, the addressable registers are saved sequentially in storage. Floating-point registers 0, 2, 4, and 6 are stored starting at location 352, general registers 0–15 are stored starting at location 384, and control registers 0–15 are stored starting at location 448. Locations assigned to control registers that are not implemented are set to zero.

CPU IDENTIFICATION

The instruction STORE CPU ID provides for determining the identity of the CPU and the amount of storage that must be allocated for the machine-check extended logout.

STORAGE VALIDATION

When error checking and correction (ECC) are incorporated in main storage, special procedures are necessary to restore or place new information in a storage area that has multiple-bit errors.

Because the check bits are associated with a block of data consisting of multiple bytes, correction must be done on a block basis rather than on a byte basis. When a block contains multiple-bit errors, and a store operation attempts to store into the block without replacing the entire block, the data in the block (including the check bits) are regenerated by the storage unit, and no new data is entered into the block. Normally the contents of the block can only be changed by presenting an entire block of data to be entered on one storage cycle.

The restoring of parity and data in a storage location is called storage validation. Validation of storage is provided as a program function and is also provided with certain manual operations.

Programmed Validation

An ECC block with multiple-bit errors is never validated under programming control unless the entire contents of the block are replaced. Even when an instruction causes the

entire contents of an ECC block to be replaced, validation may or may not occur depending on the instruction and the model. However, all models cause validation to occur under the following conditions.

A move instruction, either MVC or MVCL, validates the storage area containing the first operand if the first and second operands do not overlap, and if the first operand starts on a boundary of an ECC field and is an integral number of ECC fields in length. A machine-check condition is not generated for original errors in the first operand. Any other errors are indicated in accordance with the normal procedure.

Validation on Input Operations

Input operations do not validate an ECC block that is only partially replaced. When the entire contents of an ECC block are replaced by an input operation, validation may or may not occur, depending on the channel type.

PERMANENTLY ALLOCATED STORAGE LOCATIONS

The locations shown in Figure 5 are permanently allocated.

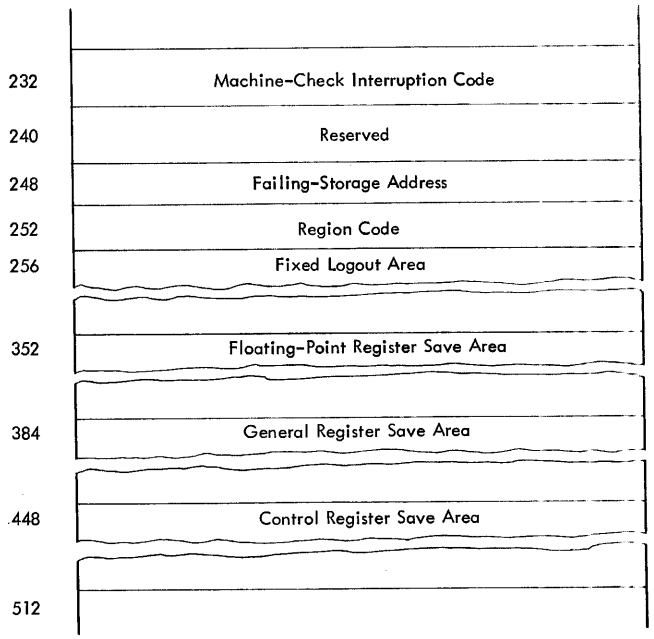


Figure 5. Permanently Allocated Storage Locations

I/O CONDITION CODES

Figure 6 lists the conditions identified and the corresponding condition codes for each instruction. The states of the system and associated abbreviations are defined in the "Input/Output Operations" section of the *IBM System/360 Principles of Operation*, GA22-6821. The digits in Figure 6 represent the numeric value of the code. The instruction START I/O can set code 0 or 1 for the AAA state, depending on the type of operation initiated. Equipment malfunctions and programming errors generally cause condition code 1 to be set and the CSW to be stored.

BLOCK-MULTIPLEXING CONTROL

To facilitate a transition from the use of a selector channel to a block-multiplexer channel, a mode control is furnished. In systems with control registers, this mode control is provided by bit position 0 of control register 0. On all other systems, this control is provided by a manual switch on the system console.

The mode dynamically controls the operation of channels capable of block-multiplexing. The mode may change at any time, and its effect on channel activity depends on the channel and subchannel states.

The mode control affects the channel only during the execution of a START I/O instruction. If, during the execution of START I/O the mode control specifies block-multiplex operation, the channel program that is initiated is executed in the block-multiplex mode. If the mode control at START I/O time prohibits block-multiplexing, the channel program is executed with the channel operating as a selector channel, that is, a channel executing a complete command chain appears to have a single subchannel until the interruption condition that terminates the channel program is cleared. When that interruption condition is cleared, the channel is free to resume execution of any disconnected channel programs.

LIMITED CHANNEL LOGOUT

The limited channel logout (LCL) field (locations 176–179) contains model-independent information related to equipment errors detected by the channel. The field, if stored, may only be stored when the CSW or a portion of the CSW is stored and may or may not be accompanied by the full channel logout.

The LCL is used to report errors successfully circumvented by the channel or to provide detailed machine status when errors have affected I/O operations.

Conditions	State Abbreviation	SIO SIOF▲	TIO	HIO	HDV	TCH
Available	AAA	0,1*	0	1*	1*	0
Interruption pending in device	AAI	1*	1*	1*	1*	0
Device working	AAW	1*	1*	1*	1*	0
Device not operational	AAN	3	3	3	3	0
Interruption pending in subchannel	AIX**					
For the addressed device		2	1*	0	0	0
For another device		2	2	0	0	0
Subchannel working	AWX**					
With the addressed device		2	2	1*#	1*#	0
With another device		2	2	1*#	0	0
Subchannel not operational	ANX**	3	3	3	3	0
Interruption pending in channel	IXX**	——	See Note		——	1
Channel working	WXX**					
With the addressed device		2	2	2	+	2
With another device		2	2	2	≠	2
Channel not operational	NXX**	3	3	3	3	3

* The CSW or its status portion is stored at location 64 during execution of the instruction.

** The symbol X stands for A, I, W, and N, and indicates that the state of the corresponding component is not significant. As an example, AIX denotes the states AIA, AII, AIW, and AIN, while IXX represents a total of 16 states, some of which do not occur.

– The condition cannot be identified during execution of the instruction.

≠ The condition code depends on the state of the subchannel, the channel type, and system model. If the subchannel is not operational, a condition code 2 or 3 is set. If the subchannel is available or working with the addressed device, a condition code 2 is set. Otherwise, a condition code 0 or 2 is set.

When a "device not operational" response is received in selecting the addressed device, condition code 3 is set.

▲ START I/O FAST RELEASE may cause the same condition code to be set as for START I/O or may cause condition code 0 to be set.

The condition code depends on the I/O interface sequence, the channel type, and the system model. If the channel ascertains that the device received the signal to terminate, a condition code 1 is set and the CSW stored. Otherwise, a condition code 2 is set.

NOTE: For the purpose of executing START I/O, START I/O FAST RELEASE, TEST I/O, HALT DEVICE, and HALT I/O, a channel containing a pending interruption condition appears the same as an available channel, and the condition-code setting depends on the states of the subchannel and device. The condition codes for the IXX states are the same as for the AXX states, where the X's represent the states of the subchannel and the device. As an example, the condition code for the IAA state is the same as for the AAA state, and the condition code for the IAW state is the same as for AAW.

Figure 6. I/O Condition Codes

The bits of the field are defined as follows:

0 *Reserved.* Stored zero.

1–3 *Identity of the storage control unit* (SCU) through which storage references were directed when an error was detected. This identity is not necessarily the identity of the storage unit involved with the transfer. When only one physical path exists between channel and storage, the storage control unit has the identity of the CPU. If more than one path exists, the storage control unit has its own identity.

When bit 3 is zero, bits 1 and 2 are meaningless. In this case, the SCU identity is implied to be the same as the CPU identity. When bit 3 is one, the binary value of bits 1 and 2 identify a physical SCU. Each SCU in the system has a unique identity.

4–7 *Detect field* identifies the type of unit that detected the error. At least one bit is present in this field, and multiple bits may be set when more than one unit detects the error.
Bit 4–CPU
Bit 5–Channel
Bit 6–Storage control unit
Bit 7–Storage unit

8–12 *Source field* indicates the most likely source of the error. The determination is made by the channel on the basis of the type of error check, the location of the checking station, the information flow path, and the success or failure of transmission through previous check stations.

Normally, only one bit will be present in this field. However, when inter-unit communication cannot be resolved to a single unit, such as when the interface between units is at fault, multiple bits (normally two) may be set in this field. When a reasonable determination cannot be made, all bits on this field are set to zero.

If the detect and source fields indicate different units, the interface between them can also be considered suspect.
Bit 8–CPU
Bit 9–Channel
Bit 10–Storage control unit
Bit 11–Storage unit
Bit 12–Control unit

13 *Reserved.* Stored zero.

14 *Successful channel retry* has been performed. The storing of the CSW and limited logout will not be caused by the error which was successfully retried. Rather, the success is reported at such time as the storing occurs for some other reason. When the storing of the CSW has not been caused by a channel control check, an interface control check, or a channel data check, bits 4–6 of the channel status field are zero, and the successful retry is indicated in the limited channel logout. This signifies that one or more successful retries have been performed and no unretriable or unsuccessful conditions have been detected.

If bits 4–6 of the channel status field are nonzero, either an unretriable or an unsuccessful retry has occurred. If the successful channel retry bit is also one, at least one successful retry has also occurred since the last storing of the CSW.

15 *Reserved.* Stored zero.

16–23 *Field validity flags.* These bits indicate the validity of the information stored in the designated fields. When the designated field is stored by the channel with the correct contents, the validity bit is one. When the designated field is stored by the channel with unpredictable contents, the validity bit is zero. The validity bits for nonstored fields are meaningless.

The fields designated are:
Bit 16–Interface address
Bit 17–(Reserved. Stored zero)
Bit 18–(Reserved. Stored zero)
Bit 19–Sequence code
Bit 20–Unit status
Bit 21–Command address and key
Bit 22–Channel address
Bit 23–Device address

24–25 *Type of termination* that has occurred is indicated by these two bits.

This encoded field has meaning only when a channel control check or an interface control check is indicated in the CSW. When neither of these two checks is indicated, no termination has been forced by the channel.
00 Interface disconnect
01 Stop, stack, or normal termination
10 Selective reset
11 System reset

26–27 *Reserved.* Stored zero.

28 *I/O Error Alert.* This bit, when set to one, indicates that the limited channel logout resulted from the signaling of I/O error alert on the I/O interface by the indicated unit. The channel performs a malfunction reset and causes interface control check to be set.

29–31 *Sequence code* identifies the I/O sequence in progress at the time of the error. It is meaningless if stored during the execution of HALT I/O or HALT DEVICE.

For all cases, the channel program address, if validly stored and if nonzero, is the address of the current CCW+8.

When a TEST I/O is issued to a channel with a pending channel logout condition, the TEST I/O causes the channel to store a CSW, perform a logout, and reset the condition without regard for the device address associated with the TEST I/O. The I/O address that pertains to the logout condition is stored in locations 185–187.

A logout resulting from an error that occurred during the execution of a TEST I/O instruction can be differentiated from a logout cleared by a TEST I/O by examining the sequence code, because sequence code zero can only occur in the first situation.

The sequence code assignments are:

000 A channel-detected error occurred during the execution of a TEST I/O instruction.

001 Command Out with a nonzero command byte on Bus Out has been sent by the channel, but device status has not yet been analyzed by the channel. This code is set with a Command Out response to Address In during initial selection.

010 The command has been accepted by the device, but no data has been transferred. This code is set by a Service Out or Command Out response to Status In during an initial selection sequence, if the status is either channel end alone, or channel end and device end, or channel end, device end, and status modifier, or all zeros.

011 At least one byte of data has been transferred over the interface. This code is set with a Service Out response to Service In and, when appropriate, may be used when the channel is in an idle or polling state.

100 The command in the current CCW has either not yet been sent to the device or else was sent but not accepted by the device. This code is set when one of the following conditions occurs:

1. When the command address is updated during command chaining or a START I/O.
2. When Service Out or Command Out is raised in response to Status In during an initial selection sequence with the status on Bus In including attention, control unit end, unit check, unit exception, busy, status modifier (without channel end and device end) or device end (without channel end).

3. When a short, control-unit-busy sequence is signaled.
4. When command retry is signaled.
5. When the channel issues a Test I/O command rather than the command in the current CCW.

101 The command has been accepted, but data transfer is unpredictable. This code applies from the time a device comes on the interface until the time it is determined that a new sequence code applies. It may thus be used when a channel goes into the polling or idle state and it is impossible to determine that code two or three applies. It may also be used at other times when a channel cannot distinguish between code two or three.

110 *Reserved.*

111 *Reserved.*

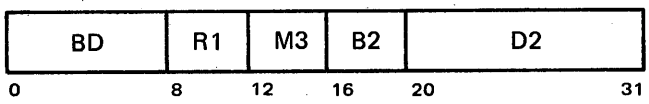
EXTENDED CHANNEL LOGOUT

The IOEL Pointer (locations 173–175) is program-set to designate an area to be used by channels not capable of storing or not choosing to store the channel logout information in the fixed logout area (locations 256–351). The low-order three bits of the pointer are reserved and are ignored by the channel so that the I/O extended logout always begins on a doubleword boundary. Channel logout information may be stored in the IOEL area only when the IOEL mask bit (CR14 bit 2) is one.

Thirteen new instructions are described in this section:
- Compare Logical Characters Under Mask (CLM)
- Compare Logical Long (CLCL)
- Insert Characters Under Mask (ICM)
- Load Control (LCTL)
- Move Long (MVCL)
- Set Clock (SCK)
- Shift and Round Decimal (SRP)
- Start I/O Fast Release (SIOF)
- Store Channel ID (STIDC)
- Store Characters Under Mask (STCM)
- Store Clock (STCK)
- Store CPU ID (STIDP)
- Store Control (STCTL)

Compare Logical Characters Under Mask

CLM R1,M3,D2(B2) [RS]



The second operand is compared with the first operand under control of a mask, and the result is indicated in the condition code.

The contents of the M3 field, bit positions 12–15, are used as a mask. The four bits of the mask, left to right, are made to correspond one for one with the four bytes, left to right, of the general register designated by the R1 field. The byte positions corresponding to ones in the mask are considered as a contiguous field and are compared with the second operand. The second operand is a contiguous field in storage starting at the second-operand address and equal in length to the number of one-bits in the mask. The bytes in the general register corresponding to zeros in the mask do not participate in the operation.

The comparison is performed considering the operands to be binary unsigned quantities, with all codes valid. Neither operand is changed.

Protection and addressing exceptions are recognized regardless of whether the mask is zero.

Resulting Condition Code:

0 Selected bytes are equal, or mask is zero
1 Selected field of first operand is low
2 Selected field of first operand is high
3 – –

Program Interruptions:

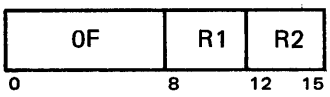
Operation: The instruction is not installed. The operation is suppressed.

Protection: The location of the second operand is protected against fetching, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

Addressing: The location of the second operand is outside the available main storage of the system. The operation is terminated.

Compare Logical Long

CLCL R1,R2 [RR]



The first operand is compared with the second operand, and the result is indicated in the condition code. The shorter operand is considered extended with the padding character.

The R1 and R2 fields each designate a pair of general registers that must start with an even register.

The leftmost byte of the first-operand and second-operand locations is designated by the contents of bit positions 8–31 of the general registers specified by the R1 and R2 fields, respectively. The number of bytes in the first-operand and second-operand locations is specified by the contents of bit positions 8–31 of general registers having addresses R1+1 and R2+1, respectively. Bit positions 0–7 of register R2+1 contain the padding character. The contents of bit positions 0–7 of registers R1, R2, and R1+1 are ignored.

The comparison is performed considering the operands to be binary unsigned quantities, with all codes valid. The comparison starts at the high-order end of both fields and proceeds to the right. The operation ends as soon as an inequality is detected or the end of the longest operand is reached. If the operands are not of the same length, the shorter operand is extended for the purpose of comparison with the padding character. Neither operand is changed.

If both operands are of zero length, the operands are considered equal, and no protection or addressing exceptions are indicated.

The execution of the instruction may be interrupted by an external event. If an I/O or external interruption condition is presented to the CPU during the execution of the instruction, with the CPU enabled for the interruption, the interruption is taken. The contents of registers R1+1 and R2+1 are decremented by the number of bytes

compared, and the contents of registers R1 and R2 are incremented by the same number, so that the instruction, when re-executed, resumes at the point of interruption. If the operation is interrupted after the shorter operand has been exhausted, the count field pertaining to the shorter operand is zero, and its address is updated accordingly. The high-order byte of registers R1 and R2 is set to zero; the contents of the high-order byte of registers R1+1 and R2+1 remain unchanged. When this type of interruption occurs, the instruction address in the old PSW appears as if the instruction had not yet been executed.

If the operation ends because of a mismatch, the count and address field at completion identify the byte of mismatch. The contents of bit positions 8–31 of registers R1+1 and R2+1 are decremented by the number of bytes that matched, unless the mismatch occurred with the padding character, in which case the count field for the shorter operand is set to zero. The contents of bit positions 8–31 of registers R1 and R2 are incremented by the amounts by which the corresponding count fields were reduced. If the two operands, including the padding character, if necessary, are equal, both count fields are made zero at completion, and the addresses are incremented by the corresponding count values. The contents of bit positions 0–7 of registers R1 and R2 are set to zero, including the case when one or both of the original count values are zero. The contents of bit positions 0–7 of registers R1+1 and R2+1 remain unchanged.

When part of an operand is specified in an unavailable or a fetch-protected area, on some models, the operation may be terminated by a protection or addressing exception, respectively, although an inequality could have been found in the comparison of the available operand parts.

When the count field for an operand has the value zero, no addressing or protection exceptions are recognized for that operand location.

Resulting Condition Code:

0 Operands are equal, or both fields have zero length
1 First operand is low
2 First operand is high
3 – –

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: An operand location is protected against fetching, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

Addressing: An operand is located outside the available main storage of the installation. The operation is terminated.

Specification: The R1 or R2 field contains an odd register address. The operation is suppressed.

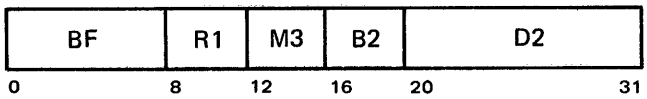
Programming Notes

When the contents of the R1 and R2 fields are the same, the condition code is set to 0, and protection and addressing exceptions are indicated when called for by the operand designation.

See also the programming notes under "Move Long."

Insert Characters Under Mask

ICM R1,M3,D2(B2) [RS]



Bytes from contiguous locations beginning at the second-operand address are inserted into the first operand under control of a mask.

The contents of the M3 field, bit positions 12–15, are used as a mask. The four bits of the mask, in the order of ascending bit numbers, are made to correspond one for one with the four bytes, in the order of ascending byte numbers, of the general register designated by the R1 field. The byte positions corresponding to ones in the mask are filled, in the order of ascending byte numbers, with bytes from the storage operand. Bytes are fetched from contiguous storage locations beginning at the second-operand address. The length of the second operand is equal to the number of ones in the mask. The bytes in the general register corresponding to zeros in the mask remain unchanged.

The resulting condition code is based on the mask and on the value of the bits inserted. When the mask is zero or when all inserted bits are zero, the condition code is made 0. When all inserted bits are not zero, the code is set according to the leftmost bit of the storage operand: if this bit is one, the code is made 1 to indicate a negative algebraic value; if this bit is zero, the code is made 2, reflecting a positive algebraic value.

Protection and addressing exceptions are recognized regardless of whether the mask is zero.

Resulting Condition Code:

0 All inserted bits are zero, or mask is zero
1 First bit of the inserted field is one
2 First bit of the inserted field is zero
3 – –

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: The second-operand location is protected against fetching, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

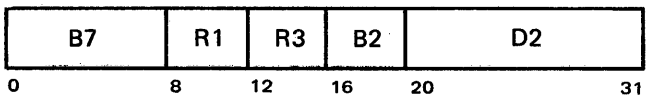
Addressing: The second-operand location is outside the available main storage of the installation. The operation is terminated.

Programming Note

The condition code for INSERT CHARACTERS UNDER MASK is defined such that when the mask is 1111, the instruction causes the same condition code to be set as for LOAD AND TEST; for partial replacement of the register, the case is identified where the inserted bytes are zero. Consequently, when the high-order bit of the inserted field is one, the condition code is set to indicate a negative value. It should be noted, however, that when the high-order byte of the register is not replaced, the result in the register does not necessarily behave as a negative number in arithmetic.

Load Control

LCTL R1,R3,D2(B2) [RS]



The set of control registers starting with the control register specified by the R1 field and ending with the control register specified by the R3 field is loaded from the locations designated by the second-operand address.

The storage area from which the contents of the control registers are obtained starts at the location designated by the second-operand address and continues through as many storage words as the number of control registers specified. The control registers are loaded in ascending order of their addresses, starting with the control register specified by the R1 field and continuing up to and including the control register specified by the R3 field, with control register 0 following control register 15. The second operand remains unchanged.

An attempt is made to fetch the operand from main storage for each of the designated control registers, regardless of whether the feature requiring the presence of the control register is provided and whether the corresponding register is installed. Whenever the storage reference causes an addressing or protection exception, the exception is indicated.

Condition Code: The code remains unchanged.

Program Interruptions:

Operation: The control registers addressed are not installed. The operation is suppressed.

Privileged Operation: The instruction is encountered with the CPU in the problem state. The operation is suppressed.

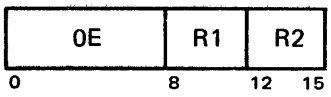
Protection: The second-operand location, or part of it, is protected against fetching, and the key in storage associated with the operand does not match the protection key in the PSW. The operation is terminated.

Addressing: The second-operand location, or part of it, is outside the available main storage of the installed system. The operation is terminated.

Specification: The second operand is not located on a word boundary. The operation is suppressed.

Move Long

MVCL R1,R2 [RR]



The second operand is placed in the first-operand location, if overlapping of operand locations does not affect the final contents of the first-operand location. The remaining low-order byte positions, if any, of the first-operand location are filled with the padding character.

The R1 and R2 fields each designate a pair of general registers that must start with an even register.

The leftmost byte of the first-operand and second-operand locations is designated by the contents of bit positions 8–31 of the general registers specified by the R1 and R2 fields, respectively. The number of bytes in the first-operand and second-operand locations is specified by the contents of bit positions 8–31 of general registers having addresses R1+1 and R2+1, respectively. Bit positions 0–7 of register R2+1 contain the padding character. The contents of bit positions 0–7 of registers R1, R2, and R1+1 are ignored.

The movement starts at the high-order end of both fields and proceeds to the right. The bytes to be moved are not changed or inspected. The operation is ended when the number of bytes specified by bit positions 8–31 of register R1+1 have been moved into the first-operand location. If the second operand is shorter than the first operand, the remaining low-order bytes of the first-operand are filled with the padding character.

As part of the execution of the instruction, the values of the two count fields are compared for the setting of the condition code, and a check is made for destructive overlap of the operands. Operands are said to overlap destructively when the first-operand location is used as a source after data has been moved into it, assuming movement to be performed one byte at a time. When the operands overlap destructively, no movement takes place, and condition code 3 is set.

Depending on whether the second operand wraps around from location 16,777,215 to location 0, movement takes place in the following cases:

1. When the second operand does not wrap around, movement is performed when the high-order byte of the first operand coincides with or is to the left of the high-order byte of the second operand, *or* if the high-order byte of the first operand is to the right of the rightmost second-operand byte participating in the operation.
2. When the second operand wraps around, movement is performed when the high-order byte of the first operand coincides with or is to the left of the high-order byte of the second operand, *and* if the high-order byte of the first operand is to the right of the rightmost second-operand byte participating in the operation.

When the count specified by bit positions 8–31 of register R1+1 is zero, no movement takes place, and the condition code is set to 0, 1, or 2 to indicate the relative values of the counts.

The execution of the instruction may be interrupted by an external event. If an I/O or external interruption condition is presented to the CPU during the execution of the instruction with the CPU enabled for the interruption, the interruption is taken. The contents of registers R1+1 and R2+1 are decremented by the number of bytes moved, and the contents of registers R1 and R2 are incremented by the same number, so that the instruction, when re-executed, resumes at the point of interruption. If the operation is interrupted during padding, the count field in register R2+1 is zero, the address in register R2 is incremented by the original contents of register R2+1, and registers R1 and R1+1 reflect the extent of the padding operation. The high-order byte of registers R1 and R2 is set to zero; the contents of the high-order byte of registers R1+1 and R2+1 remain unchanged. When this type of interruption occurs, the instruction address in the old PSW appears as if the instruction had not yet been executed.

At the completion of the operation, the count in register R1+1 is zero, and the address in register R1 is incremented by the original value of the count in register R1+1. The count in register R2+1 is decremented by the number of bytes moved out of the second-operand location, and the address in register R2 is incremented by the same amount. The contents of bit positions 0–7 of registers R1 and R2 are set to zero, including the case when one or both of the original count values are zero or when a condition code of 3 is set. The contents of bit positions 0–7 of registers R1+1 and R2+1 remain unchanged.

When the count specified by bit positions 8–31 of register R1+1 is zero, or condition code 3 is set, no addressing or protection exceptions are recognized. When the count specified by bit positions 8–31 of register R2+1 is zero, no addressing or protection exceptions for the second-operand location are recognized.

Resulting Condition Code:

0 First-operand and second-operand counts are equal
1 First-operand count is low
2 First-operand count is high
3 No movement performed because of destructive overlap

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: The second-operand location is protected against fetching or the first-operand location is protected against storing, and the key in storage associated with the operand does not match the protection key in the PSW. The operation is terminated.

Addressing: An operand is located outside the available main storage of the installation. The operation is terminated.

Specification: The R1 or R2 field contains an odd register address. The operation is suppressed.

Programming Notes

The instruction MOVE LONG can be used conveniently for clearing main storage. Clearing can be accomplished by setting the padding character to zero and the second-operand count to zero.

When the first-operand count is zero, the operation consists in setting the condition code and setting the high-order bytes of registers R1 and R2 to zero.

When the contents of the R1 and R2 fields are the same, the operation proceeds the same as when two distinct pairs of registers having the same contents are specified. Condition code 0 is set, and protection and addressing exceptions are indicated when called for by the operand designation.

Care must be exercised when MOVE LONG is made a subject instruction of EXECUTE and any register used by MOVE LONG is designated as either the R1, X2, or B2 register for EXECUTE, because on resumption of execution after an I/O or external interruption, the updated values of these registers will be used in the execution of EXECUTE. Similarly, care must be exercised in letting the second-operand location include the location of the instruction, since an interruption may cause the new contents of the location to be interpreted for a resumption of the execution.

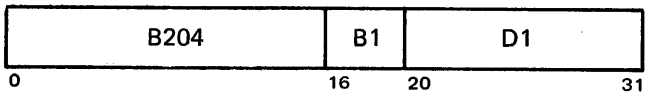
Because the execution of MOVE LONG is interruptible, the instruction cannot be used for situations where the program must rely on uninterrupted execution of the instruction or on the timer not being updated during the execution of the instruction.

When the stop key is pressed during the execution of MOVE LONG or COMPARE LOGICAL LONG, the CPU enters the stopped state at the completion of the execution of the instruction unless an external or I/O condition causes

the execution to be interrupted. If the execution is interrupted, the CPU enters the stopped state immediately after switching the PSWs. Similarly, in instruction-step mode the whole MVCL or CLCL instruction is executed in an instruction step, unless an external or I/O condition causes the execution to be interrupted. In this event, the instruction step ends with the switching of the PSWs.

Set Clock

SCK D1(B1) [SI]



The contents of the eight-byte field designated by the operand address replaces the current value of the time-of-day clock, the clock is placed in the set state, and the clock resumes counting with the new value.

The operand designated by the instruction is considered a fixed-point number consisting of a sign and a 63-bit integer field. This operand replaces the sign of the clock and its integer field as determined by the clock's resolution. Only those bits of the operand are set in the clock that correspond to the bit positions to be updated by the clock; the contents of the remaining low-order positions are not preserved in the clock and are ignored.

The value of the clock is changed and the clock is placed in the set state only if the clock security switch on the operator intervention panel is in the enable set position. If the switch is in the secure position, the value and the state of the clock are not changed. The two results are distinguished by condition codes 0 and 1, respectively. When the clock is not operational (the clock is disabled or its power is down), the value of the clock is not changed, and condition code 3 is set.

Resulting Condition Code:

0 Clock value set
1 Clock value secure
2 – –
3 Clock not operational

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Privileged Operation: The CPU is in the problem state. The operation is suppressed.

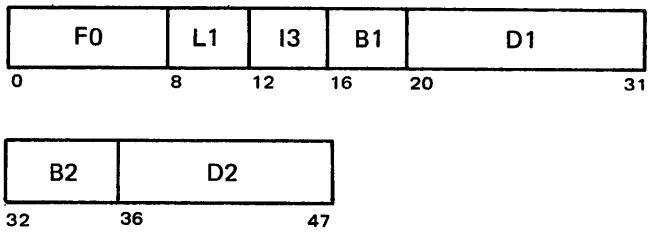
Protection: The operand location is protected against fetching, and the key in storage associated with the location does not match the protection key in the PSW. The exception is recognized regardless of whether the value of the clock is changed. The operation is terminated.

Addressing: The operand location is outside the available main storage of the installation. The exception is recognized regardless of whether the value of the clock is changed. The operation is terminated.

Specification: The operand is not located on a double-word boundary in main storage. The operation is suppressed.

Shift and Round Decimal

SRP D1(L1,B1),D2(B2),I3 [SS]



The first operand is shifted in the direction and for the number of digit positions specified by the second-operand address. When shifting to the right is specified, the first operand is rounded by the rounding factor, I3.

The second-operand address is not used to designate data; instead, the contents of bit positions 26–31 of the address are considered a signed fixed-point quantity, which indicates the direction of the shift and the number of digit positions to be shifted. The remainder of the address is ignored. When bit 26 of the second-operand address is zero, left shift is specified, and bits 27–31 of the address are considered a true binary number specifying the number of digit positions of shift. When bit 26 is one, right shift is specified, and bits 26–31, considered as a binary number in two's complement notation, specify the amount of the shift.

The first operand is considered to be in the packed decimal format and is checked for the validity of decimal digit and sign codes. Only its digit portion is shifted; the sign position does not participate in the shifting. Zeros are supplied for the vacated digit positions. The validity of the first operand is checked and the condition code is set even if a shift amount of zero is specified. A zero result is made positive.

If a significant digit is shifted out of the high-order digit position during left shift, a decimal-overflow exception is recognized. The operation is completed by ignoring the overflow.

During right shift, the contents of the I3 field, bit positions 12–15, are used as a rounding factor. The shifted operand is rounded by decimally adding the rounding factor to the last digit shifted out and propagating the carry, if any, to the left. Both the first operand and the rounding factor are considered positive quantities for the

purpose of this addition. Except for validity checking and the participation in rounding, the digits shifted out of the low-order digit position are ignored and lost. The validity of the rounding-factor code is checked regardless of the direction and amount of shift specified.

Resulting Condition Code:

0 Result is zero
1 Result is less than zero
2 Result is greater than zero
3 Result overflows

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: The first-operand location is protected, and the key in storage associated with the operand does not match the protection key in the PSW. The operation is terminated.

Addressing: The first-operand location is outside the available main storage of the installation. The operation is terminated.

Data: The sign code is invalid, or the first or third operand contains an invalid digit code. The operation is terminated for an invalid digit code; it is suppressed for an invalid sign code.

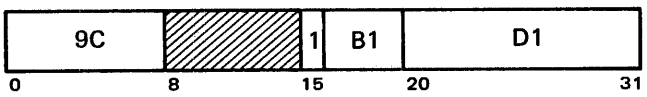
Decimal Overflow: A nonzero digit is shifted out of the high-order digit position, and the decimal-overflow mask bit is one. The operation is completed.

Programming Note

Because the two's complement notation is employed, SHIFT AND ROUND can be used for shifting up to 31 digit positions left and up to 32 digits positions right. This is sufficient to clear all digits of any decimal field even when rounding in right shift is specified.

Start I/O Fast Release

SIOF D1(B1) [SI]



A write, read, read backward, control, or sense operation is initiated with the addressed I/O device and subchannel. The instruction is executed only when the CPU is in the supervisor state. When not in block-multiplex mode, the instruction is executed as START I/O. In block-multiplex mode, the instruction is executed as follows:

Bit positions 16–31 of the sum formed by the addition of the contents of register B1 and the contents of the D1 field identify the channel, subchannel, and I/O device to which the instruction applies. The CAW contains the protection key for the subchannel and the address of the first CCW. This CCW specifies the operation to be performed, the main-storage area to be used, and the action to be taken when the operation is completed.

The I/O operation specified by START I/O FAST RELEASE may be initiated if the subchannel is available, the channel is available or is in the interruption-pending state, and errors or exceptional conditions have not been detected. The I/O operation is not initiated when the subchannel and channel are in any other state or when the channel or device detects any error or exceptional condition during execution of the instruction. The I/O operation may be initiated by some channels independent of the device. When this occurs, the device state or device-detected errors are indicated in a CSW stored during a subsequent interruption.

When any of the following conditions occurs, with the channel either available or in the interruption-pending state and with the subchannel available before the execution of the instruction, START I/O FAST RELEASE can cause the status portion of the CSW to be replaced by a new set of status bits. The status bits pertain to the device addressed by the instruction. The contents of the other fields of the CSW are not changed. Such storing of the CSW can occur only when initiation of the I/O operation is attempted at the device during the execution of START I/O FAST RELEASE. When START I/O FAST RELEASE is executed independent of the device, the following conditions will be indicated in a subsequent interruption, during which the entire CSW will be stored:

1. An immediate operation was executed, and either no command chaining is specified, or chaining is suppressed because of unusual conditions detected during the operation. An operation is called *immediate* when the I/O device signals the channel-end condition immediately on receipt of the command code. The CSW contains the channel-end bit and any other indications provided by the channel or the device. The busy bit is off. The I/O operation has been initiated, but no information has been transferred to or from the storage area designated by the CCW. No interruption conditions are generated at the device or subchannel, and the subchannel is available for a new I/O operation.
2. The I/O device contains a pending interruption condition caused by device end or attention, the control unit contains a pending control unit end for the addressed device, or, on the selector channel, the control unit contains, for the addressed device, a pending channel end following the execution of HALT I/O. The CSW unit-status field contains the busy bit, identifies the interruption condition, and may contain other bits provided by the device or control unit. The interruption condition is cleared. The channel-status field indicates any error conditions detected by the channel and contains the PCI bit if specified in the first CCW.

3. The I/O device or the control unit is executing a previously initiated operation, or the control unit has pending an interruption condition associated with a device other than the one addressed. The CSW unit-status field contains the busy bit, or, if the control unit is busy, the busy and status-modifier bits. The channel-status field indicates any error conditions detected by the channel and contains the PCI bit if specified in the first CCW.
4. The I/O device detected an equipment or programming error during execution of the instruction. The CSW identifies the error condition. The channel-end and busy bits are off, unless the error was detected after the device was selected, and the device was found to be busy, in which case the busy bit, as well as any bits indicating pending interruption conditions, are on. The interruption conditions indicated in the CSW have been cleared at the device. The I/O operation has not been initiated. No interruption conditions are generated at the I/O device or subchannel.

When the condition described in the following paragraph occurs, with the channel either available or in the interruption-pending state and with the subchannel available before the execution of the instruction, START I/O FAST RELEASE causes the status portion of the CSW to be replaced by a new set of status bits. The status bits, if any, pertain to the device addressed by the instruction. The contents of the other field of the CSW are not changed.

The channel detects an equipment or programming error during execution of the instruction. The CSW identifies the error condition. The channel-end and busy bits are off, unless the error was detected after the device was selected, and the device was found to be busy, in which case the busy bit, as well as any bits indicating pending interruption conditions, are on. The interruption conditions indicated in the CSW have been cleared at the device. The I/O operation has not been initiated. No interruption conditions are generated at the I/O device or subchannel.

On the byte-multiplexer channel, START I/O FAST RELEASE causes the addressed device to be selected and the operation to be initiated only after the channel has serviced all outstanding requests for data transfer for previously initiated operations.

Resulting Condition Code:

0 I/O operation initiated and channel proceeding with execution
1 CSW stored
2 Channel or subchannel busy
3 Not operational

Program Interruptions:

Privileged Operation: The CPU is in the problem state. The operation is suppressed.

Programming Note

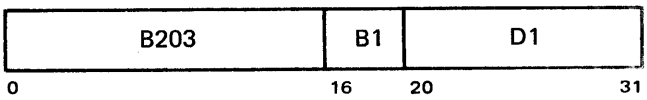
Two major differences exist between START I/O and START I/O FAST RELEASE:

1. Nonchained immediate commands on certain channels result in condition code 0 for START I/O FAST RELEASE, whereas condition code 1 is set for START I/O.
2. Condition code 0 is set by certain channels for START I/O FAST RELEASE even though the addressed device is not available or the command is invalid. The device information is supplied by means of an interruption.

The instruction START I/O FAST RELEASE may be executed by some channels as START I/O. Thus, condition code 1, 2, or 3 may be set by a channel executing START I/O FAST RELEASE as START I/O while, under identical conditions, another channel will set condition code 0.

Store Channel ID

STIDC D1(B1) [SI]

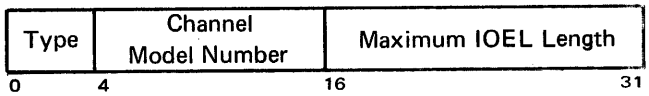


Information identifying the designated channel is stored in the four-byte field at location 168.

STORE CHANNEL ID is executed only in the supervisor state.

Bit positions 16–23 of the sum formed by the addition of the contents of register B1 and the contents of the D1 field identify the channel to which the instruction applies.

The format of the information stored at location 168 is:



Bits 0–3 specify the channel type. When a channel can operate as more than one type, the code stored identifies the channel type at the time the instruction is executed. The following codes are assigned:

0000 Selector
0001 Byte multiplexer
0010 Block multiplexer

Bits 4–15 contain the channel model number. When the channel model is implied by the channel type and the CPU model, zeros are stored in the field.

Bits 16–31 contain the length in bytes of the longest I/O extended logout that can be stored by the channel during an I/O interruption. If the channel never stores logout information using the IOEL pointer, then this field is set to zero.

Resulting Condition Code:

0 Channel ID correctly stored
1 CSW stored
2 Channel activity prohibited storing ID
3 Not operational

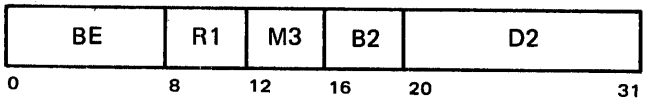
Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Privileged Operation: The CPU is in the problem state. The operation is suppressed.

Store Characters Under Mask

STCM R1,M3,D2(B2) [RS]



Bytes selected from the first operand under control of a mask are placed in contiguous byte locations beginning at the second-operand address.

The contents of the M3 field, bit positions 12–15, are used as a mask. The four bits of the mask, in the order of ascending bit numbers, are made to correspond one for one with the four bytes, in the order of ascending byte numbers, of the general register designated by the R1 field. The bytes corresponding to ones in the mask are placed in the same order in successive and contiguous storage locations beginning with the location designated by the second-operand address. The number of bytes stored is equal to the number of ones in the mask. The contents of the general register remain unchanged.

Protection and addressing exceptions are not recognized when the mask is zero.

Condition Code: The code remains unchanged.

Program Interruptions:

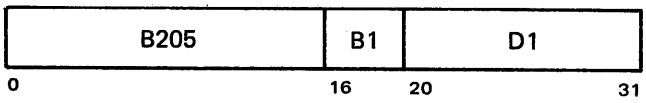
Operation: The instruction is not installed. The operation is suppressed.

Protection: The second-operand location is protected, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

Addressing: The second-operand location is outside the available main storage of the installation. The operation is terminated.

Store Clock

STCK D1(B1) [SI]



The current value of the time-of-day clock is stored at the eight-byte field designated by the operand address.

The value of the clock is expressed as a fixed-point number consisting of a sign and a 63-bit integer field. Zeros are provided for the low-order bit positions of the integer field that are not updated by the clock.

When the clock is in the error state, the value stored is unpredictable. When the clock is not operational (the clock is disabled or its power is down), zeros are stored at the operand location.

The quality of the clock value stored by the instruction is indicated by the resultant condition code setting.

Resulting Condition Code:

0 Clock in set state
1 Clock in not-set state
2 Clock in error state
3 Clock not operational

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: The operand location is protected, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

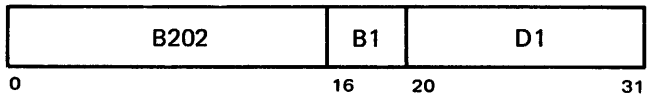
Addressing: The operand location is outside the available main storage of the installation. The operation is terminated.

Programming Note

Condition code 0 after STORE CLOCK indicates that the clock's value provides a valid measure of elapsed time since the last time it was set. This code normally indicates that the clock's value is a valid time-of-day and calendar indication. Condition code 1 indicates that the clock's value is the elapsed time since the power for the clock was turned on. In this case, the value may be used for elapsed time measurements but is not a valid time-of-day indication. Condition codes 2 and 3 indicate that the value provided by STORE CLOCK cannot be used for time measurement or indication.

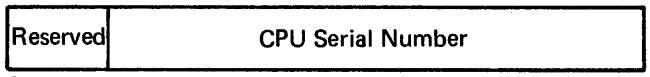
Store CPU ID

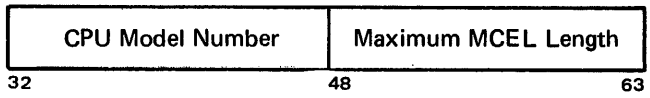
STIDP D1(B1) [SI]



Information identifying the CPU is stored in the eight-byte field designated by the operand address. STORE CPU ID is executed only in the supervisor state.

The format of the information is:





Bits 0–7 are reserved and are set to zeros.

Bits 8–31 contain the CPU serial number. The six-digit hexadecimal representation of this 24-bit field corresponds directly with the physical serial number stamped on the CPU. A sufficient number of digits from the physical serial number are recorded in this field so that when they are used with the CPU model number, the CPU is uniquely identified. The five low-order digits in the CPU serial number field are assigned to five contiguous digits in the physical serial number. When the sixth digit is not assigned, it is set to zero.

Bits 32–47 contain the CPU model number identification.

Bits 48–63 contain the length in bytes of the longest machine-check extended logout that can be stored by the machine.

Programming Note

To ensure compatibility with other models, proper operation of the program should not depend upon zeros in the reserved field.

Condition Code: The code remains unchanged.

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Privileged Operation: The CPU is in the problem state. The operation is suppressed.

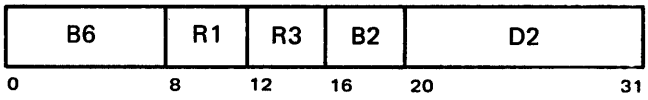
Protection: The operand location is protected, and the key in storage associated with the location does not match the protection key in the PSW. The operation is terminated.

Addressing: The operand location is outside the available main storage of the installation. The operation is terminated.

Specification: The operand is not located on a double-word boundary.

Store Control

STCTL R1,R3,D2(B2) [RS]



The set of control registers starting with the control register specified by the R1 field and ending with the control register specified by the R3 field is stored at the locations designated by the second-operand address.

The storage area where the contents of the control registers are placed starts at the location designated by the second-operand address and continues through as many storage words as the number of control registers specified. The contents of the control registers are stored in ascending order of register addresses, starting with the control register specified by the R1 field and continuing up to and including the control register specified by the R3 field, with control register 0 following control register 15. The contents of the control registers remain unchanged.

The information provided for control register positions not associated with an installed feature is unpredictable.

Condition Code: The code remains unchanged.

Program Interruptions:

Operation: The control registers addressed are not installed. The operation is suppressed.

Privileged Operation: The instruction is encountered with the CPU in the problem state. The operation is suppressed.

Protection: The second-operand location, or part of it, is protected, and the key in storage associated with the operand does not match the protection key in the PSW. The operation is terminated.

Addressing: The second-operand location, or part of it, is outside the available main storage of the installed system. The operation is terminated.

Specification: The second operand is not located on a word boundary. The operation is suppressed.

Programming Note

To ensure that presently written programs run when new features using additional control register positions are installed, only zeros should be loaded in unassigned control register positions. Similarly, in the information stored by STORE CONTROL, the program should not depend on zeros in the bit positions corresponding to the unassigned register positions.

Index

READER'S COMMENT FORM

IBM System/370 Principles of Operation GA22-7000-0

Your comments about this publication may be helpful to us. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications; this only delays the response. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name _____

Job Title _____

Address _____

_____ Zip _____

WTC users must add postage.

YOUR COMMENTS, PLEASE .....

This manual is a reference source for systems analysts, programmers and operators of IBM systems. Your comments will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

CUT ALONG THIS LINE

fold fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY .....

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: CUSTOMER MANUALS, DEPT. B98

fold fold

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM System/370 Principles of Operation Printed in U.S.A. GA22-7000-0