Reference Manual

GOODYEAR AEROSPACE CORPORATION AKRON, OHIO 44315

STARAN-E
Reference Manual

GER-16422

NOVEMBER 1977

GOODYEAR AEROSPACE
CORPORATION
AKRON, OHIO 44315

GER-16422

NOTICE

This document contains material generated by Goodyear Aerospace Corporation and is transmitted for the purpose of aiding the transaction of business between Goodyear Aerospace Corporation and the recipient. It is understood that the material contained herein will not be used, copied, or disclosed to others, without the specific written consent of Goodyear Aerospace Corporation.

The STARAN-E Computer System continues to be improved and expanded. Interested parties should contact Goodyear Aerospace Corporation, Digital Systems Marketing, Akron, Ohio 44315; or telephone (216) 794-3631 for information regarding the latest update of STARAN-E software.

STARAN is a trademark of Goodyear Aerospace Corporation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

STARAN COMPUTER SYSTEM GENERAL DESCRIPTION

STARAN, a new and unique architecture for computer systems, is the result of over a decade of intensive development effort in associative and parallel processing at Goodyear Aerospace. STARAN, the first associative processor (AP) to go into production, can operate independently or in a hybrid system to complement a conventional computer (host computer).

The STARAN approach to parallel processing is rather general and is based on the cooperative interconnection and control of three basic system components:

(1) A Multi-Dimensional Access (MDA) memory
(2) A set of processing elements (PE's)
(3) A communications network connecting the MDA to the PE's and both of these to other devices

FEATURES

MDA ARRAYS

The key component of the STARAN computer system is the MDA array memory, which provides content addressability and parallel processing capabilities.

ASSOCIATIVE PROCESSOR CONTROL

The AP control performs data manipulations within the MDA arrays as directed by instructions stored in AP control memory.

ASSOCIATIVE PROCESSOR CONTROL MEMORY

AP control memory contains high speed page memories and a High Speed Data Buffer (HSDB) to provide fast access to data and instructions that require frequent access and/or fast execution.

AP control memory also contains a main memory for program storage.

A block of AP control memory addresses are reserved for Direct Memory Access (DMA) to a host computer.

A basic STARAN Control Memory consists of:

(1) Three page memories, each containing 4096 32-bit words
(2) One HSDB containing 512 32-bit words
(3) A main memory containing up to 32,768 32-bit words
(4) Any addresses not used for main and HSDB memory are reserved for DMA

PROGRAM PAGER

The program pager moves program segments, which require fast execution, from main to the page memories.

SEQUENTIAL CONTROLLER AND MEMORY

The Sequential Controller (SC) provides offline capabilities for assembling and debugging STARAN programs, a communication link between STARAN and the operator, and control for diagnostic and test programs.

The basic system contains 16,384 16-bit words of sequential control memory.

EXTERNAL FUNCTION LOGIC

External function logic enables an element of STARAN to control and interrogate the status of other elements. An external function code may be issued by AP control, the program pager, sequential control, and the host computer.

INPUT/OUTPUT

The following input/output variations are provided on the STARAN system:

(1) Direct Memory Access to a host computer
(2) An input/output channel 32 bits wide to STARAN control memory (BIO)
(3) External Function channel to pass function codes between the STARAN control elements
(4) Multiplexed Input/Output unit (MIO) providing:
    (a) Continuous transmit mode
    (b) Block transmit mode
    (c) Exchange mode
(5) STARAN Command Channel

PHYSICAL DESCRIPTION

The basic STARAN computer consists of four standard-size cabinets along with a free-standing keyboard printer. The four cabinets are: sequential control cabinet, AP control cabinet, AP memory cabinet, and custom I/O cabinet. Two array modules can be included in the MDA array cabinet. An expanded STARAN configuration can contain up to 8 MDA array modules.

CHAPTER 1

STARAN-E ARCHITECTURE

SECTION I. INTRODUCTION

GENERAL

The STARAN-E system introduces a new concept in computers, designed to achieve very high processing rates economically. Figure 1-1 shows a block diagram of the STARAN-E computer. Each block of the diagram is discussed briefly in the following paragraphs. More detailed discussions are presented in subsequent areas of this manual.

MEMORY

STARAN-E consists of two separate memory organizations: the AP control memory, essentially a program memory, and the Multi-Dimensional Access (MDA) array memory, a data memory.

ASSOCIATIVE PROCESSOR CONTROL MEMORY

The main function of the Associative Processor (AP) control memory is to contain assembled AP application programs. The control memory can also contain items of data and act as a buffer between AP control and other elements of STANAN-E. The AP control memory consists of up to 16 memory blocks. Each block contains 4096 32 bit words. Some memory blocks are fast to enable AP control to fetch its instructions rapidly (page memories and the High Speed Data Buffer); others are slower to economically contain the entire control program (main memory).

MDA ARRAY MEMORY

The heart of the STARAN-E is the array memory, a two dimensional matrix of bits. Integrated into each array are the three basic components of the STARAN-E concept: MDA memory, a permutation (flip) network, and a set of processing elements. The arrays are the basic modules from which STARAN-E systems of varying size and power are constructed. The number of arrays in a given system is optional and is within the range of 1 to 8.

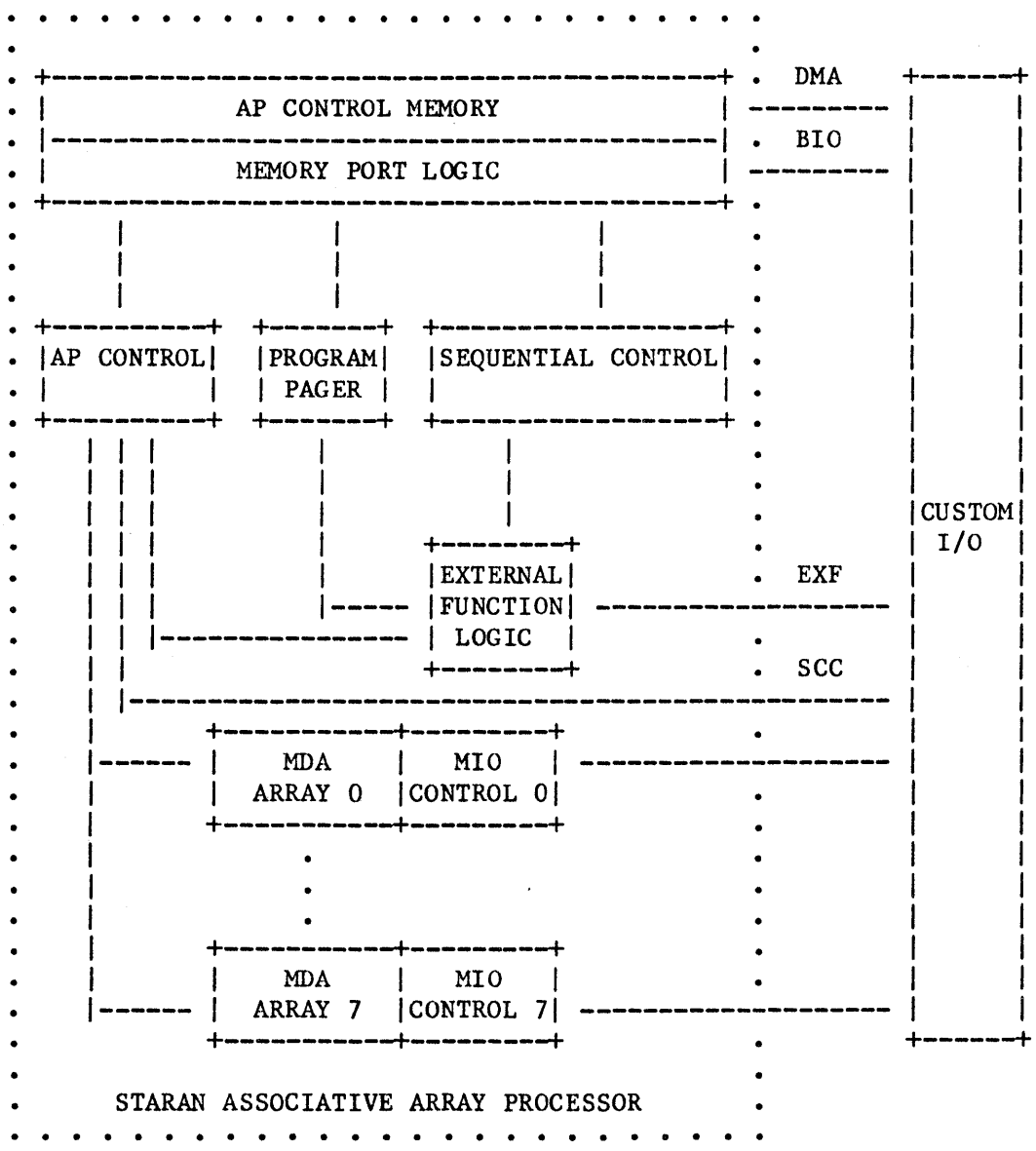


Figure 1-1. STARAN-E Block Diagram

The basic STARAN-E array module consists of a set of processing elements connected to a high bandwidth MDA memory through a permutation network. Rows, columns, or other subsets of data can be read in parallel from the memory, permuted in various ways in the permutation network, and combined with other data in the processing elements. Processed results can also be permuted and stored into memory in various ways.

Each MDA array contains 256 words, each word having a minimum of 1024 bits with up to 65,536 bits optional. Each array also contains 256 processing elements. Initially, one can think of each of the 256 PE's as being connected to one of the 256 MDA words, each PE thus having its own local 1K to 64K (K=1024) bit memory with data being transferred between PE's and memory via the flip network. However, to view each PE as being connected to its own word is needlessly restrictive. Any of the 256 PE's within an array can have access to any bit of MDA storage within that array.

CONTROL

The STARAN-E consists of 5 processing elements: AP control, multiplexed I/O control, program pager, sequential control, and the external function control which provides control and communication between the other four processing elements.

ASSOCIATIVE PROCESSOR CONTROL

AP control is directly responsible for manipulation of data within the MDA arrays under control of the program contained within the AP memory. Associative operations are coordinated and controlled by AP control.

MULTIPLEXED INPUT/OUTPUT CONTROL

Inter-array communication and input/output between arrays and external devices is controlled by the Multiplexed Input/Output Unit. There are three main modes of array I/O provided:

(a) Continuous transmit mode will transfer a block of data from an array to an IOP (input/output processor).

(b) Block transfer mode will transfer a block of data from one array to another.

(c) Exchange transfer mode is extremely flexible in that data from an array can be transferred to any and/or all of the other arrays. The only restriction is that a particular array can receive data from only one source. However, data from a particular array may be transferred to any number of other arrays. Any operation using the normal MDA instruction format can be executed in the exchange transfer mode.

PROGRAM PAGER

The program pager loads the high-speed page memories from the lower speed main memory of AP control memory. The pager performs these transfer functions independently of AP control, so that while AP control is executing a program segment out of one page memory, the pager can be loading another page memory with a future program segment.

SEQUENTIAL CONTROL

The sequential control portion of STARAN-E consists of a sequential processor (SP) having a 16k memory, a keyboard/printer, a disk drive, a paper tape reader/punch, and logic capability to interface the sequential processor with other STARAN-E elements. Sequential control is used for system software programs such as assembler, operating system, diagnostic programs, debugging, and housekeeping routines.

EXTERNAL FUNCTION CONTROL

External function (EXF) logic enables the AP control, sequential control, or an external device to control the STARAN-E operation. The external function logic facilitates coordination among the different STARAN-E elements, provides for special functions, and simplifies housekeeping, maintenance, and test functions. By issuing external function codes to the EXF logic, a STARAN-E element can interrogate and control the status of the other elements.

INPUT/OUTPUT

STARAN-E has a variety of input/output (I/O) options available. A custom I/O cabinet can be obtained as part of the basic STARAN-E system. STARAN-E can also be integrated with a variety of other computer systems. A Direct Memory Access (DMA) channel to a host-computer enables a rapid interchange of data between the systems. An I/O channel (BIO) provides access to STARAN control memory, and an external function channel permits interrupts and/or other control information to be passed between the two systems. THE STARAN Command Channel (SCC) provides a means for STARAN to control and sense external peripheral devices.

An optional Parallel Input/Output (PIO) channel, with a width of up to 256 bits per array, can also be implemented in STARAN-E. The extreme width of this channel plus its submicrosecond cycle time, gives STARAN-E an I/O bandwidth many times wider than that of a conventional computer. This PIO channel can easily accommodate the high data rates that arise in many real-time applications. Also, it is possible for STARAN-E to connect with special high-bandwidth mass-storage devices, permitting rapid retrieval, restructuring, and processing of data in a large data base.

SECTION II. ASSOCIATIVE PROCESSOR CONTROL

AP CONTROL MEMORY

GENERAL

The main function of the AP control memory is to contain the assembled AP application programs. AP control memory can also be used for data storage and as a buffer between AP control and other elements of STARAN-E. Since the AP control memory is not an integral part of the MDA array memory, AP control can overlap the AP control memory cycle with the MDA cycle time.

AP control memory is divided into several memory blocks (see Figure 1-2). Three fast memory blocks, called pages, contain the current (active) AP program segments; the slower main memory blocks contain the remainder of the AP program. A program pager is included in STARAN-E to facilitate transfer from the slow to the fast memory blocks.

Each word of AP control memory contains 32 bits of either data or instructions. The exception to this rule is page memory, which can contain instructions only, not data. Bit 0 is the left (most-significant) bit, and bit 31 is the right (least-significant) bit of each word. Each word is given a 16-bit address expressed in hexadecimal notation.

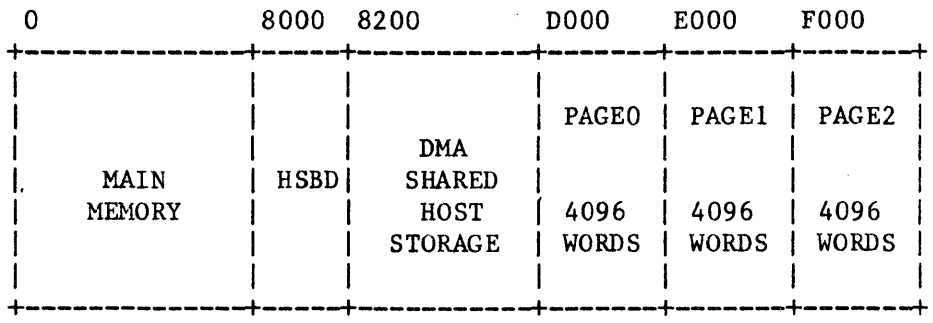


Figure 1-2. AP Control Memory Map

PAGE MEMORIES

Three page memories are included in the AP control memory: Page 0, Page 1, and Page 2. Page memories use fast, bipolar, solid-state elements that are volatile. Each page contains 4096 words in the basic STARAN-E configuration. The page memories can be doubled to 8192 words each on an optional basis.

Page 0 may contain a library of routines that require fast execution, such as arithmetic subroutines. Pages 1 and 2 can be used in ping-pong fashion, with the AP control executing instructions out of one page while the other is being loaded by the program pager. This permits the programmer to use the faster memory for certain segments of the program or the entire program if fast execution is required.

Each page memory has a port switch that connects it to one of three buses. The port switch is controlled by external function codes. At any given time, a page memory is connected to 1) the instruction bus, which allows AP control to read instructions from the page; 2) the pager bus, which allows the program pager to load the page; or 3) the sequential control bus, which allows sequential control to read items from the page. If one of these buses should try to access a word in the page memory while the port is set to another bus, a hangup results. Hangups, which are sensed by error detectors, cause an interrupt in sequential control.

HIGH SPEED DATA BUFFER

The High Speed Data Buffer (HSDB), like the page memories, uses fast, bipolar, solid-state elements and is volatile. In the basic configuration of STARAN-E it contains 512 words. As an option, its size can be doubled to 1024 words. All buses that can access AP control memory can access the HSDB, thus making the HSDB a convenient place to store data and instruction items that need to be accessed quickly by the different STARAN-E elements.

A port priority switch on the HSDB resolves any conflict among buses. Each memory cycle is given to the highest priority bus requesting an HSDB address at that time, while other buses requesting HSDB addresses wait for the next memory cycle. Priorities among buses are as follows:

1) An I/O bus to I/O cabinet (highest priority)

2) Sequential control bus

3) AP control instruction bus

4) Program pager bus

5) AP control data bus (lowest priority)

MAIN MEMORY

The main memory uses nonvolatile core storage. In the basic configuration it contains 32,768 words (hexadecimal addresses 0000 through 7FFF).

Like the High Speed Data Buffer (HSDB), the main memory is accessible to all buses that can access AP control memory (through a priority port switch that gives each memory cycle to the highest priority bus requesting a main memory address). The priorities of the buses are the same as those for the High Speed Data Buffer.

The main memory is used to contain the AP control programs. Because the main memory is slower than the page memories, it is recommended that program segments be moved into the page memories for execution. Also, since the main memory is accessible to all buses having access to AP control memory, it is also useful as a buffer for data.

DIRECT MEMORY ACCESS

A block of AP control memory addresses is reserved for the Direct Memory Access (DMA) channel to external memory. In the basic configuration this block can contain up to 36,392 addresses (hexadecimal addresses 8200 through CFFF). If any page memory or the High Speed Data Buffer is expanded, the DMA block may be reduced.

All buses accessing AP control memory can access the DMA block. A priority port switch resolves any interbus conflicts, giving each access cycle to the highest priority bus requesting a DMA address at the time. Priorities among buses are the same as those for the High Speed Data Buffer.

ADDRESSING

General

Each AP control memory word contains 32 bits of either data or instructions. Each word is given a 16-bit address.

Main Memory

The main memory contains 32,768 words of memory in the basic configuration. These words are assigned hexadecimal addresses 0000 through 7FFF.

Certain words in the main memory are dedicated to special purposes. For the basic configuration of STARAN-E, these locations are as follows:

Hexadecimal Address	Dedicated Usage
0000	First AP control instruction when activated
0001	AP control interrupt 1
.	.
.	.
.	.
000F	AP control interrupt 15

When the AP control becomes active, the instruction at 0000 is the first instruction executed. It is usually a branch instruction to the beginning of the first program segment to be executed.

Page Memories

The three page memories, 4096 words each, that are included in the AP control memory are designated Page 0, Page 1, and Page 2. In the basic configuration, Page 0 contains hexadecimal addresses D000 through DFFF; Page 1 contains hexadecimal addresses E000 through EFFF; and Page 2 contains hexadecimal addresses F000 through FFFF.

High Speed Data Buffer

The High Speed Data Buffer contains 512 words of memory in the standard configuration, but can be doubled to 1024 words on an optional basis. In the basic configuration, the High Speed Data Buffer contains hexadecimal addresses 8000 through 81FF.

Direct Memory Access

A block of AP control memory addresses is reserved for the Direct Memory Access (DMA) channel to access the memory of a host computer. This block can contain up to 20,480 addresses which are assigned hexadecimal addresses 8200 through CFFF. This block can be reduced if page memories or the High Speed Data Buffer is increased in size.

AP CONTROL MEMORY SUMMARY

Table 1-1 summarizes the AP control memory characteristics. The characteristics of each memory and the connection of each bus to each section are given.

Table 1-1. AP Control Memory Characteristics

		MEMORY BLOCKS					
ITEM		CORE	PAGE 0	PAGE 1	PAGE 2	HSDB	DMA
Implementation		Mag. Core	Bipolar	Bipolar	Bipolar	Bipolar	*
Volatile		No	Yes	Yes	Yes	Yes	*
Number of Words		32,768	4096	4096	4096	512	19,968
Bits Per Word		32	32	32	32	32	32
First Octal Address		000000	150000	160000	170000	100000	101000
Last Octal Adress		777777	157777	167777	177777	100777	147777
First Hex Address		0000	D000	E000	F000	8000	8200
Last Hex Address		7FFF	DFFF	EFFF	FFFF	81FF	CFFF
Port Switch		Priority	Ext Fcn	Ext Fcn	Ext Fcn	Priority	Priority
B U S	Buffered I/O	RW	-	-	-	RW	RW
	AP Control Data	RW	-	-	-	RW	RW
	AP Control Instr.	R	R	R	R	R	R
	Program Pager	R	W	W	W	R	R
	Sequential Control	RW	R	R	R	RW	RW

- \- Bus cannot access memory
- R Bus can only read from memory
- W Bus can only write into memory
- RW Bus can both read and write into memory
- * Depends on customized use of DMA

AP EXECUTION CONTROL

GENERAL

The major function of the AP control is to control the STARAN-E MDA arrays. AP control fetches instructions from the AP control memory. A program counter contains the address of the instruction, while an instruction register contains the instruction itself. Some instructions perform array operations, while others perform AP control functions. AP control consists of the following elements:

1) Instruction Register
2) Program Control
3) Block Transfer Control
4) Common Register
5) Field Pointers and Length Counters
6) Response Store Control
7) Array Control
8) General Registers

INSTRUCTION REGISTER

The instruction register contains the instruction being executed. The instruction loaded into the instruction register is received from AP control memory via the instruction bus. Parity is checked at the instruction register. The instruction register contains 32 bits which are numbered from 0 to 31 with bit 0 at the left. Portions of the instruction register are used as a direct source of data or addresses as a function of the instruction being executed.

PROGRAM CONTROL

The sequence in which instructions are obtained from AP control memory is controlled directly by the program control logic. The program control logic consists of the following: the Program Counter, the Start Loop Marker, the End Loop Marker, the Comparator, and the Status Register.

Program Counter

The Program Counter contains the address of the instruction being read from control memory. It is a 16-bit counter incremented by AP control. The Program Counter may be loaded from the bus logic; e.g., a branch instruction loads an address. The contents of the Program Counter form bits 0 through 15 of the program status word.

Start Loop Marker

The Start Loop Marker is used to store the address of the first instruction immediately following a loop instruction. The Start Loop Marker is a 16-bit register loaded directly from the Program Counter at the start of an instruction loop. It is loaded into the Program Counter when the last instruction of the loop has been executed and the loop is to be repeated.

End Loop Marker

The End Loop Marker is used to store the address of the last instruction of a loop. The End Loop Marker is a 16-bit register loaded from the rightmost 16 bits of the loop instruction.

Comparator

The Comparator compares the address contained in the end loop marker with the address in the Program Counter. The Comparator is a full 16-bit Comparator, the output of which is transmitted to control as an indication that the end of a loop has been reached. The control then loads the Start Loop Marker contents into the Program Counter if the loop is to be repeated.

Program Status Register

The program status register consists of three basic parts: (1) a 16 bit program counter (2) a 4 bit interrupt mask for the 15 AP control interrupts, and (3) a 4 bit condition code consisting of an overflow bit, a carry bit, a negative result bit, and a zero result bit.

Bus Logic

The bus logic provides a common data path for all pertinent registers of AP control and the data bus from control memory. The bus is 32 bits wide. Registers of less than 32 bits are grouped to form a 32-bit word. Details of registers connected to the bus and register grouping are shown in Figure 1-4.

Data transmitted via the bus logic passes through the bus shift logic. The bus shift logic shifts the bus word left end around by either 0, 8, 16, or 24 bit positions. The amount of shift is controlled by the instruction moving the data. Data received from AP control memory is checked for correct parity as it passes the bus shift logic. Data stored in the control memory has an odd parity bit generated by the bus shift logic.

BLOCK TRANSFER CONTROL

Data Pointer Register

The Data Pointer register contains the control address for the data bus for block transfers. The Data Pointer is a 16-bit counter. The Data Pointer can be stepped with each transfer within a data block.

Block Length Counter

The Block Length counter, a 16-bit decrementing counter, controls the length of a data block transfer.

COMMON REGISTER

The Common register contains the argument for a search operation performed upon the MDA arrays, the input data to be stored into an array, or the output data loaded from an array. The Common register contains 32 bits which are numbered from 0 to 31. Bit 0 is the left (most-significant) bit and bit 31 the right (least-significant) bit. The search argument or array input data is loaded via the bus logic. The array output data is loaded through a mask generated by the mask

generator. The use of the mask allows formatting of an output word from noncontiguous data in an array.

The mask generator generates a mask pattern to be used in loading array output data into the Common register. The mask enables data to be loaded for a number of contiguous bits. The mask generator requires the bit addresses of the most and least significant bits to be loaded. All bits between and including these limits are loaded, while those outside these limits are unaltered.

FIELD POINTERS AND LENGTH COUNTERS

Field pointers generally contain bit slice or word addresses in the MDA array operations. The field length counters control the number of bits to be operated on in sequence. There are three field pointers and two field length counters. In addition, one register is used as a temporary pointer or can contain a shift code for certain array operations. A selector is used to route either field pointer 1, 2, or 3 or the address field of the instruction register to the array. These registers are 8 bits in length and their contents range from 0 to 255. When attempting to increment above 255, the register will return to zero; when attempting to decrement below zero, the register contents become 255. The field length counters can only be decremented.

Field Pointer 1

Field Pointer 1 plus base register RC specifies an array word or bit address for an MDA operation. FP1 is also used to specify the address of a selected bit of the Common register to be used for a search instruction. Field Pointer 1 (like field pointers 2 and 3)is an 8-bit counter. In addition, as a result of the resolve operation, Field Pointer 1 will be loaded with the number of the array module containing the first responder (i.e., first selected word whose Y bit is set to one).

Field Pointer 2

Field Pointer 2 plus base register RD specifies an array word or bit address for an MDA array operation. Also, as a result of a resolve operation, FP2 will be loaded with the word address of the first responder in the array specified by FP1.

Field Pointer 3

Field Pointer 3 plus base register RE specifies an array bit or word address.

Field Pointer E

Field Pointer E is an 8-bit counter. It can be used for temporary storage of an array bit or word address, or it can contain a shift constant for certain MDA operations.

Field Length Counter 1

Field Length counter 1 and Field Length counter 2 are 8-bit counters. The length counters can only be decremented. When the contents of a field counter become zero, a signal is sent to AP control for test purposes. This permits the program sequence to be altered by a branch if a field length counter becomes zero. Field Length counter 1 contains the number of cycles for a loop instruction.

Field Length Counter 2

Field Length counter 2 may be used to control the cycles of a major instruction loop, such as multiply fields.

RESPONSE STORE CONTROL

The response store control logic generates the control signals required by the MDA arrays and buffers them to insure correct timing at the response store. The response store control consists of the control line conditioner and the control line buffer.

Control Line Conditioner

The control line conditioner generates the control signals required to manipulate the response store. These signals are generated as a function of the instruction register, a selected bit of the Common Register and the inclusive-OR output from the resolver.

Control Line Buffer

The control line buffer contains the control signals transmitted to the MDA arrays which allows overlapping of array instructions.

MDA ARRAY CONTROL

Control lines to the MDA arrays not generated by the response store control are generated by the array control. The array control logic selects which arrays are to be used and controls such things as bit/word mode, store masked, and shifting.

Array Select Register

The Array Select register establishes which array modules are to be enabled for an operation. The Array Select register is 32 bits wide. Each bit position controls one array. Bit 0 corresponds to Array 0, and a 1 in a bit position enables the corresponding array. The Array Select register contents are also used by the resolver logic.

Array Access

The array access logic selects either the Array Select register or Field Pointer 1 to generate the array enable signals. When Field Pointer 1 is selected, the five right-most bits of the pointer specify the one array to be enabled. This is done without modifying the contents of the Array Select register. Such operations as loading one item of data from an array or storing one item of data into an array enables only one of the MDA arrays. When more than one array is involved in an operation, the Array Select register is used to select the arrays to participate.

Array Address Mode

The array address mode is determined by the value in the most significant 8 bits of one of the six array address base registers. Table 1-2 illustrates how a particular base register is selected.

GENERAL REGISTERS

There are 16 general 32-bit registers, R0 through RF. Fourteen of these registers have various special uses.

R0-R7

Registers R0 through R7 are used as index or branch and link registers. R7 is also used by the special branch and link instruction whose mnemonic is CAL.

R8, RB-RF

These six registers serve as array address base registers. Each of these registers contains a 16-bit array base address and an 8-bit storage mode. The base register format is defined in Figure 1-3.

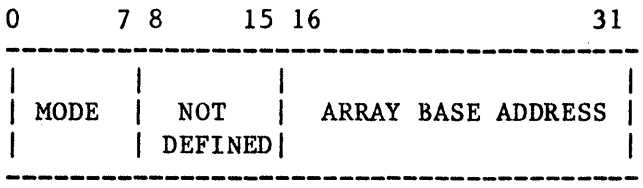


Figure 1-3. Base Register Format

Table 1-2. Array Base Register Selection

ADDRESSING MODE	INSTRUCTION BIT 5	BIT 7	BIT 8	BIT 9	ADDRESS BASE	ADDRESS* DISPLACEMENT
DIRECT	0	0	-	-	RB	I(8-15)
INDIRECT (FP1)	0	1	0	0	RC	FP1
INDIRECT (FP2)	0	1	0	1	RD	FP2
INDIRECT (FP3)	0	1	1	0	RE	FP3
INDIRECT (FP12)	0	1	1	1	RF	FP2
DIRECT	1	0	-	-	RB	I(8-15)
INDIRECT (FP1)	1	1	0	0	R8	FP1
INDIRECT (FP2)	1	1	0	1	R8	FP2
INDIRECT (FP3)	1	1	1	0	R8	FP3
INDIRECT (FP12)	1	1	1	1	R8	FP2

*I(8-15) refers to bits 8 through 15 of the MDA machine language instruction format. These registers, along with R9 and RA, may be used as general purpose accumulators.

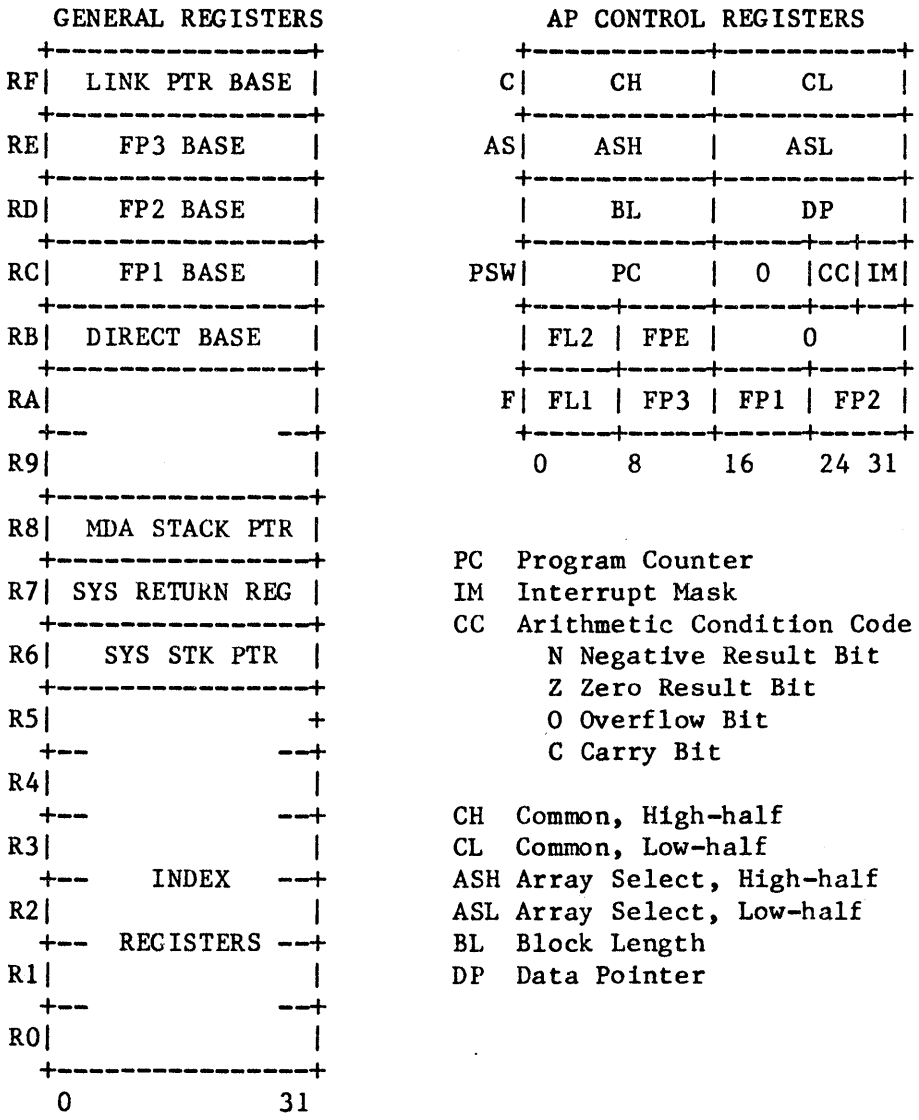


Figure 1-4. STARAN Register Map

MDA MEMORY

GENERAL

Each STARAN-E array consists of 256 words (rows) with an optional number of bits per word. The basic array contains two types of memory: (1) a high speed bipolar section, and (2) a slower speed MOS section. An array may consist of up to 7K bits per word of high speed memory (K=1024). The amount is variable in 1K increments. An array may contain a maximum of 64K bits per word. The slower speed MOS portion is variable in 4K increments. Associated with each word of array memory is a 3-bit processing element consisting of three 1-bit MDA registers, M, X, and Y, i.e., one bit for each word of array memory. Also included in each array is a permutation (flip) network to facilitate data manipulation and interword communication, and a 256 bit wide data path to a Multiplexed Input/Output Unit. The MDA array consists of three basic components: the array memory, the permutation (flip) network and the MDA registers. As many as 8 MDA arrays may be contained within one system. Any combination from 1 to 8 arrays may be selected and operated on concurrently.

Each basic array is organized as a matrix of 256 words by 9216 bits of solid-state storage. By use of a special organization within the array, access may be made in either the bit or word direction. Consider an array as organized into square segments of 256 words by 256 bits per word. The basic array would contain 36 such segments. A word of 256 bits or a bit slice, bit n of all 256 words, of any segment may be accessed. The permutation network can shift and rearrange bits in the response store portion of the MDA. The response store portion of the array consists of 256 response store elements. The M, X, and Y registers (256 bits each) may be used as temporary storage of data loaded from the array or may contain the data to be stored into the array. The X and Y registers can perform logic functions on array data. The M (MASK) register is used as a mask to select which words of array memory participate in an array store operation. The Y register is used as a responder by the resolver in a search type operation. The X register is used as temporary storage when performing logic operations. The X and Y registers can also perform logical functions simultaneously with the load operations.

MDA array input and output may be either 32 bits via the Common Register or 256 bits via the MDA registers or MIO.

ADDRESSING

Addressing within the MDA arrays is represented in hexadecimal notation. The four basic areas to be addressed are arrays, words, bit columns, and fields.

Arrays

Addressing of arrays is accomplished by an address from 0, the first array, to 7, the last array. The number of arrays within a system, therefore, ranges from 1 to 8 as dictated by the requirement placed on the system. The internal organization and addressing of all arrays is identical.

Words

Addressing of a word within an array is accomplished by an address from 0, the first word, to 255, the last word. The basic array word consists of 9216 bits. Any 256-bit section may be accessed in parallel or divided into eight fields of 32 bits each and accessed via the Common Register.

Bit Columns

Addressing of a bit position within an array is accomplished by an address from 0, the most-significant (left), to 9216, the least-significant (right) bit position. Bit (n) of all words is accessed by using address (n).

Fields

Addressing of a particular 32 bit field of an array word within an array segment is accomplished by an address from 0, the most-significant (left) field, to 7, the least-significant (right) field. Each field contains 32 contiguous bits within the word being addressed. The most-significant field starts at the most-significant bit position.

Access Mode

The array memory can be accessed horizontally, vertically or any combination thereof. Control of the access mode is via an 8-bit code. There are 256 access modes, each giving a unique mapping of data within an array.

The most commonly used access modes are mode 0 and its complement, mode FF (255). Mode 0 is the bit-slice mode where one bit of each array word is accessed. This mode is used for mapping single bit flags and arithmetic fields. Mode FF is word mode and accesses 256 bits of one array word. Other modes are intermediate to the bit-slice and word modes since they access some bits of some words. Some useful modes are defined in Table 1-3.

Table 1-3. Access Modes

Mode (Hex)	Mode (Binary)	Number of selected rows in each array	Number of accessed bits in the selected rows
00	00000000	256	1
01	00000001	128	2
03	00000011	64	4
07	00000111	32	8
0F	00001111	16	16
1F	00011111	8	32
3F	00111111	4	64
7F	01111111	2	128
FF	11111111	1	256
FE	11111110	2	128
FC	11111100	4	64
F8	11111000	8	32
F0	11110000	16	16
E0	11100000	32	8
C0	11000000	64	4
80	10000000	128	2

OPERATIONS

The operations performed within the associative array can be grouped into the following categories: load, store, logical, resolve, and exchange.

Load

Data loaded from the array memory can be sent to the 256-bit output bus or loaded into one of the MDA registers. By specifying a field address (section), a field (section) of one word may be loaded from the array memory for output over the 32-bit output bus to the Common register. Data is loaded from the array memory in a mode as selected by the particular base register. One MDA register may be loaded with the contents of another MDA register. All loads from the array storage are nondestructive. Logic may be performed on the X and/or Y registers simultaneously with the load operation.

Store

Data is stored into the array memory in a mode as selected by the particular base register. Data to be stored may come from the MDA registers, the 32-bit input bus, or the 256-bit input bus (MIO). The data, regardless of source, may be stored through a mask contained in MDA register M.

Logical

To perform operations such as exact-match search and add fields, certain logical operations must be performed on the data in the MDA registers. Data loaded from the array memory may be logically combined with the current contents of the MDA registers to accomplish these operations. Data may also be transferred among the MDA registers with logic functions applied to the X and/or Y registers.

Resolve

The array number and the address of the first Y register bit set is continuously resolved. This address and the inclusive-OR of all Y register elements are made available to control. This address (which is stored in FP1 and FP2) may be used in succeeding operations.

Exchange

Items of 256-bit data may be transferred between array modules. All of the normal MDA instructions may also be performed in this mode.

SECTION III. PROGRAM PAGER

GENERAL

The function of the program pager is to transfer program segments from the main memory to the page memories.

Under normal programming practice, the pager is activated by AP control when a new program segment is to be transferred to a page memory. The program pager transfers the segment one word at a time at a rate dictated by the source memory, while AP control executes instructions from previously loaded segments. When the pager completes the transfer, it restores the page memory port switch to the AP instruction bus and halts.

The program pager contains three registers (see Figure 1-5). The GET address register contains a 16-bit AP control memory address. If the pager is in the midst of moving data, the GET address points to the memory location containing the next source word to be moved. At other times the GET address register acts like a program counter, pointing to the location of the next pager instruction to be executed.

The PUT address register holds a 16-bit AP address. It points to the memory location into which the next destination word is to be put during a data move operation.

The 14-bit COUNT register holds the number of words still to be transferred during a transfer operation.

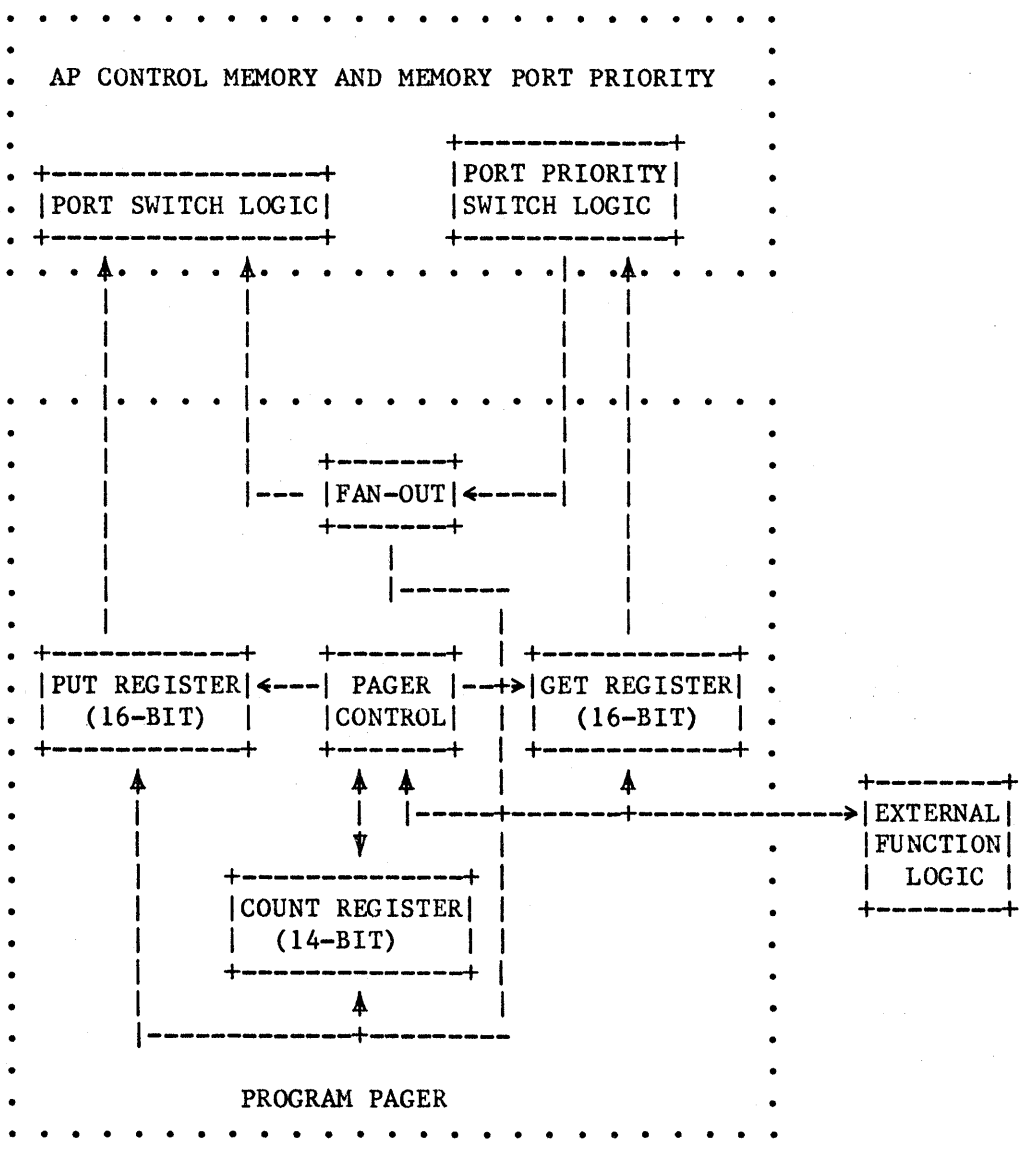


Figure 1-5. Program Pager Block Diagram

OPERATION

Pager transfer speed is governed by the cycle time of the source memory which can be slower than AP control execution speed. To keep STARAN-E

from being "pager bound", AP programs should be segmented carefully. An ideal program segment will contain enough long instructions, subroutine calls and loops, etc., so that before AP control leaves the segment, the pager has enough time to get the next segment loaded in a page memory. If all segments are ideal, AP control will never wait for the pager.

Each page memory has a port switch to prevent AP control from starting execution in a segment before the pager has loaded it. A premature attempt to execute instructions in a page memory that is still being loaded by the pager will be delayed until the pager has switched the port switch to the instruction bus.

Pager operation is initiated by an external function code that loads the GET address register. With one external function, the current pager operation can be stopped in midstream and a new operation started. For instance, suppose from program segment 1 AP control jumps to either program segment 2 or to program segment 3, depending on some condition. Most of the time it jumps to segment 2. In this case, AP control can initiate the loading of segment 2 as it begins executing segment 1, so that little or no time is lost in waiting for the pager. In the rarer case, when AP control jumps to segment 3, it can stop loading segment 2 in midstream, if necessary, and start the loading of segment 3.

When the program pager is running (busy), it fetches pager commands and source data from AP control memory. Its commands are sufficiently general to permit flexible pager programming.

SECTION IV. EXTERNAL FUNCTION CONTROL

GENERAL

Numerous hardware functions are under the control of external function (EXF) logic. These include page memory port switches, interlocks, AP and sequential control interrupts, AP control and program pager activity control, and resets and clears. Control and status sensing of these functions are accomplished by issuing 19-bit external function commands to EXF logic and receiving 1-bit sense signals in return. Three elements of STARAN-E can issue EXF commands: AP control, program pager, and sequential control (see Figure 1-1). EXF logic is expandable to allow receipt of EXF commands from the Custom I/O Unit and control of other hardware functions in the Custom I/O Unit. A resolver in EXF logic allows only one EXF command to be treated at a time.

The resolver in EXF logic resolves conflicts among the four elements issuing function codes. One function code at a time is accepted by EXF logic. The interrogation and/or control called for is performed and then another function code is accepted if one is present. A function code can interrogate and control an element in one operation without interference from another function code.

The classes of function codes are as follows: page memory port switches, interlocks, program pager, error control, AP control interrupts, sequential control interrupts, AP control activity, AP control loop indicator, and resets and clears. These classes are described in the following sections.

PAGE MEMORY PORT SWITCHES

Each page memory has a port switch that connects it to one of three buses. The port switch is controlled by external function codes. At any given time, a page memory is connected to 1) the instruction bus, which allows AP control to read instructions from the page; 2) the pager bus, which allows the program pager to load the page; or 3) the sequential control bus, which allows sequential control to read items from the page. If one of these buses should try to access a word in the page memory while the port switch is set to another bus, a hangup results. Hangups, which are detected by error detectors, cause an interrupt to sequential control.

This class of external function codes allows interrogation and control of the port switches. The current state of the selected port switch is interrogated and, depending on whether it is on the sequential bus, the instruction bus, or the pager bus, one of the sense bits of the function code is returned. The current switch status is used to select the field of the function code containing the new switch setting.

INTERLOCKS

The EXF logic contains 64 stored bits called interlocks. These bits have no predetermined meaning; software can assign functions to the interlocks and use them for various purposes. They are useful for passing signals among STARAN-E elements, such as what programs should be executed, etc. Function codes allow the current state of an interlock to be sensed and a new state to be entered in one operation.

Sixteen interlocks (hexadecimal addresses 00 through 0F) can be controlled manually by panel switches and are displayed via lights on the interlock panel to facilitate communication with an operator. The other 48 interlocks (hexadecimal addresses 10 through 3F) can only be controlled and sensed via software.

Interlocks are volatile so their states are lost whenever power is lost.

PROGRAM PAGER FUNCTIONS

Certain external function codes allow program pager operation to be initialted, halted, modified, and sensed.

The program pager has two states: off and busy. A function code allows the state to be sensed and changed. If the current pager state is off, the function code can either leave it in the off condition or turn it on, in which case (if it is operative) the pager will become busy.

If the current pager state is busy, the function code can either turn the pager off or leave it in the busy condition.

ERROR CONTROL FUNCTIONS

Error detectors are included in various elements of STARAN-E to sense hardware faults and program errors. Each error detector sets an error indicator when an error is detected. Each error indicator is given a number by which it may be sensed, set and/or reset by function codes. If any error indicator is set, an interrupt to sequential control is generated. Also, certain error indicators will make the program pager inoperative and others will make AP control inactive. Since an error indicator can be set by a function code, errors can be simulated in order to check out error handling routines, etc.

Error indicators in the basic STARAN-E configuration are shown in Table 2-4.

ASSOCIATIVE PROCESSOR CONTROL INTERRUPTS

The AP control interrupt is a class of function codes used to sense, set, and reset the state of 15 programmable interrupts to AP control. AP control interrupts are given hex addresses 01 (lowest priority) to 0F (highest priority).

Bits 28 through 31 of the program status word (PSW) in AP control contain an interrupt mask. AP control accepts interrupt n if the following conditions are satisfied: 1) AP control is active and at an interruptable point; 2) the interrupt mask is less than n; 3) no interrupt of higher priority is set; and 4) interrupt n is set. When AP control accepts interrupt n, it fetches the next instruction from hex address 0000+n (without disturbing the content of its program counter). Normally, this instruction is a swap PSW instruction which saves the old PSW and loads the new PSW, causing AP control to enter an interrupt routine. The interrupt mask of the new PSW must be n or greater to prevent AP control from accepting interrupt n again until the interrupt routine is complete and has issued an EXF command to reset interrupt n. If the new interrupt mask is less than n, error 00 will be generated.

SEQUENTIAL CONTROL INTERRUPTS

The sequential control interrupt class of function codes can sense, set, and reset the state of eight programmable interrupts to sequential control. Table 1-4 shows the vector addresses of the sequential control interrupts, together with the priority levels.

Table 1-4. Sequential Control Interrupt Vector Addresses

Vector Address (Octal)	Priority Level	Relative Priority
334	7	High
330	7	↓
324	6	↓
320	6	↓
314	5	↓
310	5	↓
304	4	↓
300	4	Low

When the sequential control processor priority is set to n, all requests for interrupts at level n and below are ignored.

Bits 19 through 27 of the instruction select one of the sequential control interrupts. The current state of the selected interrupt is sensed and a sense bit is returned. Also, a new state may be assigned to the interrupt.

AP CONTROL ACTIVITY

An external function code senses and controls the AP control activity. AP control has two states: inactive and active. In the active state it fetches instructions from AP control memory and exercises the MDA arrays.

When switched from the inactive state to the active state, AP control fetches its first instruction from hexadecimal address 0000 without disturbing the program counter. It could be a no-op, which would cause AP control to continue with its previous program, or it could be a swap program status word instruction, which would cause the old status of AP control to be saved and a new program entered.

AP CONTROL LOOP INDICATOR

When AP control executes a loop-type instruction, a loop indicator is set in the loop indicator function code and remains set until all repetitions of the loop are completed. The indicator is neither disturbed by changes in activity nor by interrupts. Execution of a loop instruction when the loop indicator is still set from a previous loop is illegal. Function codes allow the loop indicator to be sensed and/or reset. Resetting of a currently set loop indicator causes AP control to forget the loop instruction that set it, even if all repetitions of the loop were not completed.

RESETS AND CLEARS

Other external function codes are reserved for selective resetting and clearing of various STARAN-E registers and status indicators. They are used for clearing any hangup conditions that may arise.

SECTION V. SEQUENTIAL CONTROL

GENERAL

The sequential control device used in STARAN-E consists of a sequential processor (SP) with 16K of memory, interface logic to connect the SP to other STARAN-E elements, and peripheral units, which include a keyboard/monitor, disk drive, card reader, paper tape reader/punch, and line printer. The sequential controller provides:

1) A means to initially load AP memory

2) A communication link between the operator and STARAN-E for on-line control and monitoring

3) Off-line capabilities for assembling and debugging STARAN-E programs

4) Control for STARAN-E maintenance and diagnostic program routines

5) Housekeeping capabilities

SEQUENTIAL PROCESSOR ARCHITECTURE

The sequential processor is a 16-bit general-purpose minicomputer using two's complement arithmetic. The 16,384 16-bit words (32,768 8-bit bytes of memory) have octal addresses 000000 through 077777. All communication between system components is performed on a single high-speed bus. There are eight general-purpose registers, which can be used as accumulators, index registers, or address pointers, and a multilevel automatic priority interrupt system.

The sequential processor features include:

1) Single and double operand addressing

2) 16-bit word and 8-bit byte addressing

3) Simplified list and stack processing through auto-address stepping (auto-incrementing and auto-decrementing)

4) Eight programmable general-purpose registers

5) Data manipulation directly within external device registers

6) Addressing of device registers using normal memory reference instructions

7) Asynchronous operation of SP memory, central processor, I/O processor, and I/O devices

8) Hardware interrupt priority structure for devices peripheral to the SP

9) Automatic interrupt indentification without device polling

10) Direct addressing of the SP 16K words or 32K bytes

A single common path connects the SP memory and all peripherals. Addresses, data, and control information are sent along the 56 lines of the bus. All instructions that can be applied to data in the SP memory can be applied as well to data in peripheral device registers.

The common path lines are bi-directional. A peripheral device register can be either read or set by the SP or other peripheral devices; thus the same register can be used for both input and output functions.

Communication between two devices on the common path is in the form of a master-slave relationship. A controlling device (termed the bus master) controls the bus when communicating with another device on the bus (termed the slave).

Common path communication is interlocked so that for each control signal issued by the master device, there must be a response from the slave device in order to complete the transfer. The maximum transfer rate on the path is one 16-bit word every 750 nanoseconds or 1.3 million 16-bit words per second.

SEQUENTIAL CONTROL INTERFACE

GENERAL

Communication between sequential control and other STARAN-E elements is accomplished using certain interrupt vector addresses. Four forms of communication are described below.

1) Direct access to AP control memory: Words in AP control memory are given sequential control bus addresses to facilitate transfer of data and instructions between AP control and sequential control.

2) Register Readout: Certain registers in STARAN-E and in its associated I/O unit, where applicable, can be read by sequential control. This facilitiates program debugging and hardware maintenance and tests.

3) External functions: External function codes can be transmitted to the external function logic and sense bits received. This allows sequential control to activate and deactivate AP control, issue interrupts, and perform many housekeeping functions.

4) Interrupt acceptance: Other elements of STARAN-E can issue interrupts to sequential control by issuing certain function codes to the external function logic. Also, when errors such as parity errors are detected, a sequential control interrupt is issued. This function allows real-time control of the resources in sequential control.

Detailed descriptions of these forms of communication are given in the next four subsections.

DIRECT ACCESS TO AP CONTROL MEMORY

Sequential control can read and write AP control memory data. For this purpose AP control memory is considered to be divided into 16 groups. Each group contains 4096 32-bit words (16,384 bytes). Group 0 contains AP octal addresses 000000 through 007777. Group 1 contains addresses 010000 through 017777, etc. Group 15 contains addresses 170000 through 177777.

To access any of the 16,384 bytes of a group, a sequential control program should first store the group number in a special 4-bit register called the Group register (GRP), whose sequential octal address is 164064. This register can be read, written, and incremented by sequential control.

The 16,384 bytes of the selected group can be accessed using sequential control addresses 100000 through 137777. Addresses 100000 through 100003 access bytes of the first AP control memory word in a group, addresses 100004 through 100007 access bytes of the second memory word, and so on. Addresses 137774 through 137777 access bytes of the last word in the group.

Addressing of bytes within an AP control memory word takes place from right to left; e.g., address 100000 accesses the rightmost byte and address 10003 accesses the leftmost byte of the first memory word in the group.

Addressing of 16-bit halves of AP control memory words also takes place from right to left, using even addresses; e.g., address 100000 accesses the right half and address 100002 accesses the left half of the first memory word in the group.

Words in Page 0, Page 1, Page 2 memories can only be read by the sequential controller. Other AP control memory words can be both read and written by the sequential controller.

REGISTER READOUT

To assist personnel during program debugging, hardware maintenance, or test operations, certain STARAN-E registers and PIO registers can be read by sequential control by reading certain bus addresses. The sequential control addresses of the registers are shown in Table 1-5. The octal addresses for these registers are in the range of 164000 to 164777. The majority of these addresses are read-only registers. The remainder can be both written and read. Some of the read-only addresses have special functions to facilitate firmware debug of the ROM programmed Register Processing Unit. Load operations into all read-only registers are ignored. AP control must be inactive when reading AP registers or a sequential hangup error will be generated.

Table 1-5. Sequential Control Readout Registers

OCTAL ADDRESS	REGISTER SYMBOL	REGISTER	LENGTH IN BITS	ACCESS MODE
		AP CONTROL REGISTERS		
164000	CL	Common (low order)	16	R
164002	CH	Common (high order)	16	R
164004	ASL	Array Select (low order)	16	R
164006	ASH	Array Select (high order)	16	R
164010	DP	Data Pointer register	16	R
164012	BL	Block Length counter	16	R
164014	IMASK	Condition Code & Interrupt Mask	8	R
164016	PC	Program Counter	16	R
164022	FPE	Field Pointer E	8	R
164023	FL2	Field Length counter 2	8	R
164024	FP2	Field Pointer 2	8	R
164025	FP1	Field Pointer 1	8	R
164026	FP3	Field Pointer 3	8	R
164027	FL1	Field Length counter 1	8	R
164030	EFS	External Function register	16	R/W
164032	EFB	External Function register	16	R/W
164034	DAL	Sequential bus data (low order)	16	R
164036	DAH	Sequential bus data (high order)	16	R
164040	GET	Pager GET address	16	R
164042	PUT	Pager PUT address	16	R
164044	CNT	Pager word COUNT	16	R
164050	ELM	End Loop Marker	16	R
164052	SLM	Start Loop Marker	16	R
164054	PFMT	Performance monitor timer	16	R
164056	PFMC	Performance monitor counter	16	R
164060	IRL	AP instruction register (low order)	16	R
164062	IRH	AP instruction register (high order)	16	R
164064	GRP	Group register	8	R/W
164065	HOME	Home register	8	R

OCTAL ADDRESS	REGISTER SYMBOL	REGISTER	LENGTH IN BITS	ACCESS MODE
		AP BRANCH AND LINK/INDEX REGISTERS		
164100	ROL	R0 (low order)	16	R
164102	ROH	R0 (high order)	16	R
164104	R1L	R1 (low order)	16	R
164106	R1H	R1 (high order)	16	R
164110	R2L	R2 (low order)	16	R
164112	R2H	R2 (high order)	16	R
164114	R3L	R3 (low order)	16	R
164116	R3H	R3 (high order)	16	R
164120	R4L	R4 (low order)	16	R
164122	R4H	R4 (high order)	16	R
164124	R5L	R5 (low order)	16	R
164126	R5H	R5 (high order)	16	R
114130	R6L	R6 (low order)	16	R
164132	R6H	R6 (high order)	16	R
164134	R7L	R7 (low order)	16	R
164136	R7H	R7 (high order)	16	R
		AP ARRAY BASE REGISTER (BIT 5=1)		
164140	R8L	Array base (low order)	16	R
164142	R8H	Array base (high order)	16	R
		AP GENERAL PURPOSE ACCUMULATOR REGISTERS		
164144	R9L	R9 (low order)	16	R
164146	R9H	R9 (high order)	16	R
164150	RAL	RA (low order)	16	R
164152	RAH	RA (high order)	16	R

OCTAL ADDRESS	REGISTER SYMBOL	REGISTER	LENGTH IN BITS	ACCESS MODE
		AP ARRAY BASE REGISTERS (BIT 5=0)		
164154	RBL	Direct address base (low order)	16	R
164156	RBH	Direct address base (high order)	16	R
164160	RCL	FP1 base (low order)	16	R
164162	RCH	FP1 base (high order)	16	R
164164	RDL	FP2 base (low order)	16	R
164166	RDH	FP2 base (high order)	16	R
164170	REL	FP3 base (low order)	16	R
164172	REH	FP3 base (high order)	16	R
164174	RFL	Link Pointer base (low order)	16	R
164176	RFH	Link Pointer base (high order)	16	R
		AP SPECIAL PURPOSE REGISTERS		
164200	ACL	Internal processor accumulator (low order)	16	R
164202	ACH	Internal processor accumulator (high order)	16	R
164204	MAR	RPU memory address	16	R
164206	SCCR	STARAN command control	16	R
164210	SRL	Internal processor shift (low order)	16	R
164212	SRH	Internal processor shift (high order)	16	R
164220	STSF	AP status flags	16	R
164222	RSM	Resume RPU	16	RI
164224	SMAJ	Stop every major program loop	16	RI
164226	SMIN	Stop every minor program loop	16	RI
164230	SCY	Stop after each clock cycle	16	RI
164232	CLDIA	Clear diagnostic	16	RI
164234	RPUST	RPU self test	16	RI

R A read only address. A load operation is ignored.

R/W A read or write address.

RI When the address is read it initiates a change in the state of the Sequential Processor/STARAN interface. The data returned is the STSF.

EXTERNAL FUNCTIONS

The external function (EXF) logic is used to control and sense various elements of the STARAN-E. Elements of STARAN-E issue 19-bit external function codes to EXF logic and receive 1-bit sense returns. Sequential control issues external functions using two device addresses, 164030 and 164032, that are given the mnemonics EFS and EFB respectively.

To issue an external function, sequential control should put bits 13 through 15 of the external function into bits 2 through 0, respectively, of EFS and then put bits 16 through 31 of the external function into bits 15 through 0, respectively, of EFB. Loading of EFS or EFB clears bits 15 and 7 of EFS to zeros. When the external function is treated by the EXF logic, one of these bits is set to one. Bit 7 is set to one if the sense bit returned is zero. Bit 15 is set to one if sense bit returned is one. Bits 7 and 15 of EFS cannot be set by the bus.

INTERRUPT ACCEPTANCE

Sequential control can accept interrupts from other STARAN-E elements. The interrupts arise from error detection, the panel-interrupt button, or external functions. Eight different interrupt vector addresses are provided. Table 1-4 shows the vector addresses and the priority levels. Bits 7, 6, and 5 in the processor status register (processor priority) of sequential control determine which interrupts are acceptable. When the processor priority is set at n, all request for interrupts at proirity level n and below are ignored.

Some sequential control peripherals may also generate interrupts to sequential control. Such interrupts do not use this interface.

PERIPHERALS

The STARAN-E Sequential Processor has as standard peripherals a keyboard/monitor, disk drive, card reader, line printer, and paper tape reader/punch. Communication between STARAN-E and the peripherals is handled through sequential control. The keyboard/monitor provides a communication link between STARAN-E and an operator.

OPTIONAL PERIPHERALS

Additional sequential control peripherals available are additional standard peripherals as well as magnetic tapes and communication to remote devices.

CHAPTER 2

STARAN-E INSTRUCTION SET

SECTION I. GENERAL

PROGRAM SEQUENCE

The AP control fetches instructions from AP control memory sequentially unless a branch, loop, load program status word, swap program status word, or interrupt is encountered.

PROGRAM COUNTER

Normally, the Program Counter contains the address of the current instruction. After an instruction is executed the Program Counter is incremented to the next sequential instruction. However, when AP control moves from the inactive to the active state the next instruction will be fetched from a dedicated location (0000) in AP control memory without disturbing the Program Counter. Also, when an interrupt is accepted, AP control will fetch an instruction from dedicated locations (0001 to 000F) without modifying the Program Counter. This allows the Program Counter state to be preserved so that an interrupted program can be continued after an interrupt is satisfied.

INSTRUCTION LENGTH

STARAN-E instructions are 32 bits in length. The bits of an instruction are numbered from 0 to 31 with bit 0 representing the most-significant bit (left) and bit 31 the least-significant bit (right).

INSTRUCTION TYPES

STARAN-E instructions are divided into 3 major types: AP Control instructions, Program Pager instructions, and External Function instructions. These three types are discussed in detail in this chapter.

SECTION II. STARAN AP CONTROL INSTRUCTIONS

GENERAL

Associative Processor (AP) Control instructions are divided into four classes: MDA Array instructions, Execution Control instructions, AP Control Register instructions, and General Register instructions.

SPEED-UP MODE

When executing AP instruction out of a page memory, a speed-up capability is provided for certain instruction sequences.

The speed-up mode permits fetching of future instructions while the current instruction is being processed. When the speed-up mode is implemented, the execution of an instruction is decreased approximately 50 nanoseconds.

SPEED-UP CODE

To implement speed-up mode, bit zero of the selected instruction is cleared to '0' and bit one is set to '1'.

RULE FOR SPEED-UP MODE

A general rule is that all instructions which do not reference control memory or affect the program counter may be executed in speed-up mode.

1) Speed-up mode is active only when executing out of page memory.

2) The following are instruction types which may have the speed-up code set:

   a) All MDA memory and MDA register instructions
   b) Load immediate instructions
   c) AP control register instructions which do not reference memory

3) The speed-up code must not be set in the following instructions or the instruction immediately preceeding one of the following instructions:

   a) External functions
   b) Branches

c) Load immediate to PSW
d) Swap PSW
e) Load AP control register from memory
f) Store AP control register to memory
g) Loop
h) General Register instructions

4) An additional requirement is that a speed-up code may be set in one of the valid instruction types (in 2 above) only if the instruction is followed by one of the valid types.

NOTE The load immediate instruction is not executed faster in speed-up mode. However, if its speed-up code is set, it permits faster execution of the instruction which follows.

MDA ARRAY INSTRUCTIONS

MDA Array instructions perform operations on the MDA arrays and MDA registers. Each basic array contains 2,359,296 bits, organized as a matrix of 256 words by 9216 bits. Access may be made in bit mode, word mode, or a mixed mode. Each MDA array memory module also contains three 256-bit MDA registers: X, Y, and M (Mask).

ARRAY SELECTION

The MDA instruction set operates on all array modules enabled by the Array Select (AS) register or on a single array module selected by Field Pointer 1 (FP1). The single array operations are referred to in this document as Link Pointer mode operations.

The AS register has one bit associated with each array module. If a bit is set to '1', the corresponding array module will participate in the multiple array operations. If the corresponding bit is zero, the corresponding array module will not participate.

Bit 0 of the AS register selects array module 0, bit 1 of AS selects array module 1, etc.

For single array operations FP1 will contain the array module number in the right-most 5 bits.

MDA FLIP NETWORK

The flip network in each MDA array module of STARAN scrambles and unscrambles data to and from the MDA memory. The flip network can permute data on transfers from memory to the MDA registers (X, Y and M), from the MDA registers to memory, and from MDA register to MDA register. The flip network is required to scramble the data when it is stored into memory and to unscramble the data when it is read from memory.

If the flip network is enabled (i.e., bit 6 of the MDA instruction is set to '1') the 256 bit data item is scrambled when stored to the MDA memory from X or Y and conversely unscrambled when loaded to X and Y from MDA memory. The scrambled bit pattern is the exclusive-OR ($\oplus$) of the true bit position number in X (or Y) and the flip address (address in bits 8-15 of the MDA instruction or in a field pointer if indirect addressing is used). Consider the following example:

This example illustrates the bit scrambling which occurs when storing the X register into word 10 of the MDA memory. The store instruction will have bit 6 (the flip bit) set to a '1'.

X Bit #	X Bit # ⊕ 10	Bit arrangement of word 10 after store operation
0	10	10
1	11	11
2	8	8
3	9	9
4	14	14
5	15	15
6	12	12
7	13	13
8	2	2
9	3	3
.	.	.
.	.	.
.	.	.
255	245	245

When MDA word 10 is loaded back to X (or Y) with the flip bit (bit 6 of the instruction) set to '1' (flip enabled), the reverse scrambling will take place (i.e., word 10 bit position numbers will be exclusive-ORed with '10', resulting in X returning to its original order).

Word '10' bit number	Original X
10⊕10	0
11⊕10	1
8⊕10	2
9⊕10	3
14⊕10	4
15⊕10	5
12⊕10	6
13⊕10	7
2⊕10	8
3⊕10	9
.	.
.	.
.	.
245⊕10	255

It is important to note that M (unlike X and Y) is kept in a flipped state. The HOME register contains the flip address (current flipped state) of M. For example, if the HOME register contains the value of

'10' (00001010 base 2), then the M register will be in the form shown above for the bits in word '10' (i.e., the M register will have its bits arranged as follows: 10, 11, 8, 9, 14, 15, 12, 13, 2, 3, . . . 245). When loading data to M from any source, the HOME register contents will be modified. Since M is flipped automatically when it is loaded, the flip bit (bit 6 of the instruction) should be cleared to '0' when loading M from flipped MDA memory. When storing M to MDA memory, the flip bit should be set to '1' and M will be flipped the same as the MDA memory destination address.

Since M is used as a mask to select elements to be stored in a 'store masked' operation, it is required that the bits of M be scrambled identically to the destination MDA address. For example, if one wishes to store X (or Y) masked to MDA array word 20, it is necessary that the M register bits be rearranged to correspond to the arrangement of the bits in word 20. To accomplish this, a load M with M (move mask) operation must be executed with the instruction flip bit set to '1' and the destination address as the flip address.

In the above example, the flip address of this move mask instruction will contain the address 20. After this instruction is executed, the HOME register will contain the value 20 and the M register will have the same bit arrangement as word 20. If one then needs to store the next value in word 21, it is necessary to do a move mask again with a '21' as the flip address of the instruction. Even though the HOME register is somewhat transparent to the programmer, one must be constantly aware of its value when any masked store operations are executed. To further clarify the interaction of the M register and the HOME register as they relate to various MDA operations, see the following table.

Operation (1)	Flip bit	HOME register before Operation	Value in bits 8-15 or field pointer	Effective value to flip network	HOME (2) register after operation
X->MDA memory	0	a	b	0	a
M->MDA memory	0	a	b	a	a
MDA memory->X	0	a	b	0	a
MDA memory->M	0	a	b	0	b
X->X	0	a	b	0	a
X->M	0	a	b	0	b
M->X	0	a	b	a	a
M->M	0	a	b	a	b
X->MDA memory	1	a	b	b	a
M->MDA memory	1	a	b	a⊕b	a
MDA memory->X	1	a	b	b	a
MDA memory->M	1	a	b	b	b
X->X	1	a	b	b	a
X->M	1	a	b	b	b
M->X	1	a	b	a⊕b	a
M->M	1	a	b	a⊕b	b

(1) Y may be substituted whenever X occurs in this table

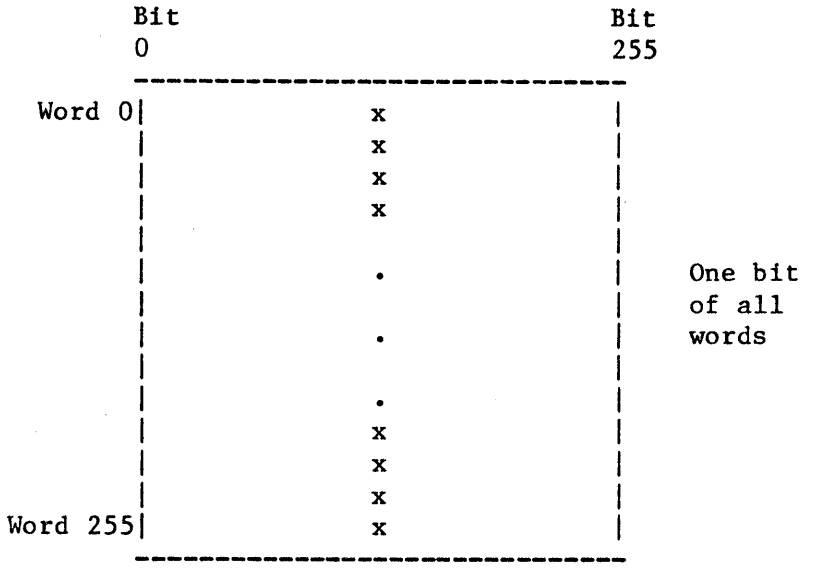
(2) Home register is altered whenever M is the destination

MIXED MODE ACCESS

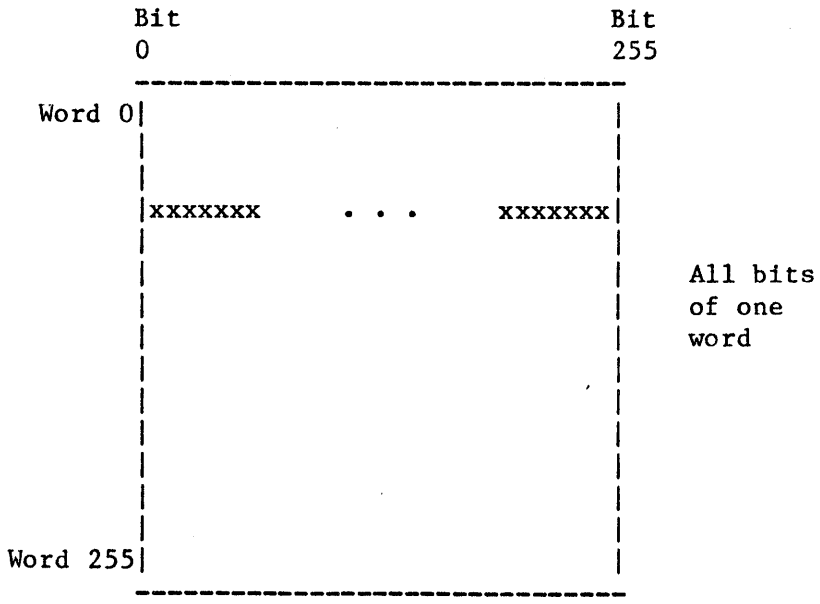
Mixed mode is simply a method of breaking up the 256-bit MDA memory slice into smaller segments and scattering these segments in various patterns throughout the MDA memory. The MDA access mode is an 8-bit quantity that determines the pattern (stencil) in the MDA memory. It is contained in the high order 8 bits of the base register. The MDA address (normally in bits 8-15 of the instruction or a field pointer) combined with the low order 16 bits of the base registers positions the pattern in the MDA memory.

A mode value of all zeros (00000000) indicates bit slice access and a mode of all ones (11111111) indicates word slice access. See the following figures.

MODE = '00000000'



MODE = '11111111'



Several other modes will be illustrated later in this discussion.

Some general rules for determining the access mode value are:

1) If the mode value contains n ones and 8-n zeros, then the stencil selects 2**n bits from each of 2**(8-n) memory words.

2) If the left-most n bits of the mode are ones and the right-most 8-n bits are zeros, then the 2**(8-n) memory words will be together in memory and the 2**n accessed bits of these words will be spaced apart. The left-most n bits of the address indicate bit addresses; the right-most 8-n bits of the address indicate word addresses.

3) If the right-most n bits of the mode are ones and the left-most 8-n bits are zeros, then 2**(8-n) memory words will be spread apart in memory and 2**n accessed bits of these words will be together. The left-most 8-n bits of the address indicate word addresses; the right-most 8-n bits of the address indicate bit addresses.

Example 1:

Assume the mode value is 00000111 and, for simplicity of the discussion, bits 8-15 of the instruction are 00000000.

Ignore the low-order 3 bits of the address, then bump through the high-order 5 bits. These 5 bits will be word addresses.

00000000	1st word address	0
00001000	2nd word address	8
00010000	3rd word address	16
00011000	4th word address	24
00100000	5th word address	32
	etc.	

The value of the low-order 3 bits of the address is the bit address which varies from 0 to 7 in each of the words noted above.

In this example, the first 8 bits are stored in word 0 starting at bit 0, the second 8 bits in word 8 starting at bit 0, etc. (because the address was zero). If the address had been 00000001 then the first 8 bits would be stored in word 1, the second 8 bits in word 9, etc.

Example 2:

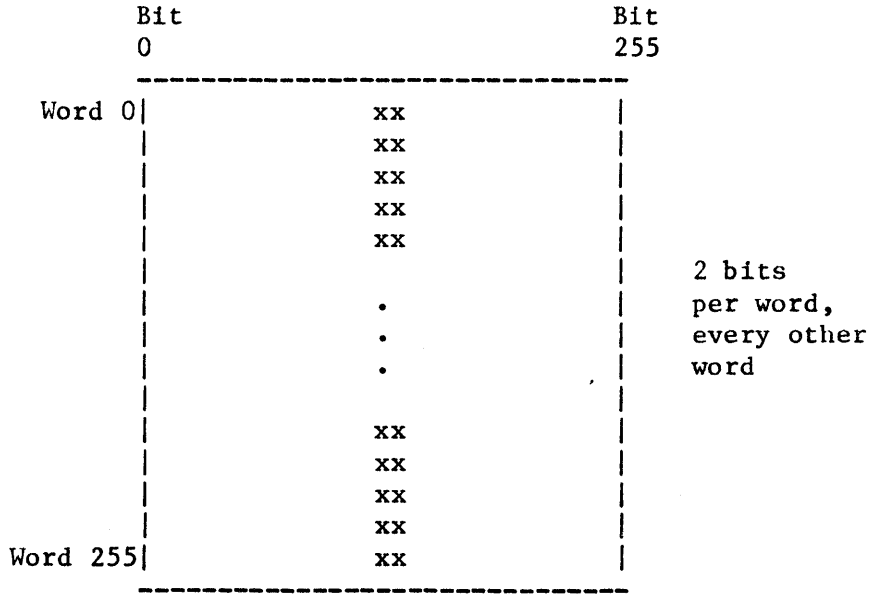
Assume the mode value is 11100000 and the address in bits 8-15 is 00000000.

Ignore the low-order 5 bits of the address, then bump through the high-order 3 bits. These 3 bits will be bit slice addresses.

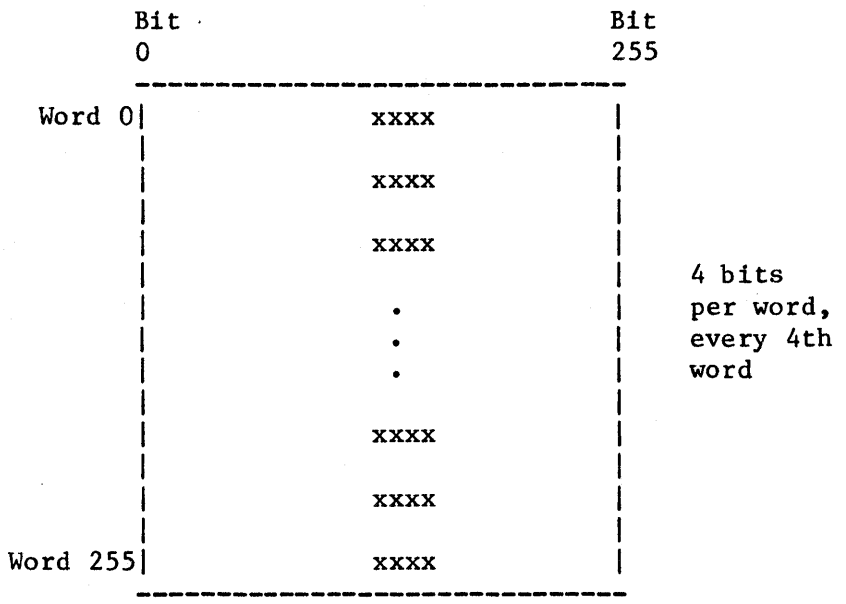
00000000	1st bit slice address	0
00100000	2nd bit slice address	32
01000000	3rd bit slice address	64
01100000	4th bit slice address	96
10000000	5th bit slice address	128
	etc.	

The value of the low-order 5 bits of the address is the word address and ranges from 0 to 31 (i.e., 32 words). Again, if the address was other than zero, the value of the address would be added to each bit slice address and would effectively shift the stencil. Some useful mixed access modes follow.

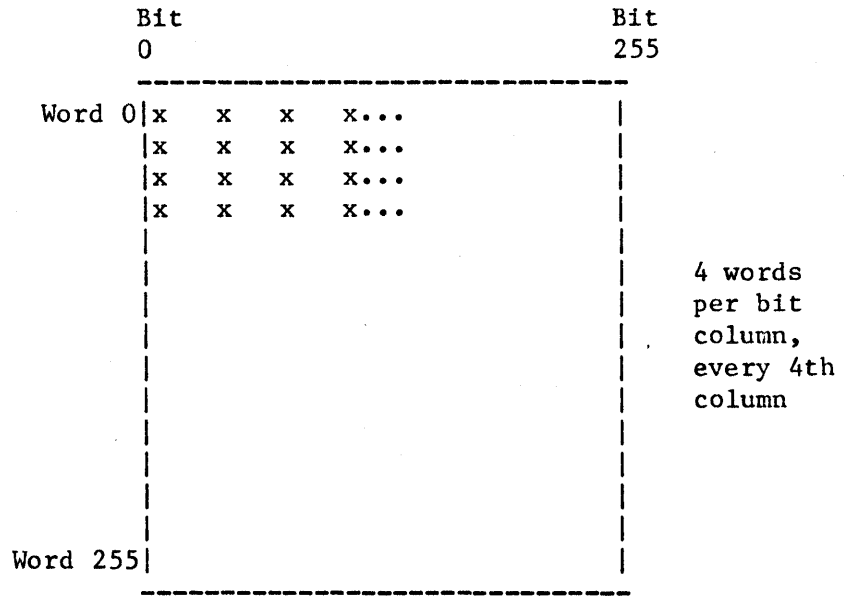
MODE = '00000001'

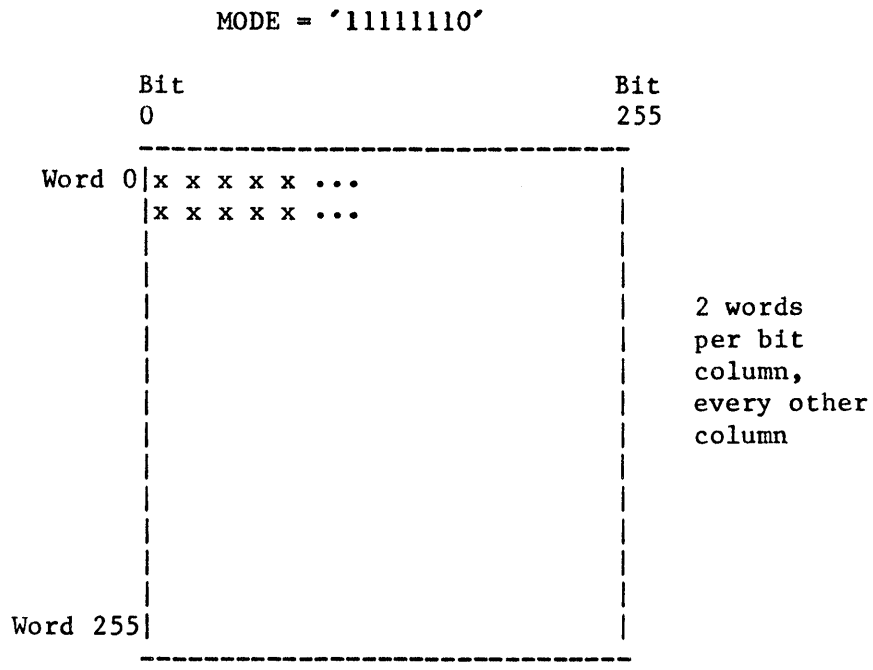


MODE = '00000011'



MODE = '11111100'





LOGIC FUNCTIONS

The logic functions performed on the X and Y registers are specified directly in the MDA instruction. Two logic functions can be specified in the instruction format. If a selected Common register bit (specified in FP1) is set to '1', the logic function in bits 16 to 19 is enabled; if the selected Common register bit is '0', the logic function in bits 20 to 23 is enabled. If the two logic function fields of the instruction are identical, the logic performed is independent of the contents of the Common register.

Table 2-1. Logic Table

Bits	24	25	26	24	25	26	24	25	26	24	25	26	24	25	26
	0	1	0	1	0	1	1	1	1	1	0	0	1	1	0
	XN	YN		XN	YN		XN	YN		XN		YN	XN		YN
0000	X	Y⊕F′		Y⊕F′	Y		X⊕F′	Y⊕F′		X⊕YF′		Y	X⊕YF′		Y⊕F′
0001	X	YvF′		XvF′	Y		XvF	YvF′		XvYF′		Y	XvYF′		YvF′
0010	X	YF		XF	Y		XF	YF		Y′XvXF		Y	Y′XvXF		YF
0011	X	Y		X	Y		X	Y		X		Y	X		Y
0100	X	F′		F′	Y		F′	F′		Y′XvYF′		Y	Y′XvYF′		F′
0101	X	Y′F′		X′F′	Y		X′F′	Y′F′		Y′XvYX′F′		Y	Y′XvYX′F′		Y′F′
0110	X	YF′		XF′	Y		XF′	YF′		Y′XvXF′		Y	Y′XvXF′		YF′
0111	X	0		0	Y		0	0		Y′X		Y	Y′X		0
1000	X	F		F	Y		F	F		Y′XvYF		Y	Y′XvYF		F
1001	X	YvF		XvF	Y		XvF	YvF		XvYF		Y	XvYF		YvF
1010	X	Y′vF		X′vF	Y		X′vF	Y′vF		Y′XvYFvYX′		Y	Y′XvYFvYX′		Y′vF
1011	X	1		1	Y		1	1		XvY		Y	XvY		1
1100	X	Y⊕F		X⊕F	Y		X⊕F	Y⊕F		X⊕YF		Y	X⊕YF		Y⊕F
1101	X	Y′F		X′F	Y		X′F	Y′F		Y′XvYX′F		Y	Y′XvYX′F		Y′F
1110	X	Y′vF′		X′vF′	Y		X′vF′	Y′vF′		Y′XvYX′vYF′		Y	Y′XvYX′vYF′		Y′vF′
1111	X	Y′		X′	Y		X′	Y′		Y⊕X		Y	Y⊕X		Y′

⊕=Exclusive OR
v=Inclusive OR
′=Complement

F=Bit from input (source determined by bits 29-31 of instruction format)
X=Old State of X Register
Y=Old State of Y Register

XN=New State of X
YN=New State of Y

SHIFTING

Shifting may also be specified in some MDA instructions directly in the instruction format. The shift constant is specified either in bits 8 through 15 of the instruction or in Field Pointer E (FPE). A shift amount of n bits with a modulus of m causes the 256-bit quantity to be divided into 256/m groups of m bits each. Within each an end-around right shift of n places occurs. Both n and m must be powers of 2. (Refer to the shift constant in Table 2-2.)

MIRRORING

Mirroring the 256-bit value is possible in some MDA instructions. The mirroring, if selected in the instruction, will cause the 256-bit input quantity to be flipped end-for-end (bit 'i' is put into bit '255-i') before shifting as specified by the shift constant.

LEFT SHIFT

A mirror, a shift of n places, and a mirror again results in a left shift of n places.

INPUT SOURCE

In each MDA array instruction, an input is selected from the following:

1) A 256-bit slice from the MDA array memory

2) The contents of the 256-bit MDA registers X, Y, or M

3) The contents of the 32-bit Common register, with zeros in the remaining 224 bits

4) The 32-bit mask generate value, with zeros in the remaining 224 bits

5) The resolver input, with one bit set to '1' and the remaining 255 bits equal to '0'

TABLE 2-2. Shift table

Hex Shift Constant	Modulus	Shift Amount	Shift Constant (Binary)
00	--	0	00000000
80	256	128	10000000
C0		64	11000000
E0		32	11100000
F0		16	11110000
F8		8	11111000
FC		4	11111100
FE		2	11111110
FF		1	11111111
40	128	64	01000000
60		32	01100000
70		16	01110000
78		8	01111000
7C		4	01111100
7E		2	01111110
7F		1	01111111
20	64	32	00100000
30		16	00110000
38		8	00111000
3C		4	00111100
3E		2	00111110
3F		1	00111111
10	32	16	00010000
18		8	00011000
1C		4	00011100
1E		2	00011110
1F		1	00011111

Table 2-2. (continued)

Hex Shift Constant	Modulus	Shift Amount	Shift Constant (Binary)
08	16	8	00001000
0C		4	00001100
0E		2	00001110
0F		1	00001111
04	8	4	00000100
06		2	00000110
07		1	00000111
02	4	2	00000010
03		1	00000011
01	2	1	00000001

Note: A shift amount of n with a modulus of m divides the 256-bit quantity into 256/m groups of m bits each. Within each group, an end-around right shift of n places occurs.

DESTINATION OR RESULT OF AN MDA INSTRUCTION

The selected input quantity is transmitted through the flip network which may shift, mirror, or perform other permutations. The 256-bit result is then transmitted to one of the following:

1) A 256-bit slice in the MDA array memory

2) A bit slice in the MDA array memory through a mask (In this case, only the bits with the corresponding mask bits set will receive data.)

3) The M register

4) The Y register combined logically with its previous content

5) The X register combined logically with its previous content

6) The X register combined with its previous content, masked by the Y register

7) The Common register which receives 32 bits and ignores the remaining 224 bits

8) A 256-bit output channel of the MIO

MDA ARRAY OPERATIONS

Load X and/or Y

The load X and/or Y operation permits loading the X and/or Y registers with a 256-bit quantity. The 256-bit value can come from any of the sources described later in the instruction format discussions. Logic may be performed simultaneously with the loading operations. The source data input can be combined logically (see Table 2-1) with the previous contents of the X and Y registers to produce new values in X and/or Y.

Load M (MASK)

The load M operation permits loading the M register with a 256-bit quantity. The 256-bit value can come from any of the sources described later in the instruction format discussions. Logic may be performed on the X and/or Y registers simultaneous with the load M (MASK) operation. Logic cannot be performed directly on the M register. When loading the M register, the HOME register is altered.

Store X, Y, or M to MDA array memory

This operation permits the storing of 256-bit quantities from the X, Y, or M registers to an MDA array bit column, word, or mixed slice.

Store X or Y to MDA array memory through a Mask

This operation permits the storing of the X or Y register through a mask in the M register. Bits will be stored in the destination MDA bit column, word or mixed slice only where the corresponding M register bit is set to '1'. In a store masked operation it is critical that the M register bits be arranged in the same order as the destination. The M register is ordered according to the current value in the HOME register. For example, if one wishes to store in word 55 of the array the M register must be flipped to the same order as word 55 before the

store masked operation is executed. This is accomplished by performing a load M with the current M register as the input and the address the same as the destination of the store operation. NOTE: The flip bit of the load M instruction should be set to '1'.

Load Common Register

The load Common register operation permits the loading of the 32-bit Common register from a selected 32-bit section of the 256-bit input of the associative instruction. The source can be any of the input sources specified in the instruction discussions that follow. When the Common register is loaded, a single array should be enabled. Note that the remaining 224 bits of the input are ignored.

Resolve Operation

The resolve operation is a special case of the associative operation set. The resolver logic makes available to the user the facility for finding the first Y register bit set in all selected arrays. This information combined with the logic capability in the associative instruction permits the user to then clear the first Y bit that is set and step the resolver to the next Y bit set. In this fashion one can step through and find all responses to a search.

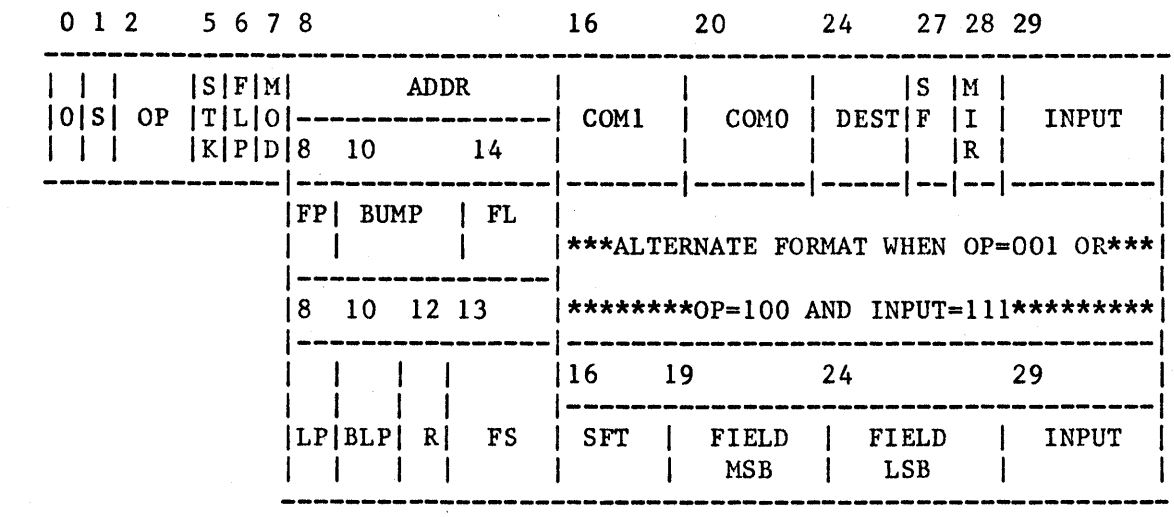
This operation typically is used after a parallel search algorithm has been performed. The search algorithms set all Y bits that satisfy the search criteria. The resolve operation permits the user to easily identify all words that satisfy the search.

The Link Pointer mode of the associative instruction is used for the resolve operation. When bit 12 of the instruction is set to '1', the resolve operation is selected. If the input field (bits 29, 30 and 31) is set to '000', the resolver output becomes the input to the associative instruction. The resolver output is a 256-bit quantity with one bit set to '1' and the remaining 255 bits cleared to zero. The bit that is set to '1' is the bit corresponding to the first Y register bit set.

When bits 10 and 11 of the resolve instruction are set to '11', the address of the first Y bit set is loaded into the Link Pointer. FP1 contains the array number and FP2 contains the Y bit number.

GENERAL MDA INSTRUCTION

Instruction Format



S

0 Normal operation
1 Speed up mode if executed in page memory

OP

000 Load X and/or Y
001 Load M (Mask)
010 Store to MDA memory through Mask
011 Store to MDA memory (256 bits)
100 Load Common register

STK

0 Selected base register associated with FP or direct address
1 Select R8 as base register (MDA stack pointer)

FLP

0	Unflipped data
1	Flipped data

MOD

0	Direct address in 8-15 of instruction
1	Indirect address in selected field pointer

ADDR — ADDR (bits 8-15 of instruction) contains the displacement which is added to the contents of selected base register to form effective MDA address (may also contain a shift constant).

COM1 — COM1 contains the logic function performed if a selected Common register bit (FP1 selects bit)=1. NOTE: If COM1=COM0, the Common register is not involved in logic selection (see Table 2-1).

COM0 — COM0 contains the logic function performed if a selected Common register bit=0 (see Table 2-1).

DEST

000	X and Y unchanged
010	Y combined logically with input
101	X combined logically with input
111	X and Y combined logically with input
100	X combined logically with input only where Y=1
110	X combined logically with input only where Y=1; Y combined logically with input

SF

0	No shift
1	Shift enabled (see Table 2-2)

MIR

0	No mirror
1	Mirroring enabled

INPUT

000	Common register bits 0-31; bits 32-255 are zero
001	M register
010	Y register
011	X register
100	Reserved for CIOU
101	MDA memory slice
110	X or Y (X if all Y bits=0; Y if any Y bit=1)
111	Mask generator input in bits 0-31; bits 32-255 are zero

FP

00	FP1 contains MDA address displacement
01	FP2 contains MDA address displacement
10	FP3 contains MDA address displacement

BUMP

0000	Field pointers not altered
0001	Field pointers not altered
0010	FPE post-incremented
0011	FPE post-decremented
0100	FP1 post-incremented
0101	FP1 post-decremented
0110	FP1 and FPE post-incremented
0111	FP1 and FPE post-decremented
1000	Selected pointer post-incremented
1001	Selected pointer post-decremented
1010	Sellcted pointer and FPE post-incremented
1011	Selected pointer and FPE post-decremented
1100	Selected pointer and FP1 post-incremented
1101	Selected pointer and FP1 post-decremented
1110	Selected pointer, FP1 and FPE post-incremented
1111	Selected pointer, FP1 and FPE post-decremented

FL

00	Length counters not altered
01	FL1 post-decremented
10	FL2 post-decremented
11	BL post-decremented

LP

11	FP1 contains array number, FP2 contains MDA address displacement, RF is the associated base register

BLP

00	No change in Link Pointer (LP)
01	LP post-decremented
10	LP post-incremented
11	Load LP with resolve address (current position of first Y bit set)

R

0	Nop
1	Step resolver to next Y bit set (input from resolver if INPUT='000')

FS

000	Field 0 (Bits 0-31)
001	Field 1 (Bits 32-63)
010	Field 2 (Bits 64-95)
011	Field 3 (Bits 96-127)
100	Field 4 (Bits 128-159)
101	Field 5 (Bits 160-191)
110	Field 6 (Bits 192-223)
111	Field 7 (Bits 224-255)

FIELD MSB

00000	Bit 0
•	•
•	•
•	•
11111	Bit 255

FIELD LSB

00000	Bit 0
•	•
•	•
•	•
11111	Bit 255

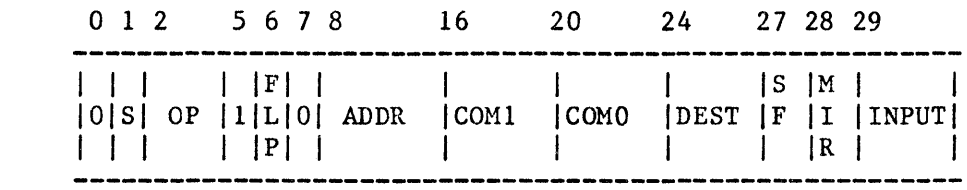
SFT

000	Shift right 1 bit
001	Shift right 2 bits
010	Shift right 4 bits
011	Shift right 8 bits
100	Shift right 16 bits
101	No shift
110	No shift
111	No shift

MDA INSTRUCTION FORMAT (DIRECT ADDRESS MODE)

This instruction performs operations in the MDA portion of the STARAN-E. The direct address mode will perform the operation in all MDA modules (arrays) selected in the Array Select register.

Instruction Format



S	Function
0	Normal execution
1	Speed-up mode if instruction is executed in page memory

OP	Function
000	Load X and/or Y
001	Load M (see note 1)
010	Store to array memory through Mask
011	Store to array memory (256 bits)
100	Load Common register (see note 1)

Bit 5 = 1 Indicates the base register R8 is used to determine the array address. The contents of R8 is added to ADDR (value in bits 8-15); R8 also contains the access mode (i.e., word, bit or mixed) in bits 0-7.

FLP	Function
0	Unflipped
1	Flipped data

Bit 7 = 0 Indicates direct addressing mode. The displacement is contained in bits 8-15.

ADDR Bits 8-15 contain the displacement which is added to the contents of R8 to determine the effective array address. Bits 8-15 may also contain a shift constant if array memory is not the input value.

COM1	The logic function performed if the Common register bit selected in FP1 is equal to '1'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see note 2).
COM0	The logic function performed if the Common register bit selected in FP1 is equal to '0'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see note 2).

DEST	Result
000	X and Y unchanged
010	Y combined logically with INPUT
101	X combined logically with INPUT
111	X and Y combined logically with INPUT
100	X combined logically with INPUT only where Y bit = 1
110	X combined logically with INPUT only where Y bit = 1 and Y combined logically with INPUT

SF	Action
0	No shifting
1	Shifting enabled (see note 3)

MIR	Action
0	No mirroring
1	Mirroring enabled (see note 4)

INPUT	Source
000	Common register in bits 0-31; bits 32-255 are zero (see note 5)
001	M register
010	Y register
011	X register
100	Reserved for CIOU
101	Array memory (256 bits). NOTE: The array access mode is contained in a base register.
110	X or Y (if all Y bits = 0, X is input; if any Y bit = 1, Y is input)
111	Mask generator input in bits 0-31; bits 32-255 are zero (see note 5)

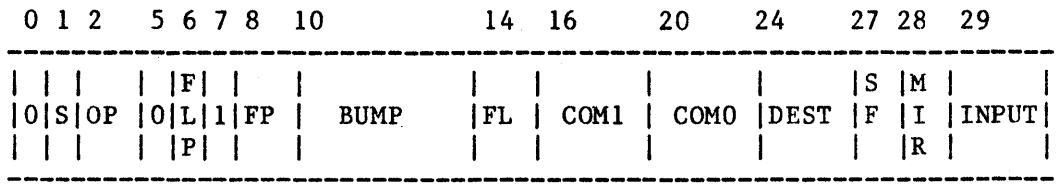
NOTES****

1. Alternate instruction format for bits 16 through 28 (covered later in this section).
2. See logic function table, Table 2-1.
3. See shift table, Table 2-2.
4. Flipping and mirroring occur prior to shifting.
5. With the shifting capability the 32 bits can be placed into any of the eight 32-bit sections of the 256-bit input value.

MDA INSTRUCTION FORMAT (INDIRECT ADDRESS MODE)

This instruction performs operations in the MDA portion of the STARAN-E. The indirect address mode will perform the operation in the one array selected in FP1.

Instruction Format



S	Function
0	Normal execution
1	Speed-up mode if instruction is executed in page memory

OP	Function
000	Load X and/or Y
001	Load M (Mask)
010	Store to array memory through Mask
011	Store to array memory (256 bits)
100	Load Common register

Bit 5 = 0 Indicates base register RB through RF will be selected; the base register selected depends on the selected field pointer (see Table 1-2).

FLP	Function
0	Unflipped
1	Flipped

Bit 7 = 1 Indicates indirect addressing mode. The address displacement is contained in one of the field pointers.

FP	Selected Pointer
00	FP1
01	FP2
10	FP3

BUMP	Post-operation
0000	Field pointers not altered
0001	Field pointers not altered
0010	FPE is post-incremented
0011	FPE is post-decremented
0100	FP1 is post-incremented
0101	FP1 is post-decremented
0110	FPE and FP1 are post-incremented
0111	FPE and FP1 are post-decremented
1000	Selected pointer (in FP) is post-incremented
1001	Selected pointer (in FP) is post-decremented
1010	Selected pointer (in FP) and FPE are post-incremented
1011	Selected pointer (in FP) and FPE are post-decremented
1100	Selected pointer (in FP) and FP1 are post-incremented
1101	Selected pointer (in FP) and FP1 are post-decremented
1110	Selected pointer (in FP), FPE and FP1 are post-incremented
1111	Selected pointer (in FP), FPE and FP1 are post-decremented

FL	Post Operation on Length Counter
00	Length counters not altered
01	FL1 post-decremented
10	FL2 post-decremented
11	BL post-decremented

COM1 — The logic function performed if the Common register bit selected in FP1 is equal to '1'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see Table 2-1).

COM0 — The logic function performed if the Common register bit selected in FP1 is equal to '0'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see Table 2-1).

DEST	Result
000	X and Y unchanged
010	Y combined logically with INPUT
101	X combined logically with INPUT
111	X and Y combined logically with INPUT
100	X combined logically with INPUT only where Y bit = 1
110	X combined logically with INPUT only where Y bit = 1 and Y combined logically with INPUT.

SF	Action
0	No shifting
1	Shifting enable (see Table 2-2)

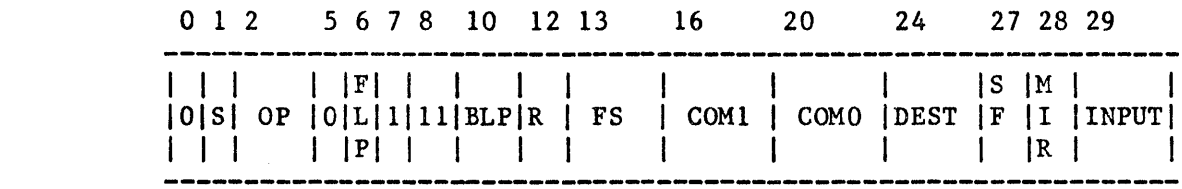
MIR	Action
0	No mirroring
1	Mirroring enabled

INPUT	Source
000	Common register in bits 0-31; bits 32-255 are zero
001	M (Mask) register
010	Y register
011	X register
100	Reserved for CIOU
101	Array memory (256 bits). NOTE: The array access mode is contained in a base register
110	X or Y (if all Y bits = 0, X is input; if any Y bit = 1, Y is input)
111	Mask generator input in bits 0-31; bits 32-255 are zero

MDA INSTRUCTION FORMAT (LINK POINTER MODE)

This instruction operates in the MDA portion of the STARAN-E. It performs the operation in <u>one</u> array only. The array is selected by FP1.

Instruction Format



S	Function
0	Normal execution
1	Speed-up mode if instruction is executed in page memory

OP	Function
000	Load X and/or Y
001	Load M (Mask)
010	Store to array memory through Mask
011	Store to array memory (256 bits)
100	Load Common register

<u>Bit 5 = 0</u> Indicates the base register RF is used to determine the array address. The contents of RF is added to FP2; RF also contains the access mode (i.e., word, bit or mixed)

FLP	Function
0	Unflipped
1	Flipped data

<u>Bit 7 = 1</u> Indirect addressing mode

<u>Bits 8,9 = 11</u> The Link Pointer (FP1 and FP2 concatenated) selects the array and array address. FP1 contains the array number. FP2 contains the array address displacement which is added to base register RF to form the effective address.

BLP	Post Operation
00	Link Pointer is not altered
01	Link Pointer is post-decremented
10	Link Pointer is post-incremented
11	Link Pointer is loaded with the resolver address

R	Operation Performed
0	No operation
1	Resolver stepped to next Y-bit set

FS — Selects field in the 256-bit value where a 32-bit input resides; used when INPUT is the Common register or the mask generator.

000	Field 0 (bits 0-31)
001	Field 1 (bits 32-63)
010	Field 2 (bits 64-95)
011	Field 3 (bits 96-127)
100	Field 4 (bits 128-159)
101	Field 5 (bits 160-191)
110	Field 6 (bits 192-223)
111	Field 7 (bits 224-255)

COM1 — The logic function performed if the Common register bit selected in FP1 is equal to '1'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see Table 2-1).

COM0 — The logic function performed in the Common register bit selected in FP1 is equal to '0'. NOTE: If COM1 is equal to COM0, the Common register is not involved in the logic function selection (see Table 2-1).

DEST	Result
000	X and Y unchanged
010	Y combined logically with INPUT
101	X combined logically with INPUT
111	X and Y combined logically with INPUT
100	X combined logically with INPUT only where Y bit = 1
110	X combined logically with INPUT only where Y bit = 1 and Y combined logically with INPUT

SF	Action
0	No shifting
1	Shifting enabled (see Table 2-2)

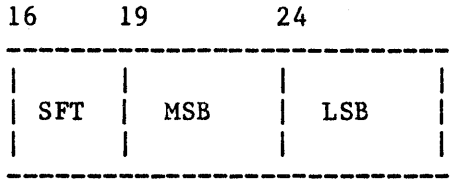
MIR	Action
0	No mirroring
1	Mirroring enabled

INPUT	Source
000	Common register in bits 0-31; bits 32-255 are zero
001	M register
010	Y register
011	X register
100	Reserved for CIOU
101	Array memory (256 bits). NOTE: The array access mode is contained in a base register.
110	X or Y (if all Y bits = 0, X is input; if any Y bit = 1, Y is input)
111	Mast generator input in bits 0-31; bits 32-255 are zero

ALTERNATE MDA INSTRUCTION FORMAT

An alternate format for MDA instruction bits 16 through 28 is used when OP is '100' (Load Common register) or when OP is '001' (Load M) and the input is '111' (mask generator).

Instruction Format



SFT — SFT is the power of 2 shift value that is applied to the 32-bit input quantity.

000	Shift 32-bit input right end around 2**0 (1) bits
001	Shift 32-bit input right end around 2**1 (2) bits
010	Shift 32-bit input right end around 2**2-(4) bits
011	Shift 32-bit input right end around 2**3 (8) bits
100	Shift 32-bit input right end around 2**4 (16) bits
101	No shift

MSB — MSB is the most significant (left-most) bit of the field selected within the 32-bit input.

00000	Bit 0
00001	Bit 1
.	.
.	.
.	.
11111	Bit 31

LSB — LSB is the least significant bit of the field selected within the 32-bit input.

00000	Bit 0
00001	Bit 1
.	.
.	.
.	.
11111	Bit 31

EXECUTION CONTROL INSTRUCTIONS

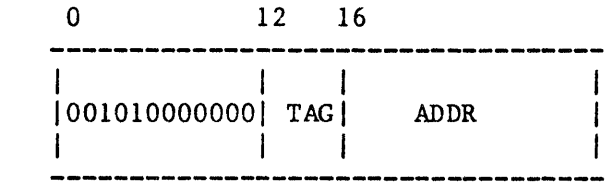
BRANCH INSTRUCTION

The normal sequence of instruction execution can be modified by a branch operation, a branch and link to a subroutine, a loop instruction, or a call subroutine instruction.

UNCONDITIONAL BRANCH INSTRUCTION

The unconditional branch instruction causes the effective address to be put into the program counter. The next operation is executed from the effective address.

Instruction Format

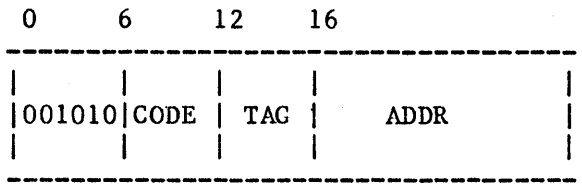


TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

CONDITIONAL BRANCH INSTRUCTION

If the condition specified by CODE is true, the Conditional Branch operation acts like an unconditional branch. If the conditon is not true, no change in control occurs and the next sequential instruction is executed. The effective branch address is a function of the address and tag fields of the instruction. This permits address modification by the DP register and the Branch and Link registers. If the tag field specifies modification of the DP or BL, these changes will occur regardless of the condition.

Instruction Format



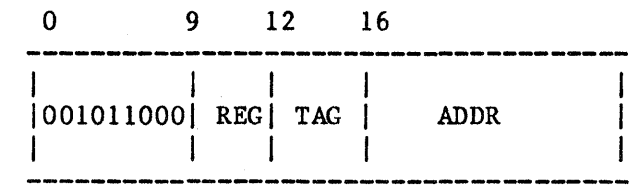
CODE	Branch If
000000	Unconditional branch
000001	No branch
000010	FP1 = 0
000011	FP1 ≠ 0
000100	FP2 = 0
000101	FP2 ≠ 0
000110	FP3 = 0
000111	FP3 ≠ 0
001000	FL1 = 0
001001	FL1 ≠ 0
001010	FL2 = 0
001011	FL2 ≠ 0
001100	Selected bit of Common = 0
001101	Selected bit of Common = 1
001110	All Y register bits in enabled arrays = 0
001111	Any Y register bit in any enabled array = 1
010000	BL = 0
010001	BL ≠ 0
010010	FPE = 0
010011	FPE ≠ 0
010100	DP = 0
010101	DP ≠ 0

TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

BRANCH AND LINK INSTRUCTION

Branch and link instructions cause the current contents of the Program Counter to be stored in the right half of a branch and link register, with the zeros stored in the left half. (The Program Counter will now contain the address of the next sequential instruction after the branch and link instruction.) Then the effective address in the branch and link instruction is loaded into the Program Counter

Instruction Format



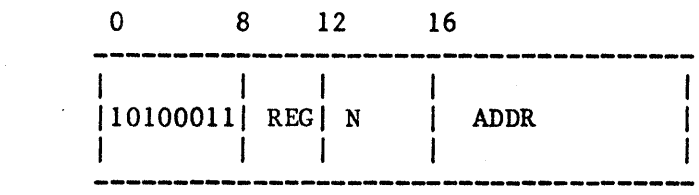
TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

REG	Register
000	R0
001	R1
010	R2
011	R3
100	R4
101	R5
110	R6
111	R7

CALL SUBROUTINE INSTRUCTION

The call subroutine instruction performs two major operations. It loads 'N' general registers with data from the parameter list and then performs a branch and link to a subroutine. The parameter list begins with the first memory location following the CAL instruction and has a length of 'N' words. If 'N' is zero, there is no parameter list and no registers are loaded. The words of the parameter list are loaded into registers numbered R through R+N-1 starting with register R. The numbering of the registers wraps around from 15 to 0. After the 'N' parameters have been loaded into the registers, the address of the word following the last word of the parameter list is stored into general register 7 (R7) and the Program Counter is loaded with the value of the address field, effecting a branch to the subroutine entry at the address.

Instruction Format



REG — The REG field selects the first general register to receive the first word in the parameter list.

0000	R0
0001	R1
0010	R2
0011	R3
0100	R4
0101	R5
0110	R6
0111	R7
1000	R8
1001	R9
1010	RA
1011	RB
1100	RC
1101	RD
1110	RE
1111	RF

N — The N field contains the number of parameters to be loaded from the words following the CAL instruction.

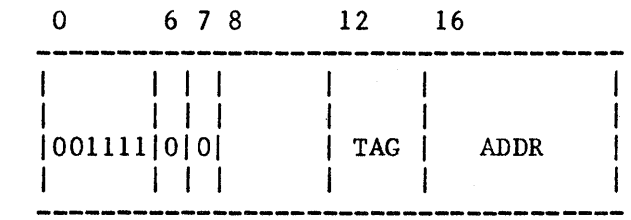
ADDR — The ADDR field contains a 16-bit program memory address (bulk memory, pages or HSDB). This address points to the first instruction of the subroutine.

Note: The return address (next instruction after the parameter list) is stored in R7.

LOOP INSTRUCTION

The loop instruction permits the repetitive execution of one or more sequential instructions. Before the loop operation is performed, FL1 should contain the number of repetitions minus 1 (i.e., if 10 repetitions are required, FL1 should contain a 9).

Instruction Format



TAG	Effective end-of-loop address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

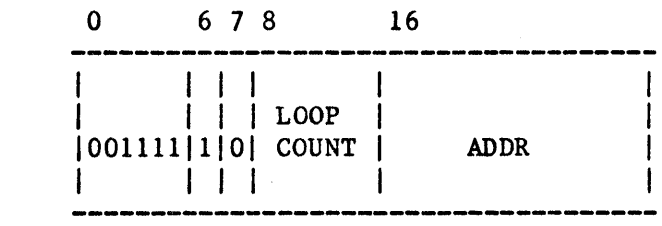
Restrictions

Instructions that alter the Program Counter, i.e., branches, external functions, etc., should not be used as the last instruction of a loop sequence. No instruction within a loop should alter FL1. FL1 is equal to zero on a normal exit from a loop.

LOAD AND LOOP INSTRUCTION

The load and loop operation is identical to the loop operation, except FL1 need not be preloaded with the repetition count. This operation permits loading of FL1 with the count in the same operation. In both the loop and the load and loop instructions, the end of loop address is the effective address.

Instruction Format



LOOP COUNT — Number of repetitions minus 1

ADDR — End-of-loop address

Restrictions

This instruction has the same restrictions as the Loop instruction.

AP CONTROL REGISTER INSTRUCTIONS

Register instructions allow AP control registers to be loaded with an immediate value, with the content of an AP control memory address, or from another AP control register. Also, the AP control registers may be stored into AP control memory. Figure 2-1 provides a map of the AP control registers on the 32-bit bus.

AP CONTROL REGISTER LOAD OPERATIONS

Three load operations are possible:

1) Load low half will load the right, or least-significant half, of the register group.

2) Load high half will load the left, or most-significant half, of the register group.

3) Load high and low half will load the entire register group specified.

Figure 2-1. AP Control Register Groups

GROUP	BITS 0-7	BITS 8-15	BITS 16-23	BITS 24-31
000	C0	C1	C2	C3
001	AS0	AS1	AS2	AS3
010	BL0	BL1	DP0	DP1
011	PC0	PC1		CC-IMASK
100	FL2	FPE		
101	FL1			FP2
110		FP3	FP1	
111	FL1	FP3	FP1	FP2

Table 2-3. Associative Processor Control Register Abbreviations

ABBR	REGISTER	NO. OF BITS
C	Common	32
CH	Common (bits 0-15)	16
CL	Common (bits 16-31)	16
C0	Common (byte 0)	8
C1	Common (byte 1)	8
C2	Common (byte 2)	8
C3	Common (byte 3)	8
AS	Array Select	32
ASH	Array Select (bits 0-15)	16
ASL	Array Select (bits 16-31)	16
AS0	Array Select (byte 0)	8
AS1	Array Select (byte 1)	8
AS2	Array Select (byte 2)	8
AS3	Array Select (byte 3)	8
BL	Block Length Counter	16
BL0	Block Length Counter (byte 0)	8
BL1	Block Length Counter (byte 1)	8
DP	Data Pointer	16
DP0	Data Pointer (byte 0)	8
DP1	Data Pointer (byte 1)	8
FL1	Field Length Counter 1	8
FL2	Field Length Counter 2	8
FP1	Field Pointer 1	8
FP2	Field Pointer 2	8
FP3	Field Pointer 3	8
FPE	Field Pointer E	8
PSW	Program Status Word	32
PC	Program Counter (bits 0-15 of PSW)	16
CC	Condition Code (bits 24-27 of PSW)	4
IMASK	Interrupt Mask (bits 28-31 of PSW)	4

LEFT SHIFT

The source data is shifted left-end-around 0, 1, 2, or 3 bytes as specified in the shift field. The direction of the shift is from bit 31 toward bit 0.

EFFECTIVE ADDRESS FORMATION

Register operations that referance AP control memory for loading from memory or storing to memory allow effective address to be a function of:

1) The address field of the instruction (bits 16-31).

2) The Data Pointer (DP) or general registers R0 to R7.

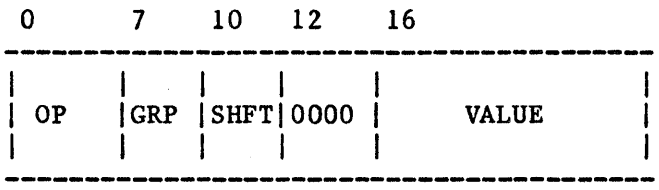
The tag field of the instruction determines the usage of the DP or general registers.

Decrementing and incrementing of DP and BL are also permitted as specified in the tag field. This permits stepping through a block of data in AP control memory.

LOAD IMMEDIATE

The Load Immediate instruction loads either 4, 8, or 16 bits of the value field (bits 16-31) of the instruction into a register group. The register receiving the immediate data is selected in the group (GRP) field (bits 7,8, and 9). The 16-bit value is extended to a 32-bit value by appending 16 zeros. The 32-bit quantity is then shifted 0, 1, 2, or 3 bytes (left shift end around) as designated by the shift (SHFT) field. Parts of the shifted quantity are then loaded into the register group specified in the operation (OP) field.

Instruction Format



OP	Action
0011001	Load low half of group
0011010	Load high half of group
0011011	Load high and low halves of group

GRP	Register Group
000	Common
001	AS
010	BL and DP
011	PSW
100	FL2 and FPE
101	FP2 and FL1
110	FP3 and FP1
111	FL1, FP3, FP1, and FP2

SHFT	Shift Amount
00	No shift
01	Left shift 8 bits
10	Left shift 16 bits
11	Left shift 24 bits

VALUE — Contains 16-bit immediate value to be loaded

Table 2-4 is described as follows:

1) The 4 digit hex code on the left indicates the high order 16 bits of the Load Immediate instruction.

2) The low-order 16 bits of the instruction are stored into the registers shown in bits 16 to 31 of the table.

3) The registers shown in bits 0-15 of the table are cleared.

Table 2-4. Load Immediate Instruction

	CLEARED		IMMEDIATE VALUE	
OPCODE	0 7	8 15	16 23	24 31
3200			C2	C3
3210	C3			C2
3230		C2	C3	
3240			AS2	AS3
3250	AS3			AS2
3270		AS2	AS3	
3280			DP0	DP1
3290	DP1			DP0
32B0		DP0	DP1	
32C0				CC-IMASK
32F0			CC-IMASK	
3320			FL2	
3330				FL2

Table 2-4. (continued)

	CLEARED		IMMEDIATE VALUE	
OPCODE	0 7	8 15	16 23	24 31
3340				FP2
3370			FP2	
3380			FP1	
3390				FP1
33C0			FP1	FP2
33D0	FP2			FP1
33F0		FP1	FP2	
3410		C0	C1	
3420			C0	C1
3430	C1			C0
3450		AS0	AS1	
3460			AS0	AS1
3470	AS1			AS0
3490		BL0	BL1	
34A0			BL0	BL1
34B0	BL1			BL0
34D0		PC0	PC1	
34E0			PC0	PC1
34F0	PC1			PC0

Table 2-4. (continued)

	CLEARED		IMMEDIATE VALUE	
OPCODE	0 7	8 15	16 23	24 31
3510			FPE	
3520				FPE
3560			FL1	
3570				FL1
3590			FP3	
35A0				FP3
35D0		FL1	FP3	
35E0			FL1	FP3
35F0	FP3			FL1
3600	C0	C1	C2	C3
3610	C3	C0	C1	C2
3620	C2	C3	C0	C1
3630	C1	C2	C3	C0
3640	AS0	AS1	AS2	AS3
3650	AS3	AS0	AS1	AS2
3660	AS2	AS3	AS0	AS1
3670	AS1	AS2	AS3	AS0
3680	BL0	BL1	DP0	DP1
3690	DP1	BL0	BL1	DP0

Table 2-4. (continued)

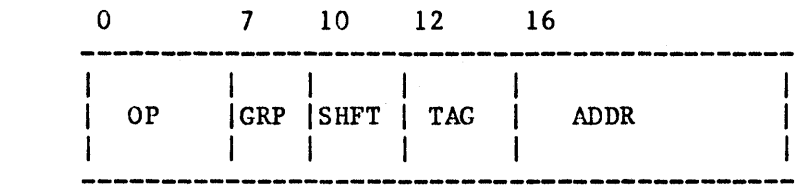
	CLEARED		IMMEDIATE VALUE	
OPCODE	0 7	8 15	16 23	24 31
36A0	DP0	DP1	BL0	BL1
36B0	BL1	DP0	DP1	BL0
36C0	PC0	PC1		CC-IMASK
36D0	CC-IMASK	PC0	PC1	
36E0		CC-IMASK	PC0	PC1
36F0	PC1		CC-IMASK	PC0
3710		FL2	FPE	
3720			FL2	FPE
3730	FPE			FL2
3740	FL1			FP2
3760		FP2	FL1	
3770			FP2	FL1
3780		FP3	FP1	
3790			FP3	FP1
37A0	FP1			FP3
37C0	FL1	FP3	FP1	FP2
37D0	FP2	FL1	FP3	FP1
37E0	FP1	FP2	FL1	FP3
37F0	FP3	FP1	FP2	FL1

LOAD AP CONTROL REGISTER FROM CONTROL MEMORY

The load register instruction loads a byte, a half word, or a word of data from main memory or HSDB into a register group.

The effective address of the source is formed using the TAG and ADDRESS fields of the instruction. A hangup will result if the effective address points to non-existant memory or to a word in page memory. The register or registers loaded are selected by the GRP and SHFT fields of the instruction.

Instruction Format



OP	Action
0011001	Load low half of group
0011010	Load high half of group
0011011	Load high and low halves of group

GRP	Register Group
000	Common
001	AS
010	BL and DP
011	PSW
100	FL2 and FPE
101	FP2 and FL1
110	FP3 and FP1
111	FL1, FP3, FP1, and FP2

SHFT	Shift Amount
00	No shift
01	Left shift 8 bits
10	Left shift 16 bits
11	Left shift 24 bits

TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

Table 2-5 is described as follows:

1) The 3 digit hex code on the left indicates the high order 12 bits of the Load AP Register instruction.

2) The bit indicators at the top of the table indicates the bit positions of the source memory word.

3) The registers indicated in the blocks show the destination registers receiving each of the 8-bit quantities from memory.

Table 2-5. Load Register from Control Memory

OPCODE	32-BIT VALUE 0 7	8 15	16 23	24 31
320			C2	C3
321	C3			C2
322	C2	C3		
323		C2	C3	
324			AS2	AS3
325	AS3			AS2
326	AS2	AS3		
327		AS2	AS3	
328			DP0	DP1
329	DP1			DP0
32A	DP0	DP1		
32B		DP0	DP1	
32C				CC-IMASK
32D	CC-IMASK			
32E		CC-IMASK		
32F			CC-IMASK	
330	FL2			
331		FL2		

Table 2-5. (continued)

OPCODE	32-BIT VALUE			
	0 7	8 15	16 23	24 31
332			FL2	
333				FL2
334				FP2
335	FP2			
336		FP2		
337			FP2	
338			FP1	
339				FP1
33A	FP1			
33B		FP1		
33C			FP1	FP2
33D	FP2			FP1
33E	FP1	FP2		
33F		FP1	FP2	
340	C0	C1		
341		C0	C1'	
342			C0	C1
343	C1			C0

Table 2-5. (continued)

OPCODE	32-BIT VALUE			
	0 7	8 15	16 23	24 31
344	AS0	AS1		
345		AS0	AS1	
346			AS0	AS1
347	AS1			AS0
348	BL0	BL1		
349		BL0	BL1	
34A			BL0	BL1
34B	BL1			BL0
34C	PC0	PC1		
34D		PC0	PC1	
34E			PC0	PL1
34F	PC1			PC0
350		FPE		
351			FPE	
352				FPE
353	FPE			
354	FL1			
355		FL1		
356			FL1	

Table 2-5. (continued)

OPCODE	32-BIT VALUE 0 7	8 15	16 23	24 31
357				FL1
358		FP3		
359			FP3	
35A				FP3
35B	FP3			
35C	FL1	FP3		
35D		FL1	FP3	
35E			FL1	FP3
35F	FP3			FL1
360	C0	C1	C2	C3
361	C3	C0	C1	C2
362	C2	C3	C0	C1
363	C1	C2	C3	C0
364	AS0	AS1	AS2	AS3
365	AS3	AS0	AS1	AS2
366	AS2	AS3	AS0	AS1
367	AS1	AS2	AS3	AS0

Table 2-5. (continued)

OPCODE	32-BIT VALUE 0 7	8 15	16 23	24 31
368	BL0	BL1	DP0	DP1
369	DP1	BL0	BL1	DP0
36A	DP0	DP1	BL0	BL1
36B	BL1	DP0	DP1	BL0
36C	PC0	PC1		CC-IMASK
36D	CC-IMASK	PC0	PC1	
36E		CC-IMASK	PC0	PC1
36F	PC1		CC-IMASK	PC0
370	FL2	FPE		
371		FL2	FPE	
372			FL2	FPE
373	FPE			FL2
374	FL1			FP2
375	FP2	FL1		
376		FP2	FL1	
377			FP2	FL1
378		FP3	FP1	
379			FP3	FL1

Table 2-5. (continued)

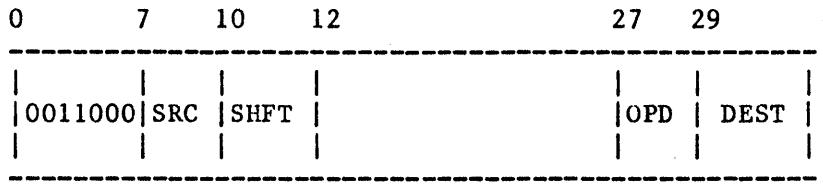
32-BIT VALUE

OPCODE	0 7	8 15	16 23	24 31
37A	FP1			FP3
37B	FP3	FP1		
37C	FL1	FP3	FP1	FP2
37D	FP2	FL1	FP3	FP1
37E	FP1	FP2	FL1	FP3
37F	FP3	FP1	FP2	FL1

LOAD AP CONTROL REGISTER FROM AP CONTROL REGISTER

The load register from register instruction allows the transfer of data between AP control registers. Registers may be transferred individually or in various group combinations.

Instruction Format



SRC	Source Register Group
000	Common
001	AS
010	BL, DP
011	PSW
100	FL2, FPE
101	FL1, FP3, FP1, FP2
110	FL1, FP3, FP1, FP2
111	FL1, FP3, FP1, FP2

SHFT	Shift Amount
00	No shift
01	Left shift 8 bits
10	Left shift 16 bits
11	Left shift 24 bits

OPD	Store Operation
01	Store low half of group
10	Store high half of group
11	Store high and low halves of group

DEST	Destination Register Group
000	Common
001	AS
010	BL, DP
011	PSW
100	FL2, FPE
101	FP2, FL1
110	FP3, FP1
111	FL1, FP3, FP1, FP2

Table 2-6 is described as follows:

1) The 3 digit hex code on the left indicates the high order 12 bits of the Load AP Register from Register instruction.

2) The bit indicators at the top of the table indicate the bit position of the source register shown in the blocks.

Table 2-7 is described as follows:

1) The 2 digit hex code on the left indicates the low order 8 bits of the Load AP Register from Register instruction.

2) The bit indicators at the top of the table indicate the bit position of the destination register shown in the blocks.

Table 2-6. Load AP Register from AP Register Source Options

OPCODE	SOURCE REGISTER 0 7	8 15	16 23	24 31
300	C0	C1	C2	C3
301	C1	C2	C3	C0
302	C2	C3	C0	C1
303	C3	C0	C1	C2
304	AS0	AS1	AS2	AS3
305	AS1	AS2	AS3	AS0
306	AS2	AS3	AS0	AS1
307	AS3	AS0	AS1	AS2
308	BL0	BL1	DP0	DP1
309	BL1	DP0	DP1	BL0
30A	DP0	DP1	BL0	BL1
30B	DP1	BL0	BL1	DP0
30C	PC0	PC1	0	CC-IMASK
30D	PC1	0	CC-IMASK	PC0
30E	0	CC-IMASK	PC0	PC1
30F	CC-IMASK	PC0	PC1	0
310	FL2	FPE	0	0

OPCODE	SOURCE REGISTER 0 7	8 15	16 23	24 31
311	FPE	0	0	FL2
312	0	0	FL2	FPE
313	0	FL2	FPE	0
31C	FL1	FP3	FP1	FP2
31D	FP3	FP1	FP2	FL1
31E	FP1	FP2	FL1	FP3
31F	FP2	FL1	FL3	FP1

Table 2-7. Load AP Register from AP Register Destination Options

DESTINATION REGISTER

OPCODE	0 7	8 15	16 23	24 31
08			C2	C3
09			AS2	AS3
0A			DP0	DP1
0B				CC-IMASK
0C	FL2			
0D				FP2
0E			FL1	
0F			FP1	FP2
10	C0	C1		
11	AS0	AS1		
12	BL0	BL1		
13	PC0	PC1		
14		FPE		
15	FL1			
16		FP3		
17	FL1	FP3		
18	C0	C1	C2	C3

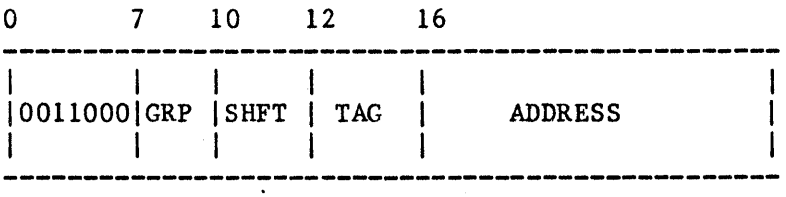
DESTINATION REGISTER

OPCODE	0 7	8 15	16 23	24 31
19	AS0	AS1	AS2	AS3
1A	BL0	BL1	DP0	DP1
1B	PC0	PC1		CC-IMASK
1C	FL2	FPE		
1D	FL1			FP2
1E		FP3	FP1	
1F	FL1	FP3	FP1	FP2

STORE AP CONTROL REGISTER TO CONTROL MEMORY

The store register instruction stores AP control registers into main memory or the HSDB. The effective address is formed by using the TAG and ADDRESS fields of the instruction. The source may be any of the AP register groups.

Instruction Format



GRP	Source Register Group
000	Common
001	AS
010	BL, DP
011	PSW
100	FL2, FPE
101	FL1, FP3, FP1, FP2
110	FL1, FP3, FP1, FP2
111	FL1, FP3, FP1, FP2

SHFT	Shift Amount
00	No shift
01	Left shift 8 bits
10	Left shift 16 bits
11	Left shift 24 bits

TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

Table 2-8 is describes as follows:

1) The 3 digit hex code on the left indicates the high order 12 bits of the Store AP Register instruction.

2) The bit indicators at the top of the table indicate the bit positions of the source register to be stored into memory.

Table 2-8. Store AP Register to Control Memory Source Options

OPCODE	32-BIT VALUE 0 7	8 15	16 23	24 31	
300	C0	C1	C2	C3	
301	C1	C2	C3	C0	
302	C2	C3	C0	C1	
303	C3	C0	C1	C2	
304	AS0	AS1	AS2	AS3	
305	AS1	AS2	AS3	AS0	
306	AS2	AS3	AS0	AS1	
307	AS3	AS0	AS1	AS2	
308	BL0	BL1	DP0	DP1	
309	BL1	DP0	DP1	BL0	
30A	DP0	DP1	BL0	BL1	
30B	DP1	BL0	BL1	DP0	
30C	PC0	PC1	0	CC-IMASK	*
30D	PC1	0	CC-IMASK	PC0	*
30E	0	CC-IMASK	PC0	PC1	*
30F	CC-IMASK	PC0	PC1	0	*
310	FL2	FPE	0	0	

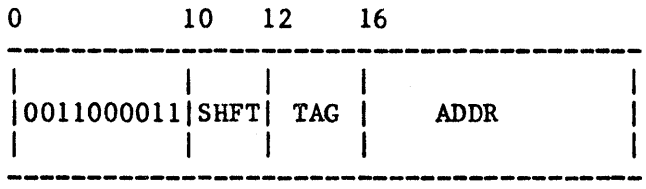
OPCODE	32-BIT VALUE 0 7	8 15	16 23	24 31
311	FPE	0	0	FL2
312	0	0	FL2	FPE
313	0	FL2	FPE	0
31C	FL1	FP3	FP1	FP2
31D	FP3	FP1	FP2	FL1
31E	FP1	FP2	FL1	FP3
31F	FP2	FL1	FP3	FP1

* SWAP PSW

SWAP PSW (Special case of store AP register to memory)

The swap PSW operation causes the current program status word to be stored in the memory location specified by the effective address and a new PSW value to be fetched from the succeeding location. (If shifting is specified, both the old and new PSW values will be shifted.) The result is a change in the PC, CC, and IMASK. The next instruction is fetched from the location specified in the left half of the PSW (PC). The location where the current PSW is store must not be a page memory address. NOTE: A swap PSW occurs anytime the PC, CC-IMASK, or PSW are stored into memory. To prevent a swap, move them into another register, then store that register into memory. See Table 2-8 for swap PSW option of Store AP register to memory.

Instruction Format



SHFT	Shift Amount
00	No shift
01	Left shift 8 bits
10	Left shift 16 bits
11	Left shift 24 bits

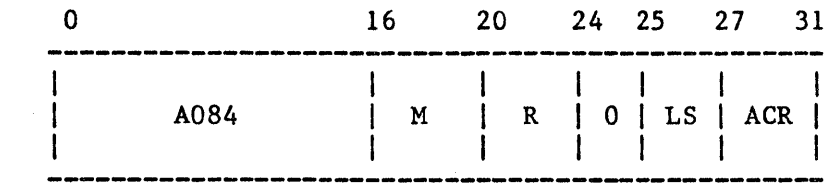
TAG	Effective Address
0001	ADDR
0010	ADDR+DP
0011	ADDR+DP, decrement BL
0100	ADDR+DP, increment DP
0101	ADDR+DP, decrement BL and increment DP
0110	ADDR+DP, decrement DP
0111	ADDR+DP, decrement BL and DP
1000	ADDR+R0
1001	ADDR+R1
1010	ADDR+R2
1011	ADDR+R3
1100	ADDR+R4
1101	ADDR+R5
1110	ADDR+R6
1111	ADDR+R7

GENERAL REGISTER INSTRUCTIONS

LOAD AP CONTROL REGISTER FROM GENERAL REGISTER OR CONTROL MEMORY

The Load AP control register instruction loads an AP control register or register group from a general register or a memory location pointed to by a general register.

Instruction Format



M — Source mode field. The value in this field selects the address mode used to fetch the source data. The data may be fetched directly from the source register R (register mode), R may be a pointer to the memory location containing the data (indirect mode), or R may be a pointer to a memory location which points to the data (double indirect mode).

M also determines if the source register R is to be incremented or decremented before or after the load operation.

The legal values for M are

1 register mode with no modification
2 indirect mode with no modification
3 double indirect mode with no modification
4 indirect mode with post-increment
5 indirect mode with post-decrement
6 indirect mode with pre-increment
7 indirect mode with pre-decrement
8 double indirect mode with post-increment
9 double indirect mode with post-decrement
A double indirect mode with pre-increment
B double indirect mode with pre-decrement
C register mode with post-increment
D register mode with post-decrement
E register mode with pre-increment
F register mode with pre-decrement

R — Source register field. This field specifies the general register (R0-RF) to be used as the source register. The interpretation of the content of R is determined by the mode field M.

LS — The LS field specifies an end-around left shift of 0, 1, 2, or 3 bytes which is applied to the source data before it is stored.

For determining the shift amounts, Figure 2-1 shows the allignment of the AP control registers.

ACR — Destination field. This field specifies the destination AP control register or register group.

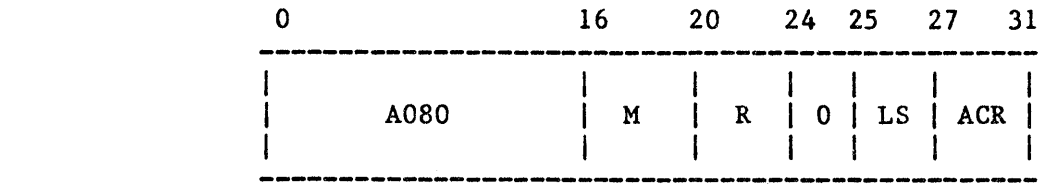
The legal values for ACR are

08	CL
09	ASL
0A	DP
0B	CC-IMASK
0C	FL2
0D	FP2
0E	FP1
0F	FP1, FP2
10	CH
11	ASH
12	BL
14	FPE
15	FL1
16	FP3
17	FL1, FP3
18	CH, CL
19	ASH, ASL
1A	BL, DP
1C	FL2, FPE
1D	FL1, FP3, FP1, FP2
1E	FL1, FP3, FP1, FP2
1F	FL1, FP3, FP1, FP2

STORE AP CONTROL REGISTER TO GENERAL REGISTER OR CONTROL MEMORY

The store AP control register instruction stores an AP control register or register group to a general register or a memory location pointed to by a general register.

Instruction Format



M — Destination mode field. The value in this field selects the address to be used as the destination. The data may be stored directly into destination register R (register mode), R may be a pointer to the destination memory location (indirect mode), or R may be a pointer to a memory location which points to the destination memory location (double indirect mode).

M also determines if the destination register R is to be incremented or decremented before or after the store operation.

The legal values for M are

1	register mode with no modification
2	indirect mode with no modification
3	double indirect mode with no modification
4	indirect mode with post-increment
5	indirect mode with post-decrement
6	indirect mode with pre-increment
7	indirect mode with pre-decrement
8	double indirect mode with post-increment
9	double indirect mode with post-decrement
A	double indirect mode with pre-increment
B	double indirect mode with pre-decrement
C	register mode with post-increment
D	register mode with post-decrement
E	register mode with pre-increment
F	register mode with pre-decrement

R — Destination register field. This field specifies the general register (R0-RF) to be used as the destination register. The interpretation of the content of R is determined by the mode field M.

LS — The LS field specifies an end-around left shift of 0, 1, 2, or 3 bytes which is applied to the source data before it is stored.

For determining the shift amount LS, Figure 2-1 shows the alignment of the AP control registers.

ACR — Source field. This field specifies the source AP control register or register group.

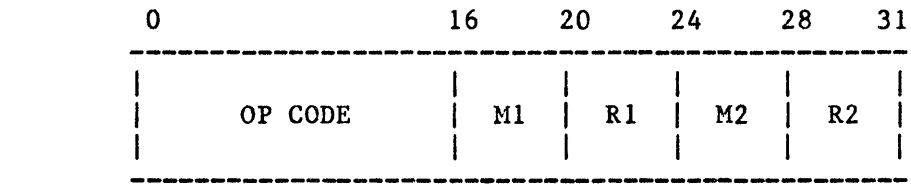
The legal values for ACR are

0	CH, CL
1	ASH, ASL
2	BL, DP
4	FL2, FPE
5	FL1, FP2
6	FP3, FP1
7	FP1, FP3, FP1, FP2

MOVE GENERAL REGISTER OR CONTROL MEMORY TO GENERAL MEMORY OR CONTROL MEMORY

The general register to general register instructions provide a means of moving data between any two of the 16 general registers (R0-RF), or between two memory locations pointed to by the general registers, or between a general register and memory. The memory specified must be a location in main memory or HSDB.

Instruction Format



OP CODE	Function
A000	Load register from register. This instruction moves the 32-bit value referenced in M1-R1 to the destination specified by M2-R2.
A001	Load register halfword. This instruction moves the least significant sixteen bits of the M1-R1 source to the least significant 16 bits of the M2-R2 destination. The source and the most significant 16 bits of the destination remain unchanged.
A002	Load register byte. This instruction moves the least significant 8 bits of the M1-R1 source to the most significant 8 bits of the M2-R2 destination. The source and least significant 24 bits of the destination remain unchanged.
A00B	Load complement of high order byte. This instruction complements the most-significant byte of the M1-R1 source and stores the result in the M2-R2 destination. The source and the low order 3 bytes of the destination remain unchanged.
M1	Source mode field. The value in this field selects the address mode used to fetch the source data. The data may be fetched directly from the source register R1 (register mode), R1 may be a pointer to the memory location containing the

data (indirect mode), or R1 may be a pointer to a memory location which points to the data (double direct mode).

M1 also determines if the source register R1 is to be incremented or decremented before or after the move operation.

The legal values for M1 are

1 register mode with no modification
2 indirect mode with no modification
3 double indirect mode with no modification
4 indirect mode with post-increment
5 indirect mode with post-decrement
6 indirect mode with pre-increment
7 indirect mode with pre-decrement
8 double indirect mode with post-increment
9 double indirect mode with post-decrement
A double indirect mode with pre-increment
B double indirect mode with pre-decrement
C register mode with post-increment
D register mode with post-decrement
E register mode with pre-increment
F register mode with pre-decrement

R1 Source register field. This field specifies the general register (R0-RF) to be used as the source register. The interpretation of the content of R1 is determined by the source mode field M1.

M2 Destination mode field. The value in this field selects the address to be used as the destination. The data may be stored directly into destination register R2 (register mode), R2 may be a pointer to the destination memory location (indirect mode), or R2 may be a pointer to a memory location which points to the destination memory location (double indirect mode).

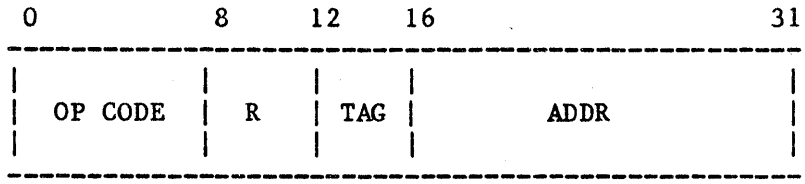
M2 also determines if the destination register R2 is to be incremented or decremented before or after the move operation. The legal values for M2 are the same as for M1.

R2 Destination register field. This field specifies the general register (R0-RF) to be used as the destination register. The interpretation of the content of R2 is determined by the destination mode field M2.

LOAD GENERAL REGISTER FROM CONTROL MEMORY

The load general register from memory instructions load the general registers with data from main memory or HSDB.

Instruction Format



OP CODE	Function
81	Load register from memory. This instruction moves a 32-bit data value from the control memory location specified by TAG and ADDR to the selected register (R). The source value remains unchanged.
82	Load halfword from memory. This instruction moves the least-significant 16 bits of the memory value specified by TAG and ADDR to the least-significant 16-bits of the selected register (R). The source and the most-significant 16-bits of the destination remain unchanged.
83	Load byte from memory. This instruction moves the least-significant byte of the source specified by TAG and ADDR to the most-significant byte of the selected register (R). The source and the least significant 24 bits of the destination remain unchanged.
R	R is the destination register field. Its value (0-F) selects the general register to be loaded.

TAG — The value in the TAG field combined with ADDR forms the effective destination address.

The legal values for TAG are

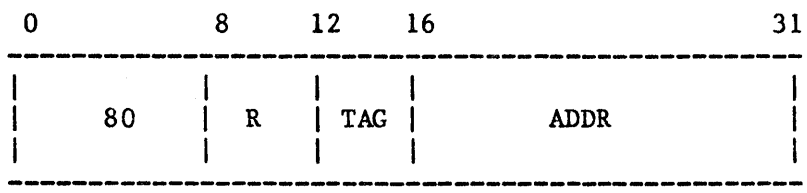
TAG	Effective Address
0	Load immediate value
1	ADDR
2	ADDR + DP
3	ADDR + DP, Decrement BL
4	ADDR + DP, Increment DP
5	ADDR + DP, Increment DP, Decrement BL
6	ADDR + DP, Decrement DP
7	ADDR + DP, Decrement DP and BL
8	ADDR + R0
9	ADDR + R1
A	ADDR + R2
B	ADDR + R3
C	ADDR + R4
D	ADDR + R5
E	ADDR + R6
F	ADDR + R7

Note that when TAG=0, the value of ADDR is loaded into the specified general register R.

STORE GENERAL REGISTER TO CONTROL MEMORY

The store general register instruction stores the content of the general register R into the control memory location in main memory or HSDB specified by the TAG and ADDR fields.

Instruction Format



R — R is the source register field. Its value (0-F) selects the general register which contains the data to be stored.

TAG — The value in the TAG field combined with ADDR forms the effective destination address.

The legal values for TAG are

TAG	Effective Address
0	ADDR
1	ADDR
2	ADDR + DP
3	ADDR + DP, Decrement BL
4	ADDR + DP, Increment DP
5	ADDR + DP, Increment DP, Decrement BL
6	ADDR + DP, Decrement DP
7	ADDR + DP, Decrement DP and BL
8	ADDR + R0
9	ADDR + R1
A	ADDR + R2
B	ADDR + R3
C	ADDR + R4
D	ADDR + R5
E	ADDR + R6
F	ADDR + R7

Note that TAG=0 is equivalent to TAG=1.

SECTION III. PROGRAM PAGER INSTRUCTIONS

GENERAL

PROGRAM SEQUENCE

Pager operation is initiated by an external function code (Load GET). When the pager is first put in the busy state, the GET address register points to the location of the first pager instruction. Normally, the pager fetches instructions from sequential memory locations, and the GET address register is incremented by 1 for each instruction (operation is similar to that of the Program Counter in AP control). The execution sequence is modified if a Move Data instruction is fetched. In this case, the memory locations immediately following the Move Data command contain the source block to be moved, and the next pager instruction immediately follows this source data block.

The execution sequence is also modified when a Load GET external function is issued by the pager or another STARAN element. In this case, the GET address register is loaded with a new address from which the next instruction is fetched.

The execution sequence is halted when an external function to stop the pager is issued by the pager or another STARAN element.

INSTRUCTION LENGTH

Each pager instruction is a 32-bit word. Bits 0 and 1 determine the instruction type.

INSTRUCTION TYPES

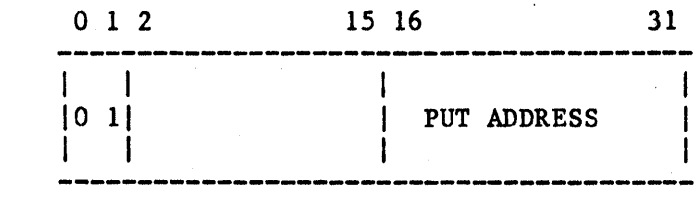
The pager instructions are: Load Put, Move Data, Load Put and Move Data, and Issue EXF.

PAGER INSTRUCTIONS

LOAD PUT

The Load PUT instruction loads the PUT address register with the address field (bits 16-31) of the instruction. The first word transferred by the next Move Data instruction will be put into this address.

Instruction Format

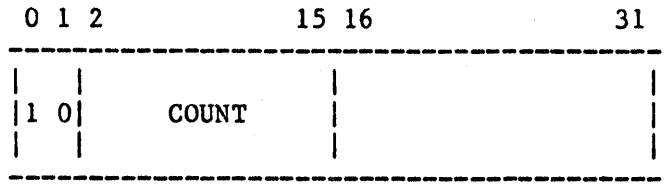


PUT ADDRESS = AP control page memory address

MOVE DATA

The Move Data instruction causes memory words immediately following the instruction to be loaded into the page memory. The block length is contained in the count field of the command. The first instruction of the block is loaded into the location pointed to by the PUT Address register. Succeeding instructions are loaded in sequential page memory locations. At the end of the Move Data execution, the PUT Address register points to the location following the last word of the destination block and the GET Address register points to the location following the last word of the source block where the next instruction is fetched.

Instruction Format

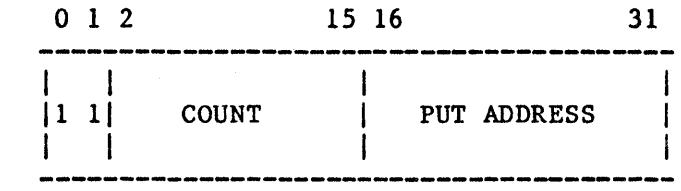


COUNT = Block Count (number of words to be moved)

LOAD PUT AND MOVE DATA

The Load PUT and Move Data instruction loads the PUT Address register and then causes a block of data to be moved. It performs the same function as a Load PUT instruction followed by a Move Data instruction.

Instruction Format



COUNT = Block Count (number of words to be moved)

PUT ADDRESS = AP control page memory address

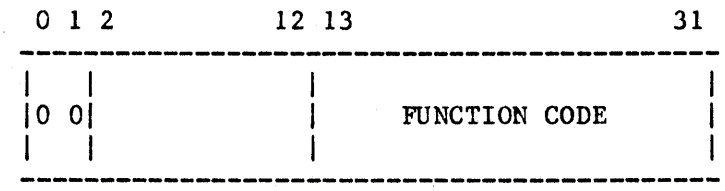
ISSUE EXF

When the pager issues an external function, it competes with other STARAN elements that are issuing external functions. The EXF logic treats one EXF at a time. If the EXF logic accepts one EXF from another element (which affects the pager) before the pager EXF is accepted, the pager EXF will be ignored.

When the pager issues a function code to the EXF logic, and the EXF logic accepts the code, it returns a sense bit dependent on the particular code and status of the STARAN element. If the sense bit is 0, the pager fetches its next instruction from the next sequential memory location. If the sense bit is 1, a skip of one instruction is performed and the GET address register is incremented by 2 instead of 1 at the end of instruction execution.

Certain external functions affect the pager itself. These codes and their functions are shown in Table 2-9.

Instruction Format



FUNCTION CODE = any one of the valid external function codes described in Appendix C.

Table 2-9. Pager EXF Functions

EXF CODE	Function
4aaaa	Causes the pager to branch to hex address aaaa
08005	Causes the next sequential instruction to be skipped
08000	Causes the pager to halt
08001	Causes the pager to halt with the GET address register pointing to the second location after the EXF instruction

PAGER COMMAND SUMMARY

A summary of Pager commands is shown in Table 2-10.

Table 2-10. Summary of Pager Commands

Hex Code (Bits 0-31)	Operation
4000nnnn	Load PUT Address register
8kkk0000 (1)	Move block from bulk core to page memory
Ckkknnnn (1)	Load PUT Address register and move block from bulk core to page memory
000fffff	Issue external function from pager
0004aaaa	Branch to new GET address
00008001	Halt Pager and skip next sequential instruction
00080005	Skip to next sequential instruction

where nnnn = page address
kkk = number of words to be moved (count)
fffff = 19 bit function code
aaaa = hexadecimal code for new address

(1) The count may extend into the left-most digit since the count field is actually 14 bits in length.

SECTION IV. EXTERNAL FUNCTION INSTRUCTIONS

GENERAL

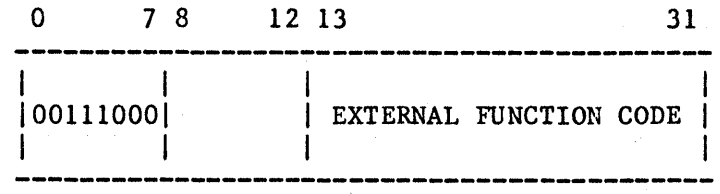
The External Function instruction issues an external function (EXF) code to EXF logic. If the EXF logic returns a sense bit of 1, the next sequential instruction is skipped; if the sense bit is 0, the next sequential instruction is executed. Each function code is 19 bits long and indicates what element of STARAN is to be interrogated and/or controlled. Function codes may be transmitted to EXF logic by AP control, the program pager, sequential control, or the host computer (if the EXF channel is implemented). One function code at a time is accepted by EXF logic. A function code can both interrogate and control an element in one operation.

FUNCTION CODE CLASSES.

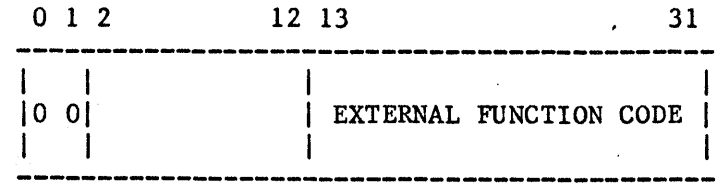
The classes of function codes are Page Port Switches, Interlocks, Program Pager, Error Control, AP Control interrupts, Sequential Control interrupts, AP Control activity, resets and clears, and spare external functions.

INSTRUCTION FORMATS

Instruction Format From Associative Processor Control



Instruction Format From Program Pager

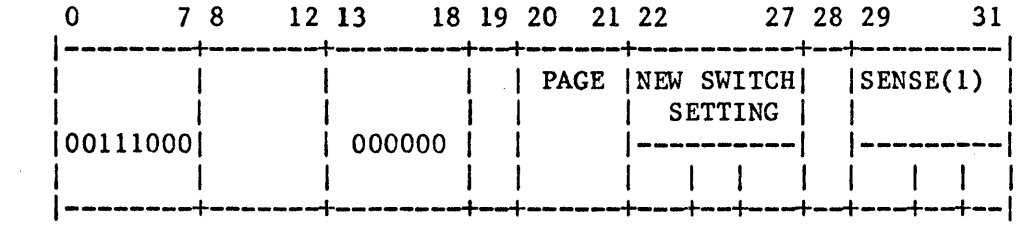


EXTERNAL FUNCTION CODES

PAGER PORT SWITCH INSTRUCTION

Each page memory has a port switch which can connect the memory to the AP control instruction bus, the pager bus, or the sequential control bus. This function code permits interrogation and control of these switches.

Instruction Format



Bits 20-21 00 = Page 0
01 = Page 1
10 = Page 2

Bits 22-23 If now on sequential bus
00 = No-op
01 = Switch to sequential bus
10 = Switch to instruction bus
11 = Switch to pager bus

Bits 24-25 If now on instruction bus
00 = No-op
01 = Switch to sequential bus
10 = Switch to instruction bus
11 = Switch to pager bus

Bits 26-27 If now on pager bus
00 = No-op
01 = Switch to sequential bus
10 = Switch to instruction bus
11 = Switch to pager bus

Bits 29-31 100 = On sequential bus
010 = On instruction bus
001 = On pager bus

(1) Only one of bits 29-31 is returned as a sense bit.

A summary of Pager Port Switch EXF instructions is shown in Table 2-11.

Table 2-11. Summary of Pager Port Switch EXF Instructions

Hex Code (Bits 0-31)	Operation
380002A0	Switch Page 0 to Instruction Bus
380006A0	Switch Page 1 to Instruction Bus
38000AA0	Switch Page 2 to Instruction Bus
380003F0	Switch Page 0 to Pager Bus
380007F0	Switch Page 1 to Pager Bus
38000BF0	Switch Page 2 to Pager Bus
38000150	Switch Page 0 to Sequential Bus
38000550	Switch Page 1 to Sequential Bus
38000950	Switch Page 2 to Sequential Bus
380001B2	Sense if Page 0 on Instruction Bus
380005B2	Sense if Page 1 on Instruction Bus
380009B2	Sense if Page 2 on Instruction Bus
380001B1	Sense if Page 0 on Pager Bus
380005B1	Sense if Page 1 on Pager Bus
380009B1	Sense if Page 2 on Pager Bus
380001B4	Sense if Page 0 on Sequential Bus
380005B4	Sense if Page 1 on Sequential Bus
380009B4	Sense if Page 2 on Sequential Bus

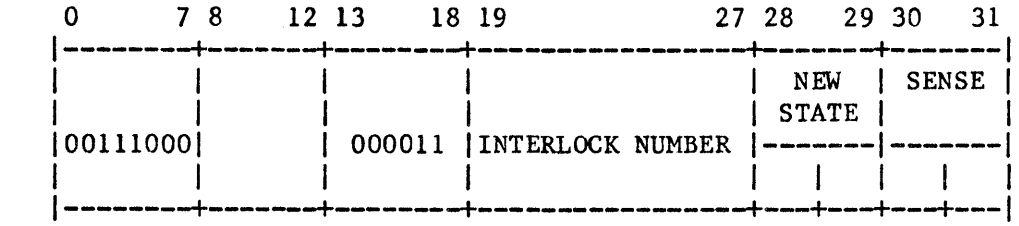
INTERLOCKS

The EXF logic contains 64 stored bits called interlocks. These bits have no predetermined meaning. Software can assign a meaning to an interlock and use it for any purpose. Function codes allow the current state of an interlock to be sensed and a new state entered in one operation. The current state of the selected interlock is sensed, and depending on the state, either bit 30 or bit 31 is returned as a sense bit. The current state also selects bit 28 or bit 29 as the new state of the interlock. If the selected bit for the new state is 0, the interlock is cleared; if the selected bit for the new state is 1, the interlock is set.

Interlock Switches and Lights

Sixteen interlocks (00 through 0F) can be controlled manually by panel switches and are displayed via panel lights. The other 48 interlocks (10 through 3F) can be sensed and controlled only by software function codes. A summary of interlock EXF instructions is shown in Table 2-12.

Instruction Format



INTERLOCK NUMBER 00 through 3F

NEW STATE

00	Clear interlock n
01	Leave interlock n unchanged
10	Toggle interlock n
11	Set interlock n

SENSE

00	Sense returned is zero
01	Sense returned is one if interlock n is set
10	Sense returned is one if interlock n is reset
11	Sense returned is one

Table 2-12. Summary of Interlock EXF Instructions

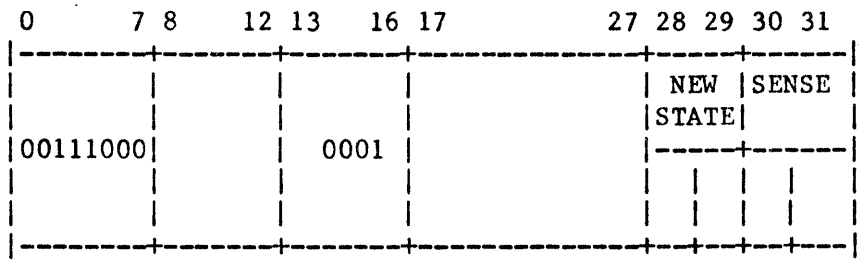
Hex Code (Bits 0-31)	Operation
38006nnC	Set Interlock nn
38006nn0	Clear Interlock nn
38006nn5	Sense if Interlock nn set
38006nn6	Sense if Interlock nn clear
38006nn1	Sense if Interlock nn set, then clear it
38006nnD	Sense if Interlock nn set, then set it
38006nn2	Sense if Interlock nn clear, then clear it
38006nnE	Sense if Interlock nn clear, then set it

where $00 \leq nn \leq 3F$

PAGER STATE INSTRUCTION

The program pager has two states, off and busy. The Pager State instruction allows the state of the pager to be sensed and changed. If the current pager state is off, bit 30 is returned as a sense bit and bit 28 governs the new pager state; if the pager is busy, bit 31 is returned and bit 29 governs the new pager state. If the selected bit for the new state is 1, the pager will become busy; if the selected bit is 0, the pager will be turned off.

Instruction Format



Bit 28	If Off
0	Pager remains off
1	Pager will become busy (if operative)
Bit 29	If Busy
0	Pager turned off
1	Pager remains busy
Bits 30-31	SENSE
00	Sense returned is zero
01	Sense returned is one if pager state is busy
10	Sense returned is one if pager state is off
11	Sense returned is one

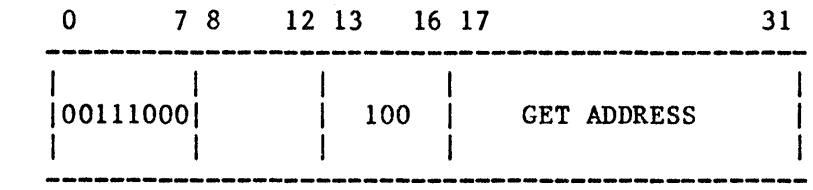
Sample Coding

Hex Code (Bits 0-31)	Operation
38008000	Stop pager.
3800800C	Start pager where previously stopped (if operative).
38008005	Sense if pager is busy.
38008006	Sense if pager off.

PAGER LOAD GET INSTRUCTION

The Pager Load GET instruction loads the pager GET address register with a new address. The sense bit returned for this function code will always be zero. If the pager is in the midst of moving data or loading the PUT address and moving data, the moving will be interrupted between word transfers. If the pager was issuing an EXF, the EXF will not be processed if EXF logic has accepted the Load GET EXF first.

Instruction Format



GET ADDRESS — Address to be loaded into the GET address register

If pager is inoperative, its GET address register is not disturbed and pager remains inoperative.

If pager is operative, its GET address register is loaded with bits 16-31 of the instruction and the count register is cleared, the pager becomes busy and fetches its first instruction from the new get address.

Sample Coding

Hex Code (Bits 0-31)	Operation
3804nnnn	Load pager GET address register.

where nnnn is the GET address

ASSOCIATIVE PROCESSOR CONTROL INTERRUPTS

The AP Control Interrupt instruction is used to sense, set, and reset the state of the 15 interrupts to AP control. AP control interrupts are given hex numbers 01 (lowest priority) to 0F (highest priority).

Interrupt Mask

Bits 28 through 31 of the Program Status Word (PSW) in AP control contain an Interrupt Mask (IMASK).

Interrupt Conditions

AP control accepts interrupt n if the following conditions exist

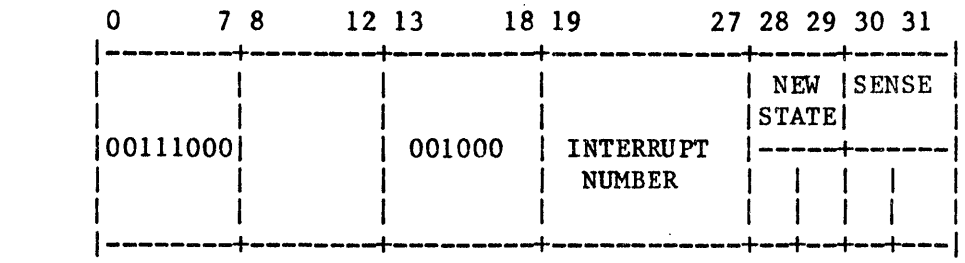
1) AP control is active and at an interruptible point.
2) The current IMASK is less than n.
3) No interrupt of higher priority is set.
4) Interrupt n is set.

Interrupt Handling

When interrupt n is accepted, AP control fetches the next instruction from hex address 0000+n without altering the program counter. Normally, the instruction at the above address is a swap PSW which saves the old PSW and loads a new PSW, causing control to be transferred to an interrupt handling routine. The IMASK of the new PSW should be n or greater to prevent AP control from accepting the n interrupt again until the interrupt handling routine is complete.

The current state of the interrupt is used to select bit 30 or 31 to be returned as a sense bit and to select either bit 28 or 29 for the new state of the interrupt. If the selected bit for the new state is 1, the interrupt is left in the set state; if it is 0, the interrupt is left in the clear state.

Instruction Format



INTERRUPT NUMBER 00 through 0F

NEW STATE

00	Reset interrupt n
01	Leave the state of interrupt n unchanged
10	Toggle interrupt n
11	Reset interrupt n

SENSE

00	Sense returned is zero
01	Sense returned is one if interrupt n is set
10	Sense returned is one if interrupt n is reset
11	Sense returned is one

Sample Coding

Hex Code (Bits 0-31)	Operation
38010nnC	Set AP Interrupt nn
38010nn0	Clear AP Interrupt nn
28010nn5	Sense if AP Interrupt nn is set
38010nn6	Sense if AP Interrupt nn is clear

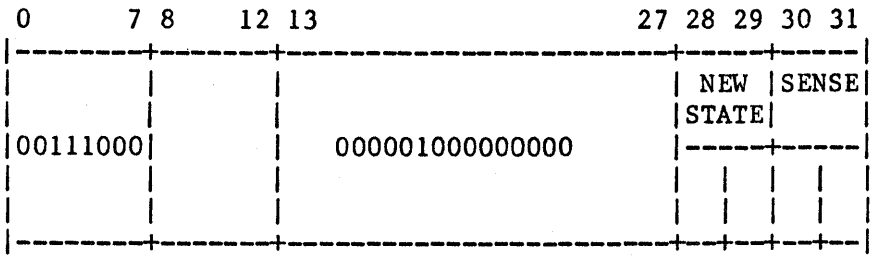
where nn is the interrupt number

ASSOCIATIVE PROCESSOR CONTROL ACTIVITY

The AP Control Activity instruction permits the sensing and controlling of AP activity. The AP control has two states: active and inactive. In the active state AP control fetches instructions from AP control memory and exercises the associative arrays. When switched from the active to the inactive state, all AP control registers remain as they were after executing the last instruction. When going from the inactive to the active state, AP control fetches its first instruction from location 0000 without altering the Program Counter. Thus, if 0000 contains a No-op instruction, AP control would continue with its previous program. If 0000 contains a branch, a new program sequence would be entered. If 0000 contains a swap PSW a new program sequence would be entered while saving the old Program Counter status to allow future re-entry into the old program sequence.

The current state of AP activity is sensed and bit 30 or 31 is returned as a sense bit; bit 28 or 29 is selected as the new state. If the selected bit for the new state is 0, the AP will become inactive; if the selected bit is 1, the AP will become active.

Instruction Format



NEW STATE

00	Clear AP activity
01	Leave AP activity unchanged
10	Toggle AP activity
11	Set AP activity

SENSE

00	Sense returned is zero
01	Sense returned is one if AP is active
10	Sense returned is one if AP is inactive
11	Sense returned is one

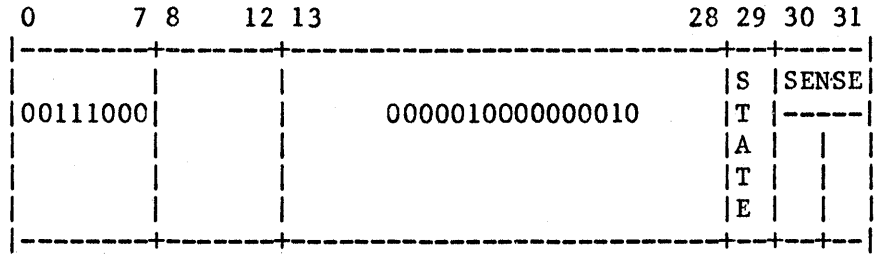
Sample Coding

Hex Code (Bits 0-31)	Operation
38002000	Halt AP (AP becomes inactive).
3800200C	Start AP at location 0000 (AP becomes active).
38002005	Sense if AP is active.
38002006	Sense if AP is inactive.

ASSOCIATIVE PROCESSOR CONTROL LOOP INDICATOR

The AP control loop indicator is set and remains set until all repetitions of a loop are completed. This function code allows the loop indicator to be sensed and/or cleared. Clearing the set loop indicator causes AP control to terminate the loop, even if all repetitions have not been completed. The current state of the loop indicator is sensed and bit 30 or 31 is returned as a sense bit; if bit 29 is set to 1, the indicator is cleared; otherwise, it remains unchanged.

Instruction Format



STATE

1	Reset loop indicator
0	Leave loop indicator unchanged

SENSE

00	Sense returned is zero
01	Sense returned is one if loop is set
10	Sense returned is one if loop is reset
11	Sense returned is one

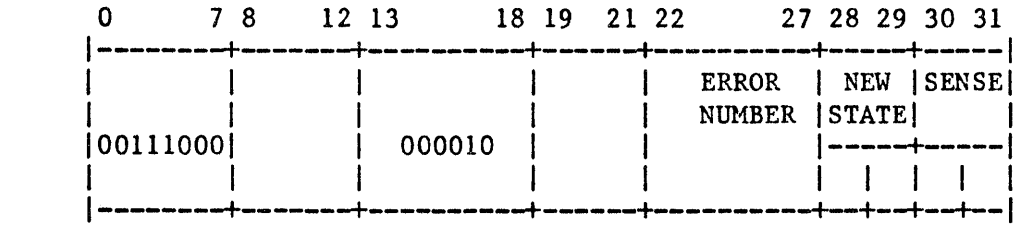
Sample Coding

Hex Code (Bits 0-31)	Operation
38002011	Sense if AP is in loop mode.
38002012	Sense if AP is not in loop mode.
38002014	Take AP out of loop mode.

ERROR CONTROL

The Error Control instruction will sense and/or set or reset error indicators. Error detectors are included in various elements of STARAN to sense hardware faults and program errors. Each error detector sets an error indicator when an error is detected. Each error indicator is given a number by which it may be sensed, set, and/or reset. If any error indicator is set, an interrupt to sequential control is generated. Certain errors will make the pager inoperative and the AP control inactive.

Instruction Format



ERROR NUMBER — One of the valid AP error numbers described in Table 2-13.

NEW STATE

00	Clear error indicator n
01	Leave error indicator n unchanged
10	Toggle error indicator n
11	Set error indicator n

SENSE

00	Sense returned is zero
01	Sense returned is one if error indicator n is set
10	Sense returned is one if error indicator n is reset
11	Sense returned is one

Sample Coding

Hex Code (Bits 0-31)	Operation
38004091	Test if there was a parity error on the AP control instruction bus; if yes, reset error indicator 09.

Table 2-13. AP Error Indicators

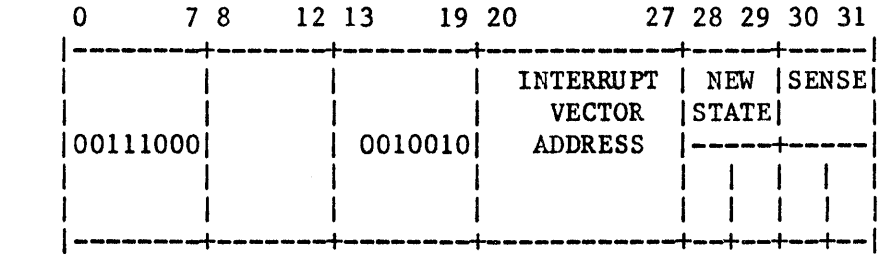
AP ERROR NUMBER (Hexadecimal)	ERROR DESCRIPTION	ACTION WHEN SET(1)
00	Illegal instruction in AP control	AP control inactive
01	Instruction bus inactivity time out error	AP control inactive
02	Buffered I/O bus hung up	-
03	AP control data bus hung up	AP control inactive
04	AP control instruction bus hung up	AP control inactive
05	Pager GET bus hung up	Pager inoperative
06	Pager PUT bus hung up	Pager inoperative
07	Parity error on SCC	-
08	Parity error on AP control data bus	AP control inactive
09	Parity error on AP control instruction in bus	AP control inactive
0B	Parity error on sequential control bus	-
0C	RPU time out error	-

(1) The current state also selects either bit 28 or 29 and loads the selected bit into the error indicator as a new state. If the selected new state bit is 1, the error indicator will be set. If the selected bit is 0, the error indicator will be cleared.

SEQUENTIAL CONTROL INTERRUPT

The Sequential Control Interrupt instruction can sense, set, and reset the state of eight interrupts to sequential control. The current state of the selected interrupt is sensed and, depending on the state, either bit 30 or bit 31 is returned as a sense bit. The current state also selects bit 28 or bit 29 and uses the selected bit as the new state of the interrupt. If the selected bit for the new state is 1, the interrupt is left in the set state; if the selected bit is 0, the interrupt is left in the clear state. When an interrupt occurs, control is transferred to the Sequential Controller at the interrupt vector address shown in Table 2-14.

Instruction Format



INTERRUPT VECTOR ADDRESS — One of the interrupt vector addresses defined in Table 2-14.

NEW STATE

00	Clear interrupt n
01	Leave interrupt n unchanged
10	Toggle interrupt n
11	Set interrupt n

SENSE

00	Sense returned is zero
01	Sense returned is one if interrupt n is set
10	Sense returned is one if interrupt n is reset
11	Sense returned is one

Sample Coding

Hex Code (Bits 0-31)	Operation
38012nnC	Set Sequential Interrupt nn
38012nn0	Clear Sequential Interrupt nn
38012nn5	Sense if Sequential Interrupt nn is set
38012nn6	Sense if Sequential Interrupt nn is clear

where C0≤nn≤DC

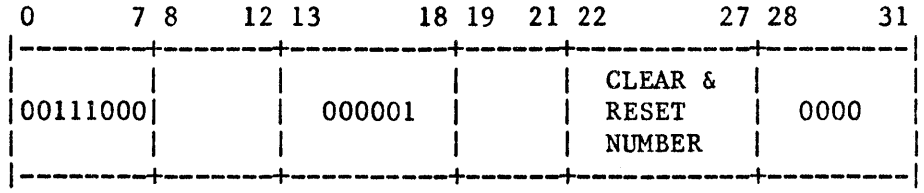
Table 2-14. Sequential Control Interrupt Vector Addresses

INTERRUPT VECTOR ADDRESS	VECTOR ADDRESS (OCTAL)	PRIORITY	REMARKS
C0	300	4	
C4	304	4	
C8	310	5	Panel Interrupt
CC	314	5	Errors
D0	320	6	PFM Event Overflow
D4	324	6	PFM Timer Overflow
D8	330	7	STARAN Command Channel
DC	334	7	

RESETS AND CLEARS

The reset and clear function codes are used for resetting and clearing various STARAN logic and status bits. They are used for clearing hangup conditions and initial clearing when STARAN power is turned on. Because certain clears will clear out the logic issuing the EXF, these function codes are not usually issued by AP control or by the program pager.

Instruction Format



Sample Coding

Hex Code (Bits 0-31)	Operation
38002(n)0	Reset n (02≤n≤E)
38002020	Clear SCC Devices
38002030	Clear Errors, Sequential Interrupts
38002040	Clear AP Interrupts
38002050	Clear DMA Logic
38002060	Clear Core Logic
38002070	Clear Next Instruction Logic
3800208C	Clear Branch Logic
38002090	Clear Register Logic
380020A0	Clear Pager Logic
380020B0	Clear HSDB Logic
380020C0	Clear Sequential Control Interface
380020D0	Clear Associative Instruction Logic
380020E0	Clear Page 0 Logic
380020F0	Clear Page 1 Logic
38002100	Clear Page 2 Logic
38002110	Clear Register Processor Unit
38002300	Master Clear

SPARE EXTERNAL FUNCTIONS

Certain function codes are left as spare functions to be used by the custom I/O cabinet. They may be used for generating interrupts to a host computer or setting up I/O channels.

CHAPTER 3

INPUT/OUTPUT OPTIONS

GENERAL

A wide variety of input/output options are available with STARAN. Input/output (I/O) control in STARAN is located in an I/O cabinet which can be customized to fit the needs of a particular installation. Since the custom I/O cabinet will differ from installation to installation, this chapter discusses only what I/O options are available. The I/O controls for particular installations are discussed in a separate document.

One I/O option available with STARAN is integration with another computer system (called the host computer), either with Direct Memory Access (DMA), buffered I/O, external functions, STARAN Command Channel (SCC), or a combination of these. Another option available is interfacing with a wide variety of data gathering, data transmitting, or data storing devices with either the standard STARAN buffered I/O channel or the very high bandwidth parallel I/O channel. Error checking can also be included in the I/O cabinet to check for errors in data transmission. The different types of I/O options available are discussed in the following subsections.

DIRECT MEMORY ACCESS CHANNEL

Direct access to a host computer memory allows that memory to act as part of the AP control memory. Items in the host computer memory are accessible by the host computer and by STARAN, thus reducing the need for buffered I/O transfers from the host computer and STARAN.

Implementation of Direct Memory Access (DMA) from STARAN to the host computer depends on the characteristics of the host computer. A wide variety of computers can be accommodated. There are, however, some that do not support DMA without extensive modification. Cycle times with DMA will be somewhat longer than the normal cycle times of the host computer memory because of extra cable lengths and logic gating. If the host computer word length is less than 32 bits, the host computer word is padded with zeros to provide 32 bits when it is read by STARAN; the padded bits are truncated when the word is written by STARAN. If the host computer word length is more than 32 bits, the host computer word is truncated to 32 bits when it is read by STARAN; the truncated bits are padded with blanks when the word is written by STARAN.

Address translation may be required on the DMA interface to match the AP control addresses with the host computer addresses. This translation can be accommodated in the custom I/O cabinet.

The DMA block of addresses may be put to other uses besides access to a host computer memory. Possible uses include access to an external memory, which may or may not be accessible by other devices, and access to special I/O devices.

BUFFERED INPUT/OUTPUT CHANNEL

The BIO channel is a 32-bit wide input/output port on the STARAN computer. This port allows external devices to access the STARAN control memory.

The BIO channel is under control of the external device which initiates the transfer of either data or command words over the channel. This channel can be used to perform many operations. A few possible uses of the BIO channel are listed below.

1) The transfer of data to be processed from an external device to STARAN such as sensor inputs, etc.

2) The transfer of data to and from a host computer to STARAN for processing

3) The transfer of processed data to a display subsystem

4) The transfer of command information from STARAN to an intelligent peripheral

Devices placed on the BIO channel directly address the STARAN control memory. The BIO does not access the high speed control memory pages. It can only access Bulk Core, High Speed Data Buffer and DMA addresses. Communications between STARAN and the BIO devices normally take place through software established buffers in the STARAN control memory.

Of the several data buses accessing the High Speed Data Buffer (HSDB), the Bulk Core Memory (BCM), and the external memory accessed via the DMA channel, the priority resolver logic gives the BIO port highest priority.

EXTERNAL FUNCTION CHANNEL

GENERAL

The external function (EXF) channel is used extensively for direct communication with the control logic of peripherals connected to the custom I/O cabinet. Up to 19 bits of function code can be received simultaneously at the custom I/O from a large number of peripheral units. The custom I/O handles each of the function codes one by one on a priority basis until all have been processed.

HOST COMPUTER EXTERNAL FUNCTION INTERFACE

When STARAN is integrated with a host computer, the external function interface is used to pass control information back and forth.

If the host computer can accept external interrupts, STARAN can generate such interrupts by using spare external function codes. The custom I/O cabinet generates an interrupt to the host whenever these external function codes are decoded. A large number of separate interrupts can be accommodated if necessary.

STARAN can accept interrupts from a host computer if the host can generate signals on discrete output lines. The custom I/O cabinet will accept such signals and generate certain function codes to the external function logic in the mainframe which can cause interrupts or control other elements of STARAN.

If the host computer has an external function output (direct output, etc.) capability which can output a data item at least 19 bits wide, the custom I/O cabinet can accept the item and send it to the EXF logic in the mainframe as a function code. This allows the host computer to generate any external function code and exercise complete control of STARAN. Not only can it generate interrupts, but it can also activate and deactivate elements of STARAN, control interlock bits, initiate transfers, etc. If the host computer can accept one bit of data (for instance, a condition code bit) while it is outputting the 19 bits, then the sense bit output of the EXF logic can be sent back to the host, allowing it to interrogate various elements of STARAN.

BUFFERED INPUT/OUTPUT EXTERNAL FUNCTIONS

Buffered I/O can be easily initiated by means of EXF codes. Word counts, starting addresses, and other information can be packed into the specific format necessary to communicate with a peripheral device. This data is sent along with the command that initiates a sequence of events.

PARALLEL INPUT/OUTPUT EXTERNAL FUNCTIONS

Parallel I/O is initiated with EXF codes in much the same manner as BIO. However, PIO may need more data than can be packed in one or two EXF codes. Because of this, it may be desirable to use the EXF code to initiate an operation that transfers several data words over buffered I/O. These data words then allow a PIO transfer to be executed.

STARAN COMMAND CHANNEL

The STARAN Command Channel is the vehicle by which command and control information is passed to I/O processors. The SCC is driven by I/O instructions executed by STARAN. As new devices are added to the STARAN system the instruction set will be upgraded to effectively deal with them.

The SCC provides a data output and input path 36 bits wide including four bits of parity. The IOP number, function code, and device address are held in a 16-bit register called the SCCR. The function code and address are decoded by the IOP to determine the nature of the command. IOP generated interrupts are supported by the channel at levels 8, 9, A, and B in STARAN.

PARALLEL INPUT/OUTPUT CHANNEL

GENERAL

Each associative array can have up to 256 inputs and 256 outputs into the custom I/O cabinet. (An 8-array system could have 2,048 input and 2,048 output lines.) These can be used for various purposes. For example, it could greatly speed up inter-array data communication

through the shifting of array data, allow a very high bandwidth I/O device to communicate with STARAN, or allow any device to talk directly with the associative arrays. Two of these applications are discussed in the following subsections.

INTER-ARRAY DATA COMMUNICATION

The PIO lines for each array may be interconnected in one or more ways to effect the fast transfer of data between arrays. The control of these lines will be set up by various external function codes sent to the I/O cabinet from STARAN prior to the transfer. Without inter-array communications, data can still be transferred from one array to another by sending it through the Common register.

HIGH BANDWIDTH I/O

The basic width of the PIO is 256n, where n is equal to the number of associative arrays in the system (n can have a maximum value of 8). The custom I/O cabinet is capable of buffering and reformatting the data received from any peripheral device to match the width necessary to communicate with the STARAN associative array. In order to synchronize read and write operations with an external device operating through PIO, certain AP control instructions wait for sync pulses from the external device. As in inter-array communication, an external function code sent to the custom I/O will set up and initiate the PIO operation.

APPENDIX A

GLOSSARY OF TERMS AND ABBREVIATIONS

AND

Logical AND

AP

Associative Processor

AP CONTROL INTERRUPTS

AP control interrupts are given hex addresses from 01 (lowest priority) to 0F (highest priority). A class of external codes is used to sense, set, and reset the state of the 15 interrupts to AP control. Bits 28 through 31 of the Program Status Word (PSW) in AP control contain an Interrupt Mask. AP control accepts interrupt n if AP control is active and at an interruptible point, the interrupt mask is less than n, no interrupt of higher priority is set, and interrupt n is set. When it accepts interrupt n, it fetches its next instruction from hex address 0000+n without disturbing the contents of its program counter.

AP CONTROL MEMORY

The main function of AP control memory is to store assembled AP application programs. It can also store items of data and act as a buffer between AP control and other elements of STARAN. Each AP control memory word contains 32 bits of either data or instructions. Bit 0 is the left (most-significant) bit and bit 31 is the right (least-significant) bit of each word. Each word is given a 16-bit address.

ARRAY

The MDA array consists of two basic components: array storage and response store. Each basic array contains 36 square segments 256 words by 256 bits per word. Access may be made in either bit, word, or mixed mode. An entire word of 256 bits or a bit slice, bit n of all 256 words, may be accessed. Array input and output may be either 32 bits or 256 bits in parallel. Input data may be written into the array through a mask contained in the response store.

ARRAY ADDRESS SELECT

The array address select logic in AP array control selects either the Array Select register or Field Pointer 1. If FP1 is selected, the rightmost five bits of the pointer are decoded to indicate the one and only array to be enabled. Selection is made without modifying the contents of the Array Select register. Such operations as reading one item of data from an array or writing one item of data into an array enable only one of the associative arrays. The Array Select register selects arrays when more than one array participates in an associative operation.

ARRAY CONTROL

Control lines from AP control to the MDA arrays that are not generated by the response store control are generated by the array control. The array control logic selects the arrays that are to be used and controls such things as bit/word mode, store masked, and shifting. The array control logic includes:

1) Array Select Register
3) Array Mode
4) Shift Control

ARRAY MODE

The array mode logic controls the accessing mode. Either a bit slice, a word slice or a mixed slice is selected for loading or storing. The mode is contained in the high byte of a base register.

ARRAY SELECT REGISTER

The Array Select register in array control establishes those array modules that are to be active for an associative operation. The Array Select register is 32 bits wide. Each bit position controls one array. Bit 0 corresponds to Array 0, and a one in a bit position enables the corresponding array. The Array Select register is loaded and read through the bus logic. The Array Select register contents are also used by the resolver logic.

AS

Array Select register (32 bits)

ASH

Array Select register, high half (bits 0-15)

ASL

Array Select register, low half (bits 16-31)

AS0

Array Select register, byte 0 (bits 0-7)

AS1

Array Select register, byte 1 (bits 8-15)

AS2

Array Select register, byte 2 (bits 16-23)

AS3

Array Select register, byte 3 (bits 24-31)

BIO

Buffered Input/Output

BIT COLUMN

A bit column is a selected bit in every word in an array.

BITS

An array slice consists of 256 bits, all of which may be accessible in parallel or divisible into eight fields of 32 bits each. Addressing a bit position within an array is accomplished by an address between 0, the most-significant (left) bit, and 255, the least-significant (right) bit position. Bit n of all words is accessible from address n.

BL

Block Length counter (16 bits)

BL0

Block Length counter byte 0 (bits 0-7)

BL1

Block Length counter byte 1 (bits 8-15)

BLOCK LENGTH COUNTER

The Block Length counter is a 16-bit decrementing counter. It controls the length of a data block transfer.

BRANCH AND LINK REGISTERS

Eight of the 16 general registers are used to facilitate subroutine linkage. They are R0 through R7.

BUFFERED INPUT/OUTPUT

Buffered I/O (BIO) is available for tying several different types of peripherals into the AP control memory of STARAN. In addition, BIO can be used to transfer blocks of data and/or programs between the AP control memory and the host-computer memory. The basic width of the BIO interface is 32 bits plus a parity bit. The custom I/O cabinet can include buffers that allow a wide variety of repacking to take place so that I/O channels of any width can be assembled. Normally, the EXF channel is used to set up and initiate buffered I/O transfers. Of the several data buses accessible to the High Speed Data Buffer (HSDB), the bulk core memory, and the external memory accessed from a DMA channel, the priority resolver logic gives the BIO port highest priority.

BUS LOGIC

The bus logic provides a common data path for all pertinent registers of AP control and the data bus from AP control memory. The bus is 32 bits wide. Registers of less than 32 bits are grouped to form a 32-bit word.

BYTES

A byte is a portion of a word consisting of 8 consecutive bits.

C

Common register (32 bits)

CC

Condition Code (bits 24-27 of PSW)

CH

Common register, high half (bits 0-15)

CL

Common register, low half (bits 16-31)

C0

Common register, byte 0 (bits 0-7)

C1

Common register byte 1 (bits 8-15)

C2

Common register, byte 2 (bits 16-23)

C3

Common register, byte 3 (bits 24-31)

COMMON REGISTER

The Common register is an AP control register that contains 32 bits numbered 0 to 31. Bit 0 is the left (most-significant) bit. Bit 31 is the right (least-significant bit). The Common register may contain the argument for a search operation performed upon the arrays, the input data stored into an array, or the input data received from an array in a load operation. Data from an array is loaded into the Common register through a mask generated by the mask generator.

COMPARATOR

The comparator is a portion of the program control logic in AP control that compares the address contained in the end loop marker with the Program Counter. The comparator transmits an indication to control when the end of a loop sequence has been reached. Control then loads the start loop register contents into the Program Counter if there are more repetitions to be completed in the loop count.

CONTROL LINE BUFFER

The control line buffer in the response store control controls the timing of the control lines transmitted to the arrays.

CONTROL LINE CONDITIONER

The control line conditioner in the response store control generates the control lines required to manipulate the response store. Control line signals are generated as a function of the instruction register, a selected bit of the Common register, and the inclusive-OR output from the resolver.

DATA POINTER

The Data Pointer in the bus logic of AP control contains the control memory address for the data bus for block transfers. It is a 16-bit register. The Data Pointer can be stepped with each transfer within a data block.

DIRECT MEMORY ACCESS

A block of AP control memory addresses is reserved for the Direct Memory Access (DMA) channel to external memory. In the basic configuration this block can contain up to 30,720 addresses. Direct access to a host-computer memory allows that memory to be shared with AP control memory and a host computer. Items in the memory are equally accessible by a host computer and by STARAN, thus reducing the need for buffered I/O transfers between the host computer and STARAN.

DMA

Direct Memory Access

DP

Data Pointer (16 bits)

DP0

Data Pointer, byte 0 (bits 0-7)

DP1

Data Pointer, byte 1 (bits 8-15)

EFB

External Function Buffer

EFS

External Function Status

ELM

End Loop Marker

END LOOP MARKER

The End Loop Marker is a 16-bit register in the program control logic of AP control. It is used to store the last address of an instruction loop. The End Loop Marker register is loaded from the right-most 16 bits of the loop instruction format.

EXF

External Function

EXTERNAL FUNCTION LOGIC

The external function (EXF) logic facilitates coordination between the different elements of STARAN. By issuing external function codes to the EXF logic, an element of STARAN can control and interrogate the status of other elements. Each function code is 19 bits long and indicates what element of STARAN is to be interrogated and/or controlled. For each function code transmitted to EXF logic, the logic returns a single sense bit indicating the result of the interrogation. Function codes may be transmitted to EXF logic by AP control, the program pager, sequuential control, and a host computer (if the EXF channel to the host computer is implemented).

FIELD LENGTH COUNTERS

Field length counters are 8-bit AP control registers used to contain the length of data fields. They may be decremented to allow stepping through the bits of a data field. When the counter's contents equal zero, an indication is sent to the AP control for test purposes. There are two field length counters: FL1 and FL2.

FIELD POINTER

A field pointer is an 8-bit AP control register that generally contains an array bit column or word address. Field pointers may be incremented or decremented to facilitate stepping through data fields. There are four field pointers: FP1, FP2, FP3, and FPE.

FIELD POINTER 1

Field Pointer 1 is an 8-bit AP control register that may contain an array bit column or word address for the indirect addressing mode. Field Pointer 1 is used by the resolver for the address of the array module containing the first responder.

FIELD POINTER 2

Field Pointer 2 is an 8-bit AP control register that may contain an array bit column or word address. Field pointer 2 is also used by the resolver for the array bit column or word address of the first responder in the array specified in FP1.

FIELD POINTER 3

Field Pointer 3 is an 8-bit AP control register that may contain an array bit column or word address.

FIELD POINTER E

Field Pointer E is an 8-bit AP control register that may contain an array bit column or word address or a shift constant.

FIELDS (SECTIONS)

Each 256-bit word in an array contains eight fields (or sections). Each field contains 32 contiguous bits within the word being addressed. Addressing of a particular field of an array word is accomplished by an address between 0, the most-significant (left) field, and 7, the least-significant (right) field. The most-significant field starts at the most-significant bit position.

FL1

Field Length Counter 1 (8 bits)

FL2

Field Length Counter 2 (8 bits)

FPE

Field Pointer E (8 bits)

FP1

Field Pointer 1 (8 bits)

FP3

Field Pointer 3 (8 bits)

GET

Pager GET address register

GET ADDRESS REGISTER

The GET address register in the program pager holds a 16-bit AP control memory address. If the pager is in the midst of moving data, the GET address points to the memory location holding the next source word to be moved. If the pager is executing instructions, the GET address acts like a program counter that points to the location of the next pager instruction.

GRP

Group register

HIGH SPEED DATA BUFFER

The High Speed Data Buffer (HSDB) is a section of AP control memory using fast bipolar solid-state elements. In the basic configuration of STARAN it contains 512 words. All buses accessible to AP control memory can gain access to the HSDB, making it a convenient place to store data and instruction items that need to be accessed quickly by different elements of STARAN.

HOST COMPUTER

A conventional sequential-type computer integrated with STARAN to provide efficient processing for sequential operations. (STARAN performs tasks requiring parallel operations.)

HSDB

High Speed Data Buffer

IMASK

Interrupt Mask (bits 28-31 of PSW)

INSTRUCTION REGISTER

The instruction register in AP control contains the current instruction being executed. The instruction loaded into the instruction register is received from AP control memory through the instruction bus. Parity is checked at the instruction register. The instruction register contains 32 bits, which are numbered from 0 to 31, with bit 0 at the left. Portions of the instruction register are used as a direct source of immediate data or addresses as a function of the instruction being executed.

INTERLOCKS

The EXF logic contains 64 stored bits called interlocks. These bits have no predetermined meaning. Software may assign a meaning to an interlock and use it for any purpose. Function codes allow the current state of an interlock to be sensed and a new state to be entered in one operation. Sixteen interlocks (hex addresses 00 through 0F) can be controlled and sensed manually by panel switches and lights to facilitate communication with an operator. The other 48 interlocks (hex addresses 10 through 3F) can be sensed and controlled only by function codes.

INTERRUPT MASK

The Program Status Word in the program control logic contains the interrupt mask for the 15 AP control interrupts. All interrupts with numbers greater than the mask are accepted. The Interrupt Mask is contained in bits 28 through 31 of the Program Status Word.

I/O

Input/Output

LEAST-SIGNIFICANT BIT

A significant bit is a bit that contributes to the precision of a numeric value. The number of significant bits is counted beginning with the bit contributing the most value, called the most-significant bit, and ending with the one contributing the least value, called the least-significant bit. In STARAN memory the least-significant bit is bit 31 in a 32-bit word or field or bit 255 in a 256-bit array word.

LP

Link Pointer (FP1 and FP2 linked)

LSB

Least Significant Bit (bit 31 of 32 bit word); right most bit; low order bit

LSF

Least-Significant Field

M

M Array register; Mask (256 bits)

M REGISTER

The M register (MASK) is a 256-bit register contained in the response store element of each array. Its special use is to select array words participating in an associative operation.

MAIN MEMORY

The main memory is a section of AP control memory using nonvolatile core storage. In the basic configuration it contains 16,384 words and is expandable to 32,768 words. Like the High Speed Data Buffer, main memory is accessible to all buses that can gain access to AP control memory (a priority port switch giving each memory cycle to the highest priority bus requesting a main memory address). The priority of the buses is the same as for the High Speed Data Buffer. Since it is large and nonvolatile, the main memory is useful for storing the AP control programs. Because it is slower than the page memories, it is recommended that program segments be paged into the page memories for execution. Since it is accessible by all buses accessible to AP control memory, it is also useful as a buffer for data items that do not require the higher speed of the HSDB.

MASK

M Array register

MASK GENERATOR

The mask generator in AP control generates a mask pattern to be used in loading array output data into the Common register. The mask enables data to be loaded for a number of contiguous bits. The mask generator requires the bit addresses of the most and least-significant bits of the field to be loaded. All bits between and including these limits are loaded while those outside these limits are unaltered.

MDA

Multi-Dimensional Access memory; associative array

MIRRORING

Mirroring will cause the 256-bit input quantity to an associative instruction to be flipped end-for -end (i.e., bit i is put into bit 255-i).

MOST-SIGNIFICANT BIT

A significant bit is a bit that contributes to the precision of a numeric value. The number of significant bits is counted beginning with the bit contributing the most value, called the most-significant bit, and ending with the bit contributing the least value, called the least-significant bit. In STARAN memory the most-significant bit is bit 0.

MSB

Most Significant Bit (bit 0 of word); left-most bit; high order bit

MSF

Most-Significant Field

OR

Logical inclusive-OR

PAGE MEMORY

Three page memories are included in the AP control memory: Page 0, Page 1, and Page 2. Each page memory uses fast bipolar solid-state elements and is volatile. Each page contains 4096 words in the basic STARAN-E configuration.

PAGE PORT SWITCHES

Each page memory has a port switch that connects the memory to the AP control instruction bus, the pager bus, or the sequential control bus. A page port switch function code is used for interrogation and control of these switches.

PAGE0

High speed solid state memory; 512 32-bit words

PAGE1

High speed solid state memory; 512 32-bit words

PAGE2

High speed solid state memory; 512 32-bit words

PARALLEL INPUT/OUTPUT

Each associative array in STARAN can have up to 256 inputs and 256 outputs into the custom I/O cabinet. The basic width of the PIO is 256n, where n is equal to the number of associative arrays in the system (n can have a maximum value of 8). The custom I/O cabinet is capable of buffering and reformatting the data received from any peripheral device to match the width necessary to communicate with the STARAN associative array. In order to synchronize read and write operations with an external device operating through PIO , certain AP control instructions wait for sync pulses from the external device. As in inter-array communication, an external function code sent to the custom I/O will set up and initiate the PIO operation.

PC

Program Counter (16 bits)

PC0

Program Counter (bits 0-7)

PC1

Program Counter (bits 8-15)

PIO

Parallel Input/Output

PROGRAM COUNTER

The Program Counter occupies bits 0-15 of the Program Status Word in AP control. The Program Counter contains the address of the current instruction being executed. It is normally incremented sequentially through control memory. Its normal sequence may be altered by a branch or loop instruction.

PROGRAM PAGER

The program pager loads the high-speed page memories from the bulk core portion of AP control memory. The pager performs these transfers independent of AP control, so that while AP control is executing a program segment out of one page memory, the pager can be loading another page memory with a future program segment. Pager operation is initiated by external function codes. The program pager contains three registers: a GET address register, a PUT address register, and a word count register. The GET address register holds a 16-bit AP control memory address. If the pager is in the midst of moving data, the GET address register points to the memory location holding the next source word to be moved. When executing instructions, the GET address register acts like a program counter, pointing to the location of the next page memory instruction. The PUT address register contains a 16-bit AP control address. It points to the memory location into which the next destination word is to be put during a move-data operation. The word count register, which is 14 bits in length, contains the number of words still to be transferred during a transfer operation.

PROGRAM STATUS WORD

The Program Status Word (PSW) consists of the Program Counter (PC) (bits 0-15) which contains the address of the current AP control instruction being executed, the Condition Code (CC) (bits 24-27) and the Interrupt Mask (IMASK) (bits 28-31) which contains the current interrupt status. (All interrupts with numbers greater than the IMASK are accepted.)

PSW

Program Status Word

PUT

Pager PUT Address register

PUT ADDRESS REGISTER

The PUT address register in the program pager holds a 16 bit page memory address. It points to the memory location into which the next destination word is to be put during a move-data operation.

R0

General register, index

R1

General register, index

R2

General register, index

R3

General register, index

R4

General register, index

R5

General register, index

R6

General register, index

R7

General register, index

R8

General register, stack base

R9

General register, index

RA

General register, index

RB

General register, direct mode base

RC

General register, FP1 base

RD

General register, FP2 base

RE

General register, FP3 base

REGISTER

A register is a memory device capable of containing one or more bits or words.

RESOLVER

The resolver logic in AP control finds the array address and word address of the first (most-significant) responder. The array address is loaded into Field Pointer 1 and the word address is loaded into Field Pointer 2. This allows subsequent operations to only affect the first responder of a search.

RESPONDER

A responder is a response store element in an enabled array whose Y register bit is set. Generally, responders indicate those words satisfying some search criteria.

RESPONSE STORE CONTROL

The response store control logic generates the control signals required by the MDA arrays and buffers them to insure correct timing at the response store. The response store control consists of two basic portions: 1) the control line conditioner and 2) the control line buffer.

RESPONSE STORE ELEMENT

The response store portion of the array memory consists of 256 response store elements, each containing an X, Y, and M register bit. The 256 X bits are considered the X register; the 256 Y bits are considered the Y register; the 256 M bits are considered the M register or the MASK.

RF

General register, Link Pointer base

SC

Sequential Controller

SEQUENTIAL CONTROL

The sequential control block of STARAN contains a sequential processor (SP) with 16K of memory, a keyboard/monitor, a disk drive, and interface logic to connect the SP to other STARAN elements.

SEQUENTIAL CONTROL INTERRUPTS

Sequential control can accept interrupts from other STARAN elements. The interrupts arise from error detection, the panel interrupt button, or external functions. Eight different interrupt vector addresses are provided.

SEQUENTIAL PROCESSOR

The sequential processor is a 16-bit, general-purpose, parallel-logic computer using two's complement arithmetic for addressing 16,384 16-bit words (32,768 8-bit bytes) of memory. All communication between system components is done on a single high-speed bus. There are eight general purpose registers, which can be used as accumlators, index registers, or address pointers, and a multi-level automatic priority interrupt system.

SHIFT CONTROL

The shift control logic in array control generates the controls signals required by the array to perform shifting and mirroring operations.

SHIFT LOGIC

Data transmitted from the bus logic of AP control passes through the bus shift logic. The bus shift logic is used to shift the 32-bit bus word left-end around by either 0-, 8-, 16-, or 24-bit positions. The shift is controlled by the instruction moving the data. Data received from AP control memory is checked for correct parity as it passes the bus shift logic. Data being stored in the control memory has a parity bit generated at the bus shift logic.

SLM

Start Loop Marker

SP

Sequential Processor

START LOOP MARKER

The start loop marker is a 16-bit register in the program control logic of AP control. The start loop marker is used to store the first address of the loop whenever a loop instruction is executed. The start loop marker is loaded directly from the program counter at the start of an instruction loop. It is read into the program counter whenever the last instruction of the loop is executed and the loop is to be repeated.

WORD COUNT REGISTER

The word count register is a 14-bit register in the program pager which contains the number of words still to be transferred during a move-data operation.

WORDS

An array word consists of 256 bits, all of which may be accessed in parallel or via fields of 32 bits each. Addressing of a word within an array is accomplished by using an address from 0, the first word, to 255, the last word. An AP control memory word is 32 bits in length. A sequential control word is 16 bits in length.

X

X Array register (256 bits)

X REGISTER

The X register is a 256-bit register contained in the response store element of each array. It may be used as temporary storage of data loaded from the array or stored in the array. It can be combined logically with the input data and/or the Y register. It is useful as temporary storage in parallel arithmetic operations or searches.

XOR

Logical exclusive-OR

Y

Y Array register (256 bits)

Y REGISTER

The Y register is a 256-bit register contained in the response store element of each array. It may be used as temporary storage of data loaded from the array or stored in the array. It can be combined logically with the input data. It is useful as temporary storage in parallel arithmetic operations and searches. It is also used as the responder in a resolve operation.

APPENDIX B

INSTRUCTION SUMMARY IN HEX CODE ORDER
(APPLE MNEMONICS ARE IN PARANTHESES)

OPCODE	FUNCTION PERFORMED
LOAD MDA REGISTERS (L, LN, LOR, LORN, LAND, LANDN, LXOR, LXORN)	
00	Load unflipped from MDA array by direct address
01	Load unflipped from MDA array by field pointer
02	Load flipped from MDA array by direct address
03	Load flipped from MDA array by field pointer
04	Load unflipped from MDA stack by direct address
05	Load unflipped from MDA stack by field pointer
06	Load flipped from MDA stack by direct address
07	Load flipped from MDA stack by field pointer
08	Load Mask unflipped from MDA array by direct address
09	Load Mask unflipped from MDA array by field pointer
0A	Load Mask flipped from MDA array by direct address
0B	Load Mask flipped from MDA array by field pointer
0C	Load Mask unflipped from MDA stack by direct address
0D	Load Mask unflipped from MDA stack by field pointer
0E	Load Mask flipped from MDA stack by direct address
0F	Load Mask flipped from MDA stack by field pointer
STORE MASKED TO MDA ARRAY (SM, SNM, SORM, SORNM, SANDM, SANDNM, SC, SCW)	
10	Store through Mask unflipped to MDA array by direct address
11	Store through Mask unflipped to MDA array by field pointer
12	Store through Mask flipped to MDA array by direct address
13	Store through Mask flipped to MDA array by field pointer
14	Store through Mask unflipped to MDA array by direct address
15	Store through Mask unflipped to MDA array by field pointer
16	Store through Mask flipped to MDA stack by direct address
17	Store through Mask flipped to MDA stack by field pointer
STORE TO MDA ARRAY (S, SN, SOR, SORN, SAND, SANDN)	
18	Store unflipped to MDA array by direct address
19	Store unflipped to MDA array by field pointer
1A	Store flipped to MDA array by direct address
1B	Store flipped to MDA array by field pointer
1C	Store unflipped to MDA array by direct address
1D	Store unflipped to MDA array by field pointer
1E	Store flipped to MDA array by direct address
1F	Store flipped to MDA array by field pointer

LOAD COMMON REGISTER
(LC, LCM, LCW)

20	Load Common unflipped from MDA array by address
21	Load Common unflipped from MDA array by field pointer
22	Load Common flipped from MDA array by addresss
23	Load Common flipped from MDA array by field pointer
24	Load Common unflipped from MDA array by address
25	Load Common unflipped from MDA array by field pointer
26	Load Common flipped from MDA array by address
27	Load Common flipped from MDA array by field pointer

BRANCH INSTRUCTIONS
(B, BAL, BBS, BBZ, BNOV, BNR, BNZ, BOV, BRS, BZ)

28	Branch on array control register zero
29	Branch on array control register non-zero
2C	Branch and link

AP CONTROL REGISTER STORE INSTRUCTIONS
(LRR, SPSW, SR)

30-31	Store AP control register to AP control register or memory

AP CONTROL REGISTER LOAD INSTRUCTIONS
(LI, LR, LPSW)

32	Load low half of control register from memory or with immediate value
33	Load low half of field pointer from memory or with immediate value
34	Load high half of control register from memory or with immediate value
35	Load high half of field pointer from memory or with immediate value
36	Load control register from memory or with immediate value
37	Load field pointer from memory or with immediate value

EXTERNAL FUNCTION INSTRUCTIONS

38	Issue external function

LOOP INSTRUCTIONS
(LOOP, RPT)

3C	Repeat instruction
3D	Loop
3E	Load FP1 and repeat instruction
3F	Load FP1 and loop

SPEED-UP MODE

40-7F	Same as 00-3F with early fetch (speed-up) option enabled

GENERAL REGISTER INSTRUCTION

80	Store general register to memory (ST)
81	Load general register from memory (LD)
82	Load general register halfword from memory (LDH)
83	Load general register byte from memory (LDB)
A000	Move general register (or memory) to general register (or memory) (MOV)
A001	Move general register (or memory) to general register (or memory) halfword (MOVH)
A002	Move general register (or memory) to general register (or memory) byte (MOVB)
A00B	Move complement general register (or memory) upper byte to general register (or memory) (MCUB)
A080	Store AP control register to general register (or memory) (SAC)
A084	Load general register (or memory) to AP control register (LAC)

CALL SUBROUTINE INSTRUCTION

A3	Subroutine link (CAL)

APPENDIX C

SUMMARY OF EXTERNAL FUNCTION CODES

EXF CODE (BITS 13-31)	FUNCTION PERFORMED
PAGE PORT SWITCHES	
002A0	Page 0 to instruction bus
006A0	Page 1 to instruction bus
00AA0	Page 2 to instruction bus
003F0	Page 0 to pager bus
007F0	Page 1 to pager bus
00BF0	Page 2 to pager bus
00150	Page 0 to Sequential controller
00550	Page 1 to Sequential controller
00950	Page 2 to Sequential controller
001B2	Page 0 on instruction bus?
005B2	Page 1 on instruction bus?
009B2	Page 2 on instruction bus?
001B1	Page 0 on pager bus?
005B1	Page 1 on pager bus?
009B1	Page 2 on pager bus?
001B4	Page 0 to Sequential controller?
005B4	Page 1 to Sequential controller?
009B4	Page 2 to Sequential controller?
INTERLOCKS (ILOCK)	
06nn0	Reset interlock nn
06nn5	Is interlock nn set?
06nn6	Is interlock nn reset?
06nnC	Set interlock nn
PAGER CONTROL (PAGER)	
08000	Stop pager
0800C	Resume pager
08005	Pager busy?
08006	Pager non busy?
4nnnn	Start pager at address 'nnnn'

AP INTERRUPTS
(INT)

100n0	Reset AP interrupt n
100n5	Is AP interrupt n set?
100n6	Is AP interrupt n reset?
100nC	Set AP interrupt n

AP CONTROL ACTIVITY

0200C	Start AP at location 0
02000	Stop AP execution (WAIT)
02005	Is AP active?
02006	Is AP inactive?
040EC	Place AP in 'single step' mode
040E0	Remove AP from 'single step' mode
022F0	Resume 'single step' mode
040FC	Set AP trap
040F0	Reset AP trap
040F5	Is AP trap set?
040F6	Is AP trap reset?
02011	Is AP loop indicator set?
02012	Is AP loop indicator reset?
02014	Reset AP loop indicator

ERROR CONTROL

040n5	Is error n set?
040n6	Is error n reset?
040n0	Reset error n
040nC	Set error n
040DC	Set error interrupt enable
040D0	Reset error interrupt enable
040D5	Is error interrupt enable set?
040D6	Is error interrupt enable reset?
040AC	Set lockout control (2)
040A0	Reset lockout control
040A5	Is lockout control set?
040A6	Is lockout control reset?

NOTES: (1) Errors 0, 1, 3, 4, 7, 8, or 9 will force the AP activity bit into the reset state.

(2) No other EXF's may be issued by these devices until this lockout is reset by the Sequential controller.

CLEARS AND INTERFACE CONTROLS

02020	Clears STARAN command channel devices
02030	Clears errors, Sequential interrupts
02040	Clears AP interrupts
02050	Clears DMA control
02060	Clears bulk core control
02070	Clears next instruction
02080	Clears branch instruction
02090	Clears register instruction
020A0	Clears pager control
020B0	Clears HSDB control
020C0	Clears Sequential controller interface
020D0	Clears associative instruction
020E0	Clears Page 0 control
020F0	Clears Page 1 control
02100	Clears Page 2 control
02110	Clears Register Processor Unit
02120	Scope trigger
02130	Unassigned (1)
02140	Unassigned (2)
02150	Unassigned (2)
02160	Unassigned (2)
02170	Unassigned (2)
02180	Write status for RADC display interface
02190	Read control word for RADC display
021A0	RADC display interface
021B0	Clears PHD errors
021C0	Clears PHD control
021D0	Clears CIOU Sigma-9 control
021E0	Sets HIS 645 single step
021F0	Clears HIS 645 single step
02200	Clears HIS 645 interface control
02210	Clears HIS 645 interface pointers
02220	Unassigned (3)
02230	Clears CIOU control memory controller
02240	Clears CIOU next instruction
02250	Clears CIOU register instruction
02260	Clears CIOU branch instruction
02270	Clears CIOU associative instruction
02280	Clears CIOU array assignment
02290	Clears CIOU BIO channel interface
022A0	Clears CIOU EXF fan-in
022B0	Clears CIOU EXF decode
022C0	Clears CIOU performance monitor
022D0	Unassigned (4)

022E0	Issue CIOU Resume
022F0	Issue AP Resume
02300	Master Clear

NOTES: (1) can cause unexpected results in AP execution

(2) may interfere with another's use of this memory

(3) may cause execution faults in issuing device

(4) interference with host interface

INDEX