

**Project Report  
ATC-87  
Volume 1**

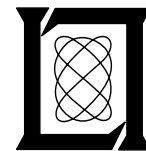
# **The Aircraft Reply and Interference Environment Simulator (ARIES) Volume 1: Principles of Operation**

**M. Goon  
D. A. Spencer**

**22 March 1979**

---

**Lincoln Laboratory**  
MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
*LEXINGTON, MASSACHUSETTS*



Prepared for the Federal Aviation Administration,  
Washington, D.C. 20591

This document is available to the public through  
the National Technical Information Service,  
Springfield, VA 22161

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.



## CONTENTS

	<u>Page</u>
1.0 INTRODUCTION	1
1.1 Introduction	1
1.2 DABS Summary	3
1.2.1 ATCRBS Modes	3
1.2.2 DABS Modes	5
1.2.3 Channel Time Allocation	8
1.2.4 Azimuth Determination	8
1.2.5 Back-to-Back Antenna Operation	11
1.2.6 Primary Radar Data	11
1.2.7 The DABS Network	11
1.3 ARIES System Summary	12
1.3.1 Capabilities	12
1.3.2 Simulated Transponders	13
1.3.2.1 Interrogation/Reply Cycle	13
1.3.2.2 Interrogation and Reply Types Handled by ARIES	13
1.3.3 Radar Report Generation	16
1.3.4 Fruit Reply Generation	16
1.3.5 The Traffic Model	18
1.3.6 Multi-Site Operation	19
1.3.7 Data Recording	20
1.3.8 Self Test Capabilities	20
2.0 DETAILED ARCHITECTURE AND OPERATION	22
2.1 Overall Architecture	22
2.1.1 I/O Interface Conventions	26
2.2 Timing	28
2.2.1 ARIES Clock	28
2.2.2 Universal Interval Timer (UIT)	30
2.2.2.1 Universal Interval Timer Interface	33
2.3 Receiver	35
2.3.1 Analog Circuits	35
2.3.2 Digital Circuits	35
2.3.2.1 Mode Decoding	35
2.3.2.2 Parity Decoder	41
2.3.2.3 Data Transfer Sequence	43
2.3.3 Receiver Interface	45
2.3.3.1 Data Channel Control Logic	47
2.3.3.2 Memory Address Counter	50
2.3.3.3 Register File and Control	50
2.3.3.4 Interrogation Counter	52
2.3.3.5 Interrupt Logic	54



## CONTENTS (Cont'd)

	<u>Page</u>
2.4 Reply and Fruit Generation	54
2.4.1 Controlled Reply Generator (CRG)	56
2.4.1.1 Controlled Reply Controller (CRC)	56
2.4.1.1.1 Architecture	56
2.4.1.1.2 Program Control Logic	58
2.4.1.1.3 Branch Control Logic	62
2.4.1.1.4 Microprocessor	67
2.4.1.1.5 Input Circuits	73
2.4.1.1.6 Output Circuits	75
2.4.1.2 Controlled ARIES Targets (CAT's)	75
2.4.1.2.1 Reply Generator Data Bus Structure	78
2.4.1.2.2 Major Elements of the Reply Generator	81
2.4.1.2.3 Operation of the Reply Generator Elements	81
2.4.1.2.3.1 Input Buffer and Register	81
2.4.1.2.3.2 Delay-To-Trigger Timing Circuit	91
2.4.1.2.3.3 Reply Assembler	92
2.4.1.2.4 Reply Simulation	93
2.4.1.2.4.1 DABS Replies	93
2.4.1.2.4.2 ATCRBS Replies	98
2.4.1.3 Controlled Reply Generator Interface	102
2.4.1.3.1 Memory and Memory Address Counter	105
2.4.1.3.2 Memory Write Control Logic	105
2.4.1.3.3 Memory Read Control Logic	107
2.4.1.3.4 Reply Counters	109
2.4.1.3.5 Reset Bit Register	112
2.4.1.3.6 Control Bits	112
2.4.2 Fruit Generator (FG)	112
2.4.2.1 Fruit Generator Controller (FGC)	113
2.4.2.2 Fruit ARIES Targets (FAT'S)	113
2.4.2.3 Random Process Generator	113
2.4.2.3.1 General Description	113
2.4.2.3.2 Random Number Generators	116
2.4.2.3.3 Generation of Random Range (Power) Control Parameter	119
2.4.2.3.3.1 Generation of Mainbeam/Sidelobe Control Parameter	120

## CONTENTS (Cont'd)

	<u>Page</u>
2.4.2.3.4 Generation of Monopulse Angle Control Parameter	120
2.4.2.3.4.1 Generation of Monopulse Angle Sign Control Parameter	120
2.4.2.3.5 Generation of Random ATCRBS Codes	120
2.4.2.3.6 Generation of the Delay-to-Trigger Control Parameter	123
2.4.2.3.7 Reading the RPG	127
2.4.2.4 Fruit Generator Interface	129
2.4.2.4.1 Mode Decoding Logic	129
2.4.2.4.1.1 Normal Mode	132
2.4.2.4.1.2 RPG Diagnostic Mode	135
2.4.2.4.1.3 FAT Diagnostic Mode	135
2.4.2.4.1.4 Loop Test Mode	135
2.5 IF Circuits	136
2.5.1 Controlled Reply and Fruit IF Circuits	136
2.5.2 IF Combiner	136
2.6 ACP Decoder	147
2.6.1 Sensor Mode	147
2.6.2 ARIES Mode	147
2.6.2.1 Programmable ACP Counter and Correction Counter	150
2.6.3 Output Circuits	150
2.7 Status Formatter	152
2.7.1 Operation of the Formatter	152
2.7.2 Status Formatter Polling Cycle	155
2.7.3 Formatter Self-Diagnostic Mode	155
2.7.4 Status Formatter Interface	155
2.8 Self Test Unit	160
2.8.1 Modes of Operation	160
2.8.2 Interrogation Generator	160
2.8.3 Reply Sampling	167
2.8.4 Data Register Controller	172
2.8.5 STU Controller	175
2.8.6 STU Interface	176
2.9 Radar Report Interface	179
2.10 Random Number Generator Interface	181

## CONTENTS (Cont'd)

	<u>Page</u>
3.0 OPERATIONAL SOFTWARE	186
3.1 Major Software Functions	186
3.2 Interrogation Processing	189
3.2.1 Assumed Interrogation Pattern	189
3.2.2 Interrogation Processing Overview	189
3.2.3 Discrete Interrogation Processing	192
3.2.3.1 Inputs, Outputs, and Timing	192
3.2.3.2 Location of the Track File Record	192
3.2.3.3 Determining Whether the Target is to Reply	202
3.2.3.4 Reply Time, Power, and Monopulse Calculations	204
3.2.3.5 Interrogation Data Bit Processing	205
3.2.3.6 Reply Data Generation	207
3.2.3.7 Timing Considerations for Discrete Interrogations	208
3.2.4 ATCRBS/All-Call Reply Generation	211
3.2.4.1 Inputs, Outputs, and Timing	211
3.2.4.2 Processing	211
3.2.5 ATCRBS/All-Call Interrogation Processing	213
3.2.5.1 Inputs, Outputs, and Timing	213
3.2.5.2 Acquiring the All-Call Interrogation Pattern	213
3.2.5.3 ATCRBS/All-Call Pattern Verification	214
3.2.5.4 Initializing the Interval Timer	214
3.2.5.5 Update to the Antenna Azimuth and Rate File	215
3.2.5.6 Update of the Fruit Rate	215
3.2.5.7 Transmission of Radar Data	215
3.3 Pre-interrogation Processing	215
3.3.1 Purpose	215
3.3.2 Inputs, Outputs, and Timing	216
3.3.3 Locating Tracks to be Processed	216
3.3.4 Update to the Track File	219
3.3.5 Preparing the Reply Time Sorted Index	221
3.3.6 Radar Report Generation	221
3.4 Traffic Model Input	224
3.4.1 Purpose, Input/Output, and Timing	224
3.4.2 Input Processing	224
3.4.3 Starting a New Track	228
3.4.4 Update to an Existing Track	229
3.4.5 Deleting a Track	230

## CONTENTS (Cont'd)

	<u>Page</u>
3.5 Inter-ARIES Communication	231
3.5.1 Purpose	231
3.5.2 Message Formats and Line Protocols	232
3.5.3 Overview of the All-Call Lockout Protocol	233
3.5.4 Overview of Acknowledgment Request (AR) Processing	234
3.5.5 Overview of Downlink Request Processing	237
3.5.6 Overview of the Transponder Timeout Protocol	238
3.5.7 New Track Initialization	238
3.5.8 System Synchronization	239
3.5.9 Adjacent ARIES Input Task	241
3.6 Track File Timer Countdown	241
3.7 Operator Communications	245
3.7.1 Starting the Simulation	245
3.7.2 Stopping the Simulation	246
3.7.3 Reinitializing to a New Starting Time	246
3.7.4 System Status Command	246
3.7.5 Starting a Track	247
3.7.6 Dropping a Track	247
3.7.7 Changing the Recording Mode	247
3.7.8 Enabling/Disabling Interrogation Data Recording	248
3.8 System Initialization	248
3.9 Diagnostic Data Recording	248
3.10 System Tasks and Task Scheduling	254
REFERENCES	256

## LIST OF ILLUSTRATIONS

<u>Figure No.</u>		<u>Page</u>
1.1-1	ARIES Equipment, Overall View	2
1.2-1	ATCRBS/DABS All-Call Interrogations	4
1.2-2	ATCRBS Reply Format	4
1.2-3	DABS Interrogation	4
1.2-4	DABS Reply Format	6
1.2-5	DABS Data Block Format	7
1.2-6	DABS Roll-Call Scheduling	9
1.2-7	Sum and Difference Antenna Patterns	10
1.3-1	Overall System Diagram	14
2.1-1	ARIES Block Diagram	23
2.1-2	Digital Subsystem, Right View	24
2.2-1	ARIES System Clock, Block Diagram and Waveforms	29
2.2-2	Universal Interval Timer, Block Diagram	31
2.2-3	Universal Interval Timer Interface, Block Diagram	34
2.3-1	Analog Drawer (Top View) Showing Receiver Components	36
2.3-2	Analog Receiver, Block Diagram	38
2.3-3	Uplink Receiver Analog and Digital Circuits, Block Diagram	39
2.3-4	Mode Decoder, Block Diagram	40
2.3-5	Mode Decoder Waveforms	42
2.3-6	Parity Decoder, Block Diagram	44
2.3-7	Data Transfer State Diagram	46
2.3-8	Receiver Interface, Block Diagram	48
2.3-9	Data Channel Control Network	49

List of Illustrations (Continued)

<u>Figure No.</u>		<u>Page</u>
2.3-10	Memory Address Counter, Block Diagram	51
2.3-11	Register File and Control Circuits, Block Diagram	53
2.3-12	Modified Data General Interrupt Logic, Block Diagram	55
2.4-1	Controlled Reply Controller, Block Diagram	57
2.4-2	Microprogram Storage and Pipeline Register	59
2.4-3	Block Diagram of a 2909	60
2.4-4	Program Control Logic	61
2.4-5	Branch Control Logic	63
2.4-6	Condition Code Multiplexer	65
2.4-7	Microprocessor Slice, Block Diagram	68
2.4-8	16-Bit Microprocessor, Block Diagram	69
2.4-9	Input Circuits	74
2.4-10	Output Circuits	76
2.4-11	Data Bus Interconnecting the Controller and Its Target Generators	79
2.4-12	Reply Generator, Block Diagram	82
2.4-13	Reply Generator, Detailed Block Diagram	83
2.4-14	State Diagram for Controller No. 1	85
2.4-15	State Diagram for Controller No. 2	86
2.4-16	Memory Write Logic	88
2.4-17	Waveforms for the Memory Write Logic During LOAD 1-4 Sequence	89
2.4-18	Controller No. 3	94
2.4-19	Sync Logic	96

List of Illustrations (Continued)

<u>Figure No.</u>		<u>Page</u>
2.4-20	Control Signal Waveforms, DABS Reply (Long)	97
2.4-21	PPM Modulator	99
2.4-22	Parity Encoder	100
2.4-23	ATCRBS Reply	98
2.4-24	PAM Modulator	103
2.4-25	CRG Controller Interface	104
2.4-26	Memory Write Control Logic	106
2.4-27	Memory Writing Cycle Waveforms	108
2.4-28	Memory Read Control Waveforms	110
2.4-29	Reply Counter Control Logic	111
2.4-30	Daisy-Chaining of Three Fruit ARIES Targets (FAT's)	114
2.4-31	RPG Block Diagram	115
2.4-32	7-Bit Pseudo Random Number Generator (RNG1, RNG2)	117
2.4-33	Resetting Circuit and Waveforms	118
2.4-34	ATCRBS Code, Range, Monopulse, LR and M/S Generators	121
2.4-35	Poisson Sequence Generator	124
2.4-36	Poisson Sequence Generator State Diagram	126
2.4-37	Read Control Logic and Timing Diagram	128
2.4-38	Fruit Generator Interface	130
2.4-39	Output Channel	131
2.4-40	Mode Decoding Logic	133
2.4-41	Read/Write Counters	134
2.5-1	Reply Generator (Side No. 1)	137

List of Illustrations (Continued)

<u>Figure No.</u>		<u>Page</u>
2.5-2	Reply Generator (Side No. 2)	139
2.5-3A	IF Unit, Block Diagram	141
2.5-3B	IF Unit, Block Diagram	142
2.5-4	IF Combiner	143
2.5-5	IF Combiner (Left View)	145
2.6-1	Digital SubSystem Sub Panel (Front View)	148
2.6-2	ACP Decoder, Block Diagram	149
2.6-3	ACP and Correction Counters, Block Diagram	151
2.7-1	Status Formatter, Block Diagram	153
2.7-2	Status Report Logic	154
2.7-3	Status Formatter Control Sequence	156
2.7-4	Status Formatter Interface	159
2.8-1	STU Block Diagram	161
2.8-2	Interrogation Generator (Digital Section)	163
2.8-3	Interrogation Generator (RF Section)	165
2.8-4	Comm-A Interrogation Waveform	166
2.8-5	Reply Sampling Waveforms	168
2.8-6	Simplified Diagram of VPQ	169
2.8-7	VPQ Waveforms	170
2.8-8	Video Digitizer	171
2.8-9	Data Register Controller	173
2.8-10	Sample and Load Command for DABS Replies	174
2.8-11	State Diagram of STU Controller	177
2.8-12	Control Registers	178
2.8-13	11-Word Buffer	180



List of Illustrations (Continued)

<u>Figure No.</u>		<u>Page</u>
2.9-1	Memory Address and Word Count Registers	182
2.9-2	Radar Report Interface	183
2.10-1	Random Number Generator Interface	184
3.1-1	ARIES Software System	187
3.2-1	Channel Time Allocation	190
3.2-2	Interrogation Processing Flowchart	193
3.2-3	Derivation of Worst Case Timing for Processing Discrete Interrogations	209
3.3-1	Pre-interrogation Processing	217
3.3-2	Merging Reply Time Sorted Lists	222
3.4-1	Traffic Model Input	225
3.5-1	Pilot Response Protocol	235
3.5-2	Adjacent ARIES Input	242
3.6-1	Timer Countdown	244
3.8-1	System Initialization	249
3.9-1	Data Block Format	253
3.9-2	Recording Buffer Format	253

## 1.0 INTRODUCTION AND SYSTEM OVERVIEW

### 1.1 Introduction

The Aircraft Reply and Interference Environment Simulator (ARIES) is designed to simulate a radar beacon environment of up to 400 transponders plus high rates of interfering beacon replies ("fruit") for the purpose of testing the operation of beacon interrogators under heavy load. In particular, ARIES is designed to test a new class of interrogators being developed for the Federal Aviation Administration as part of the Discrete Address Beacon System (DABS). In this document these will be called "sensors".

DABS provides increased capacity, better azimuth measurement precision, and reduced interference between sensors due to reduced interrogation rates as compared with the current beacon system (the Air Traffic Control Radar Beacon System-ATCRBS). In addition, for aircraft equipped with DABS transponders, the system provides ground-to-air and air-to-ground data transmission capabilities which are the basis for future automation of various air traffic control functions.

Due to the large target capacity of a DABS sensor, it is not possible to find a current air traffic environment that is dense enough to fully test it under heavy load conditions. (DABS has been designed for the anticipated worst case environments of the 1980's and beyond). Furthermore, there currently exist only a few DABS transponders, and so it is not possible to provide any significant loading of the DABS-specific functions. The complexity of a DABS sensor precludes simply extrapolating performance from a less dense benchmark test. For this reason, and also due to a desire to be able to repeat the identical test several times, an environment simulator has been built which appears to the sensor under test to be a dense beacon environment plus a dense interference environment typical of what might be encountered in the future. ARIES is intended to provide this environment for purposes of factory and field acceptance testing, and also during the test and evaluation phase of the DABS development program. It can also provide a long term capability of being able to recreate various scenarios for purposes of continued sensor software development and debugging.

Note that ARIES is intended primarily to provide a load test for the DABS sensor. ARIES is not designed to be able to test all the system features of a DABS sensor. Capabilities which are costly to implement and have little effect on the sensor's behavior under load have not been implemented. Those capabilities which have been implemented can, of course, be used for general system testing whether heavily loaded or not. A large portion of the DABS sensor can be tested in this fashion.

The ARIES equipment consists of interrogation receiving circuitry, reply generation circuitry, and a computer and associated peripheral equipment to control the system. This equipment is housed in two standard racks as shown in Fig. 1.1-1.

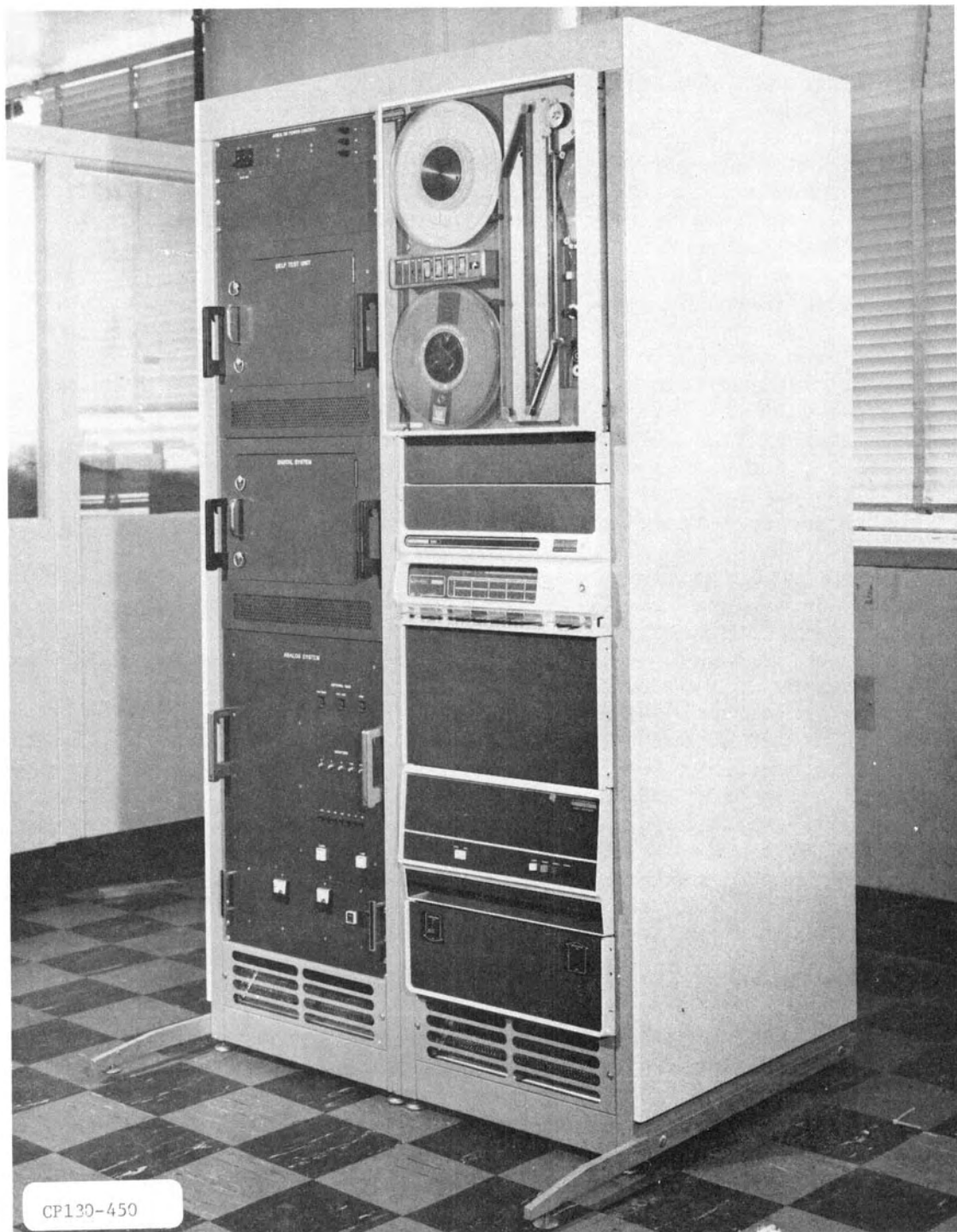


Fig. 1.1-1. ARIES equipment, overall view.

In the rest of this section an overview of the ARIES system is presented. While it is not the purpose of this document to describe the DABS system, those features required for an understanding of ARIES are presented\*. Subsequent sections of this report go into much greater detail about the operation of ARIES. Section 2.0 deals with the operation of the ARIES hardware, and Section 3.0 with the real time computer programs. These sections provide enough background information so that a person experienced in digital or analog hardware design, or in software design, should be able to go directly to hardware drawings or program listings for more information.

## 1.2 DABS Summary

### 1.2.1 ATCRBS Modes

A DABS sensor must be able to support the current Air Traffic Control Radar Beacon System (ATCRBS) in order to permit an orderly transition from it to the Discrete Address Beacon System. Therefore, the sensor looks like a standard ATCRBS sensor to all aircraft equipped with ATCRBS transponders.

Standard ATCRBS interrogations consist of three pulses (P1, P2, P3) spaced as shown in Fig. 1.2-1. ATCRBS interrogations transmitted by a DABS sensor add an additional fourth pulse, as shown, which is ignored by all ATCRBS transponders, but is recognized by DABS transponders as indicating that they should reply with a DABS all-call reply (see below) rather than a standard ATCRBS reply. The mode of an interrogation is determined by the P1-P3 spacing. Two modes are relevant for ARIES, these being modes A and C. A mode A interrogation requests that the aircraft transponder respond with its identity code as entered by the pilot on front panel switches. A mode C interrogation requests that the transponder return its coded altitude.

The P2 pulse is transmitted over an omni-directional antenna, whereas the other pulses are transmitted over the main antenna. The transponder compares the relative level of P2 to the other pulses and will respond only if P2 is lower. If this is not true it indicates that the interrogation was received via a sidelobe of the interrogator antenna.

The replies to either mode of interrogation consist of 16 amplitude modulated pulses as shown in Fig. 1.2-2. The two framing pulses F1 and F2 are always present and delimit the encoded altitude or identity data. The SPI pulse is transmitted upon the pilot's pushing a button on the transponder. It is used by the air traffic control display systems to cause the symbol for that aircraft to stand out on the controllers display. The controller can use this to identify a particular aircraft by requesting the pilot to push this button.

---

\* More details about DABS can be found in FAA-RD-74-189, "DABS: A System Description", by P.R. Drouilhet.

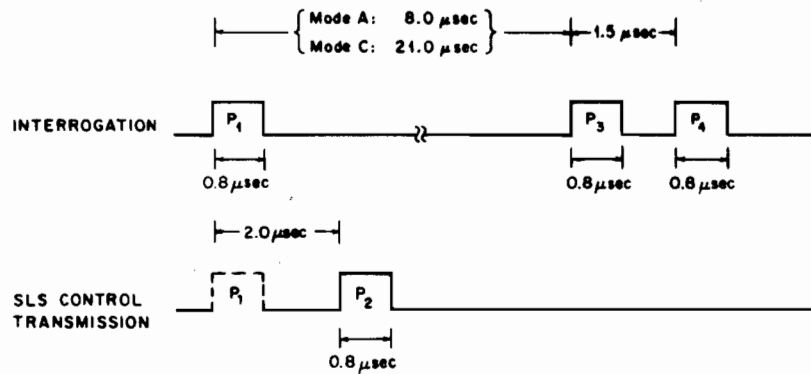


Fig. 1.2-1. ATCRBS/DABS all-call interrogations.

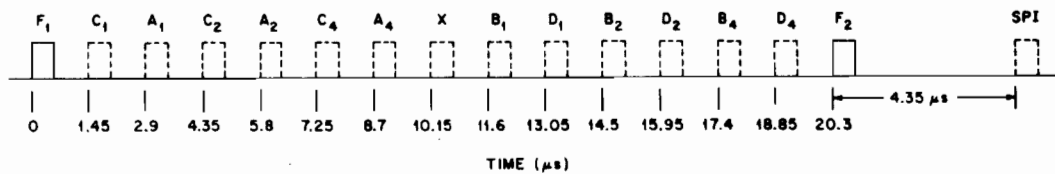


Fig. 1.2-2. ATCRBS reply format.

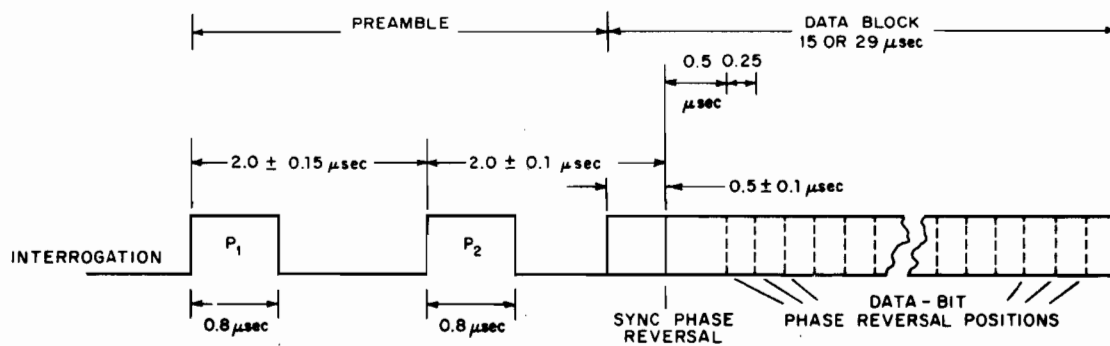


Fig. 1.2-3. DABS interrogation.

The sensor normally goes through a fixed cycle of interrogations, where each interrogation is distinguished by its mode and the interval between it and the previous interrogations. After each interrogation there is a listening interval which depends on the maximum desired sensor range. All transponders in the beam will respond and their replies appear at the sensor after the appropriate round trip times plus a transponder turnaround delay of 3  $\mu$ sec. Replies from different transponders may overlap if the aircraft are close to each other in range.

### 1.2.2 DABS Modes

As shown in Fig. 1.2-3, a DABS interrogation consists of an amplitude modulated preamble followed by a differential phase shift keyed (DPSK) modulated data block of either 56 or 112 bits. The preamble pulses have the effect of suppressing ATCRBS transponders, as the P1 and P2 levels are equal. This prevents these transponders from erroneously triggering on the message portion of the interrogation.

DABS replies are sent in a pulse position modulated (PPM) format as shown in Fig. 1.2-4. The preamble pulses are designed to allow the sensor to detect the reply with high probability even in dense interference due to ATCRBS replies triggered by interrogations from other sensors. (This form of interference is called "fruit".) The pulse spacing is also such that the probability of false preambles being generated by overlapping fruit replies is low. As with the uplink format, the downlink data field can contain either 56 or 112 data bits.

Within these modulation formats there are several different interrogation and reply types defined. These are shown in Fig. 1.2-5. They are distinguished by different values assigned to the first two data bits. Appendix D gives the definitions for each of the bit fields shown.

Of the reply types shown, the all-call is unusual in that it is the response of a DABS transponder to a sensor's mode A or C interrogation with the extra P4 pulse as discussed above. The P4 pulse identifies the interrogator as a DABS sensor, and so the DABS transponder replies with the all-call, which identifies it as a DABS transponder and contains the transponder's identity code. This is the acquisition mode of DABS.

Once a transponder has been acquired in this manner, subsequent interrogation is done via discrete (i.e., DABS) interrogations addressed specifically to that target. The address field is contained in the last 24 bits of the message field, and is overlaid with a parity code for purposes of error detection (on the downlink it is also used for error correction). One of the first interrogations has the DL field set to indicate to the transponder that it should not respond any further to ATCRBS/All-Call interrogations (i.e., those with a P4 pulse). This "lockout" state continues until either the sensor sends a new DL value to unlock the transponder or until no discrete interrogations have been received for 16 seconds.

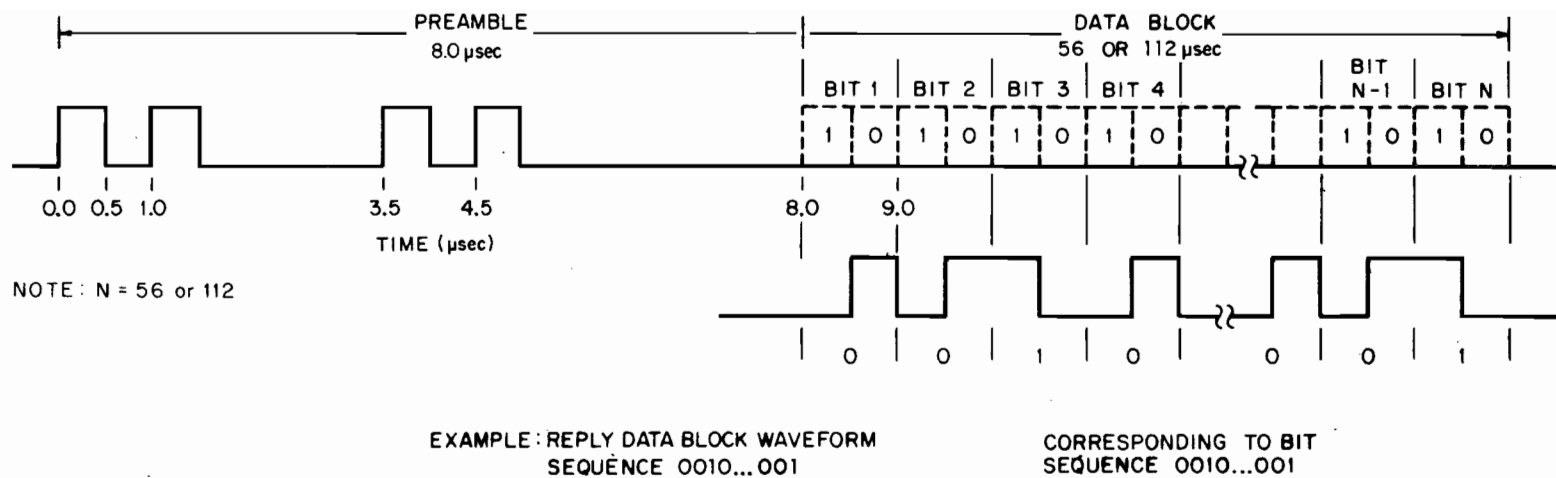
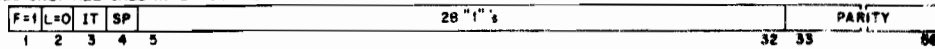
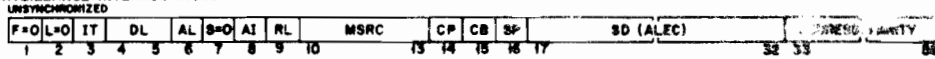


Fig. 1.2-4. DABS reply format.

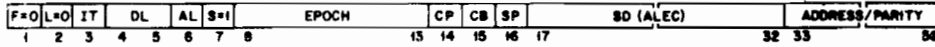
# DABS ONLY ALL-CALL INTERROGATION



## SURVEILLANCE INTERROGATIONS



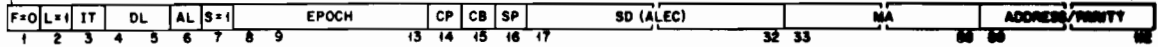
### SYNCHRONIZED



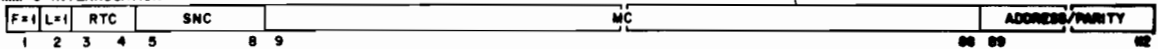
## COMM-A INTERROGATIONS



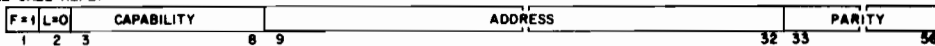
### SYNCHRONIZED



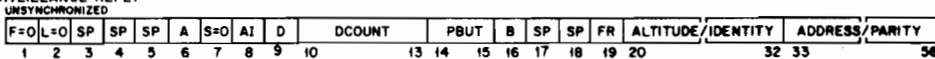
## COMM-C INTERROGATION



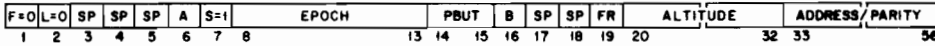
## ALL-CALL REPLY



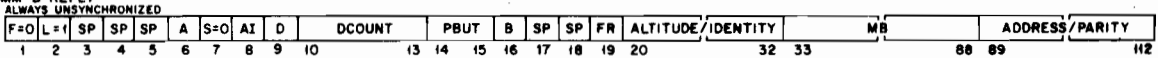
## SURVEILLANCE REPLY



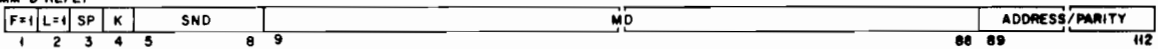
### SYNCHRONIZED



## COMM-B REPLY



## COMM-D REPLY



A:	Alert	DL:	DABS All-Call Lockout	MB:	Air-to-Ground Data Link Message
AI:	Altitude/Identity Designator	EPOCH:	Synchronous Reply Time	MSRC:	Air-to-Ground Data Link Message Source
AL:	ATCRBS Lock-out	F:	Format Type	PBUT:	Pilot Acknowledgment Buttons
ALEC:	Altitude Echo	FR:	Flight Rule	RL:	Reply Length
B:	Air-to-Ground Data Link Message Waiting	IT:	Interrogator Type	RTC:	Reply Type for Comm-C Interrogations
CB:	Clear Comm-B	K:	Extended-Length Message Control Indicator	S:	Synchronization Indicator
CP:	Clear PBUT	L:	Data-Block Length	SD:	Special Data
D:	Air-to-Ground Extended-Length Message Waiting	MA:	Ground-to-Air Data Link Message	SP:	Spare
DCOUNT:	Number of Segments in Air-to-Ground Extended-Length Message	MC:	Ground-to-Air Extended-Length Message Segment	SNC:	Segment Number of Ground-to-Air ELM Segment
		MD:	Air-to-Ground Extended-Length Message Segment	SND:	Segment Number of Air-to-Ground ELM Segment

Fig. 1.2-5. DABS data block format.



If the transponder receives an interrogation with the correct address and no detected errors, it will reply after a turnaround time (measured from the sync phase reversal to the leading edge of the first preamble pulse) of 128  $\mu$ sec. The reply length is determined by the RL bit in the interrogation.

Normal surveillance is conducted using surveillance interrogations and replies. The Comm-A interrogation and Comm-B reply are used only for the transfer of messages from ground to air or vice versa. The MA and MB fields, respectively, contain the additional message data. The Comm-C and D message formats involve a special communications protocol for efficiently transferring even longer messages. Synchronized interrogations and replies are used in a special protocol which can provide collision avoidance information to an appropriately equipped aircraft. All of these formats and their usage are described more fully in FAA-RD-74-62. DABS sensors currently being implemented do not use the DABS only all-call or synchronized interrogations.

### 1.2.3 Channel Time Allocation

As mentioned above, ATCRBS/All-Call interrogations occur in a fixed cycle, with each interrogation followed by a listening interval. These intervals are separated by much larger intervals of channel time during which DABS interrogations can be conducted. This time contains zero, one or more DABS cycles. Each cycle consists of a set of interrogations, followed by all the replies to those interrogations. The sensor schedules the interrogation times such that replies do not overlap each other, thus assuring a garble free channel (except, of course, for the fruit replies mentioned above). The scheduling is such that replies occur essentially back-to-back. Targets are scheduled within a cycle in decreasing range order. This, combined with the relatively shorter lengths of interrogations as compared to replies, assures that the interrogations also cannot overlap and that if target A is interrogated before target B, then A's reply will appear before B's reply. Fig. 1.2-6 shows the relationships described above.

### 1.2.4 Azimuth Determination

Another important feature of the DABS system is the use of a monopulse technique to allow the determination of an aircraft's azimuth to within  $0.1^\circ$  on the basis of a single reply. This is accomplished by generating two antenna patterns, the sum ( $\Sigma$ ) and difference ( $\Delta$ ) beams, as shown in Fig. 1.2-7. A third pattern, the omni-directional ( $\Omega$ ) pattern is also used for transmitting the P2 pulse for sidelobe suppression (SLS) purposes.

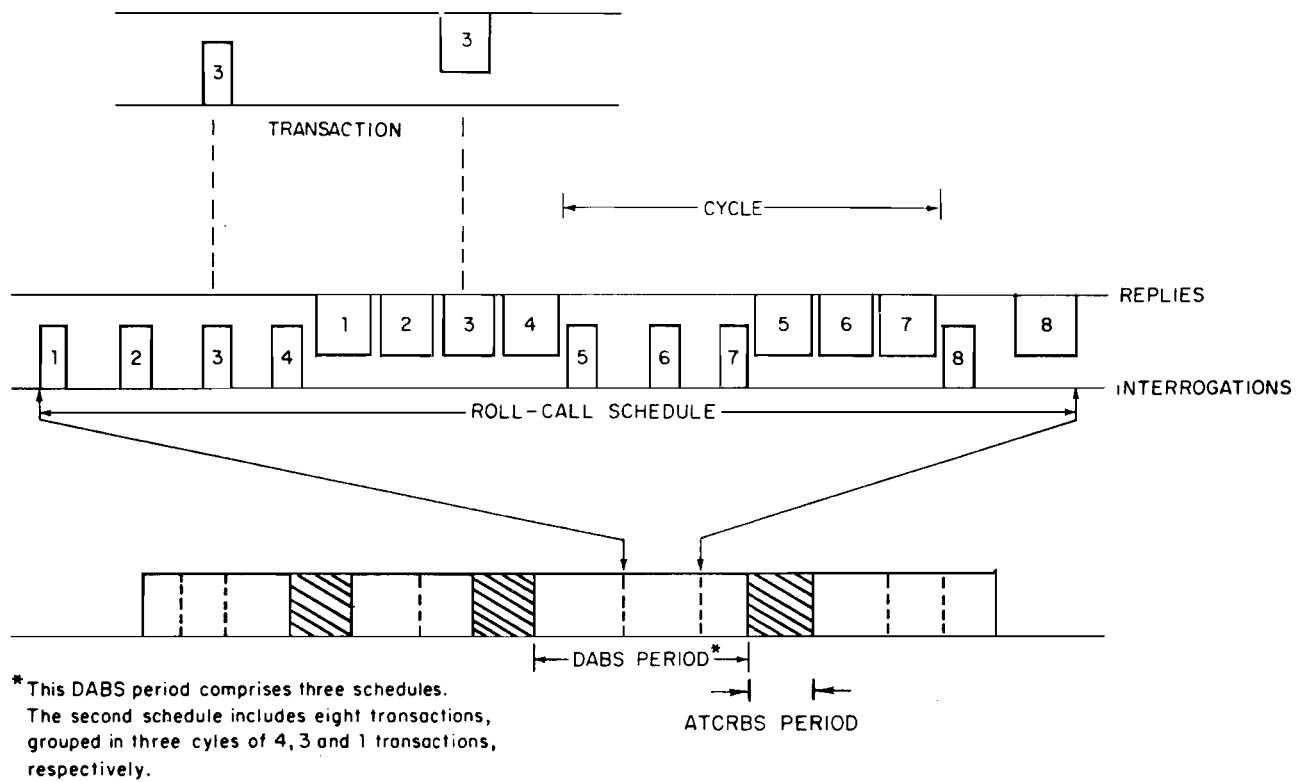
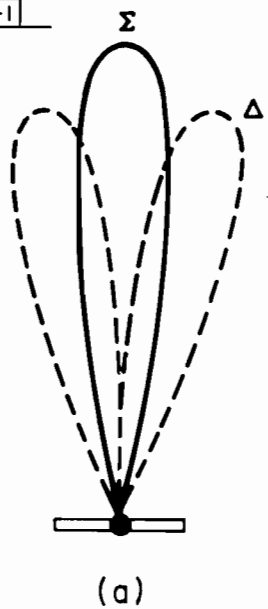
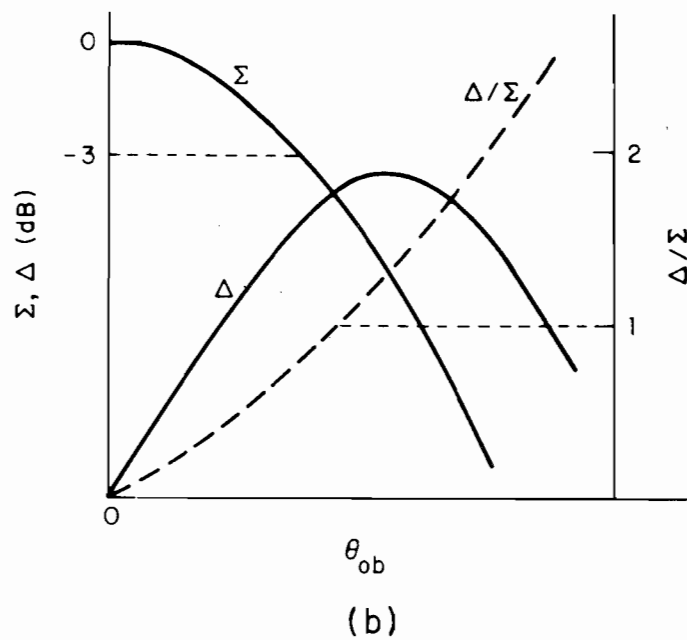


Fig. 1.2-6. DABS roll-call scheduling.

18-4-16404-1



18-4-16404-2



Σ SUM  
Δ = DIFFERENCE  
 $\theta_{ob}$  = OFF-BORESIGHT ANGLE

Fig. 1.2-7. Sum and difference antenna patterns.

To determine the offboresight azimuth of a target, the receiver circuitry essentially compares the relative amplitude of the signal on the  $\Delta$  channel to the signal on the  $\Sigma$  channel. Left and right of boresight can be distinguished due to the  $\Delta$  signal being in phase with the  $\Sigma$  signal on one side, and  $180^\circ$  out of phase with it on the other side. A detailed discussion of the implementation of this process is contained in FAA-RD-76-219. This produces a number which can then be used by software to perform a lookup in a calibration table to determine the offboresight angle measurement. This is then combined with the boresight azimuth as determined from a shaft encoder to obtain the true target azimuth.

The signals from the  $\Delta$  and  $\Omega$  channels are also compared with the  $\Sigma$  signal for purposes of receive sidelobe suppression (RSLS). If  $\Delta$  or  $\Omega$  is greater than  $\Sigma$ , a reply is declared to be a sidelobe reply. While this is not strictly true (i.e., it may be on the edge of the mainbeam) this is a region in which the monopulse azimuth estimate is less accurate and it is undesirable to process such replies.

#### 1.2.5 Back-to-Back Antenna Operation

The long range sensors used to provide surveillance data for enroute control centers typically require 12 seconds to make a complete scan. Proposed automated ground based collision avoidance systems require more frequent position updates in order to be able to accurately predict the positions of turning targets. Therefore, enroute DABS sites are to be equipped with back-to-back antenna systems to provide a position measurement every six seconds. The sensor indicates to the antenna system which antenna is to be used for each interrogation and reply.

#### 1.2.6 Primary Radar Data

The DABS sensor provides an input connection for receiving digitized radar reports. This is to allow data from a co-located primary radar to be entered into the DABS sensor and correlated with the beacon reports. Such reports can then be marked as "radar reinforced". Uncorrelated radar reports are forwarded to the air traffic control center by the sensor, to allow tracking of aircraft not equipped with radar beacons.

#### 1.2.7 The DABS Network

DABS sensors can be connected together into a network by means of telephone lines. This serves several purposes. First, if a sensor fails, adjacent sites can expand their areas of coverage to take over the targets being handled by the failed sensor. Second, if an individual target should temporarily become invisible at one site, information about it can be requested from other sites which may still be in contact. Third, as an aircraft flies

from one sensor's coverage area to another, it is necessary to coordinate the handling of message traffic and lockout by both sites so that an orderly transfer of primary responsibility is made.

### 1.3 ARIES System Summary

#### 1.3.1 Capabilities

ARIES is capable of simulating a beacon environment of up to 400 transponders, with certain limitations, described below, on the amount of bunching in azimuth that can be handled. There can be any mix of ATCRBS and DABS transponders. Not all DABS interrogation and reply types are handled. Those that are handled and also the special data bit protocols that are simulated are described in Section 1.3.2.2 below.

Along with the simulated traffic ARIES can generate a simulated fruit environment. The arrival times of fruit replies are not based on the traffic model. To do this would also require modeling the nearby interrogators that cause these interfering replies to be generated. Instead, fruit is modeled as a random process with Poisson statistics. The operator can control the average fruit rate by setting parameters in a file on the system disk.

For both the simulated transponder (controlled) replies and fruit replies ARIES provides the necessary signals to accurately simulate the monopulse offboresight angle. Also, an omnidirectional signal is provided so that sidelobe replies can be simulated. These signals are connected to the DABS sensor via an interface specific to ARIES. Inside the sensor they are summed with similar signals from the sensor's own antenna. This allows a simulated environment to be superimposed on a live environment if desired.

In addition to the beacon data, ARIES provides simulated digitized radar data in the output format of a Production Common Digitizer (PCD). The radar targets correspond to the simulated beacon targets. The reported coordinates are those that would be seen by a primary radar whose antenna rotates with the beacon antenna about the same axis.

Finally, ARIES is also capable of multisite operation in order to exercise the multisite network aspects of DABS. This is accomplished by locating an ARIES at each sensor site. The traffic model at each site is coordinate adjusted so that each sensor sees a view of the environment consistent with that of the other sensors. The ARIES sites are linked by communications lines which allow them to maintain time synchronization and also to maintain consistent transponder internal state data for DABS targets that can be seen by more than one site.

Each of the above mentioned features of ARIES is described in more detail in the following sections. Fig. 1.3-1, an overall view of the ARIES hardware, will be useful in understanding those paragraphs describing the implementation of these features.

### 1.3.2 Simulated Transponders

#### 1.3.2.1 Interrogation/Reply Cycle

Interrogations are received by ARIES from the DABS sensor at 1030 MHz and processed by the receiver circuitry. The receiver transfers a data block to the computer giving the type of interrogation, the time of arrival, the boresight azimuth of the antenna, whether the interrogation was a front antenna or back antenna interrogation, and any data content. The azimuth data is provided by the azimuth decoder/simulator. This can operate in either of two modes. In the decoder mode, azimuth change pulses (ACP's) and azimuth reference pulses (ARP's) are received from the sensor's antenna system, and used to increment and reset an azimuth register, respectively. The ACP's and ARP's are then "daisy-chained" back to the sensor. In effect, ARIES is inserted into these lines between the antenna and the sensor. In the simulator mode, ACP's and ARP's are produced by ARIES and sent to the sensor. ACP's and ARP's from the antenna, if an antenna is connected, are ignored. Thus, ARIES can be run with a sensor that is not connected to an antenna system. The decoder mode is needed for the case where live data from the sensor's own antenna and simulated data from ARIES are to be superimposed.

In the computer, the response depends on the type of interrogation. If it is an ATCRBS/All-Call interrogation, all ATCRBS transponders and unlocked DABS transponders that are in the antenna beam are located in the track file which is resident in the computer's memory. If it is a discrete interrogation the appropriate track record is located by means of its DABS address. Reply data blocks are generated for each such target. These specify the time of reply, the reply type, the reply power level, the simulated offboresight angle, and the data content. This data is transferred to the controlled reply generator circuitry. This generates the appropriate reply waveforms at the 60 MHz intermediate frequency of the DABS sensor. The parity check encoding for DABS replies is also performed here. Up to three overlapping replies can be simulated, there being three independent reply generators. The signals from all of these are summed together in the IF combiner and output to the sensor.

#### 1.3.2.2 Interrogation and Reply Types Handled by ARIES

ARIES is capable of processing both mode A and mode C ATCRBS/All-Call interrogations. [The system software is not designed to process ATCRBS interrogations without the P4 pulse. It would be relatively easy to modify the system for use with an ATCRBS sensor, however]. The reply modes, as mentioned above, are mode A or mode C ATCRBS replies from simulated ATCRBS

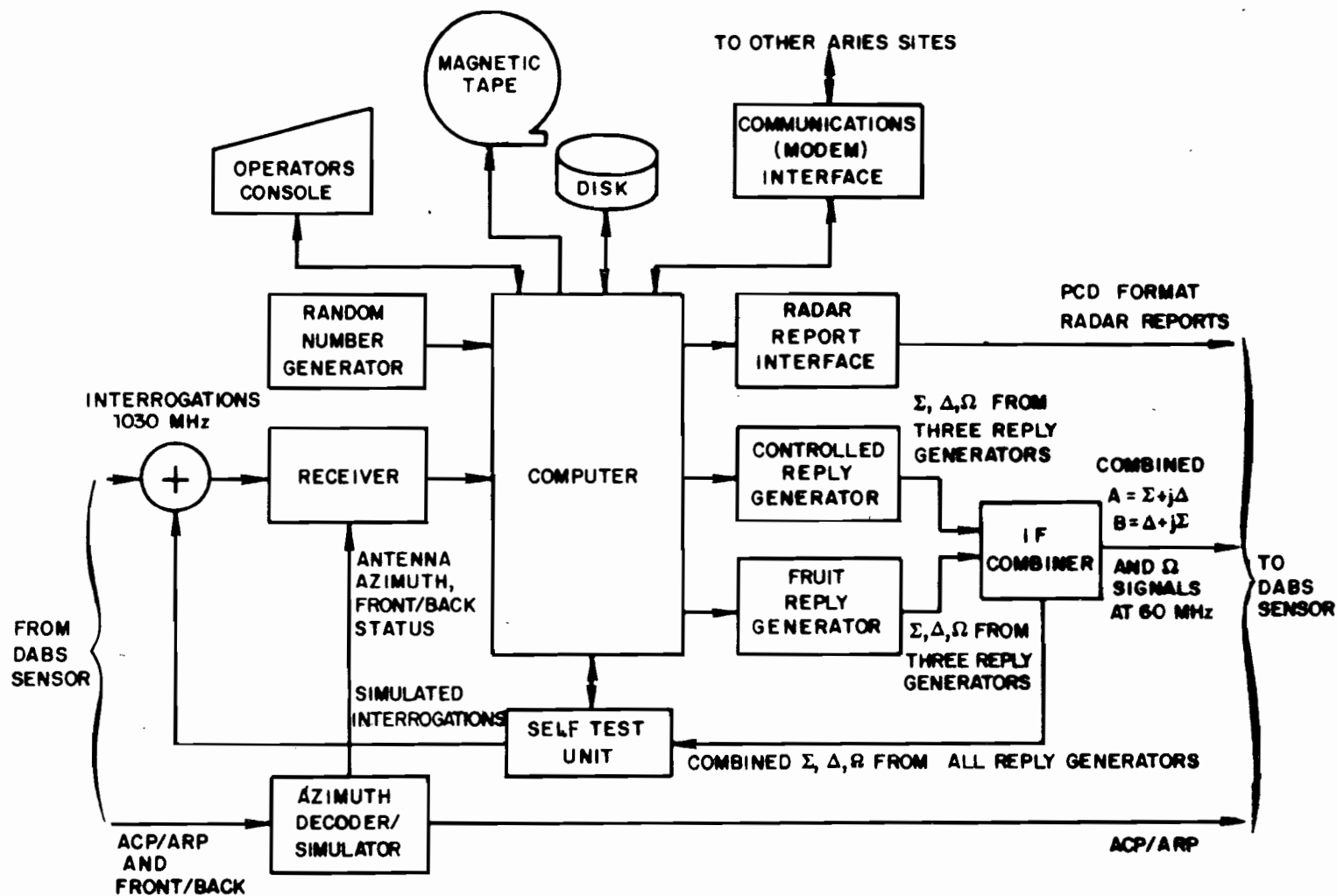


Fig. 1.3-1. Overall system diagram.

transponders and all-call replies from DABS transponders. The SPI bit can be set in ATCRBS replies under the control of the traffic model data. ARIES is not capable of simulating military emergency codes, which consist of several sequential ATCRBS replies.

Of the DABS modes, ARIES is capable of processing surveillance and Comm-A interrogations and of generating surveillance and Comm-B replies. Comm-C interrogations and DABS-only all-call's are ignored by ARIES. Thus, the simulated transponders are typical of what might be expected of a general aviation or small aircraft transponder, as these aircraft are not likely to carry the equipment necessary for generating or displaying the extended length messages (ELM's) for which the Comm-C and Comm-D formats were designed. The rationale for leaving out these modes is that they are not expected to constitute a significant portion of the total system load and would add significantly to ARIES' complexity. The DABS-only all-call is not used in the current sensor design.

Certain data bits of discrete interrogations and replies are used to implement various message protocols and also to allow the ground to control the transponder state. A summary of these functions is presented here. Please see Appendix D for a definition of the message bits involved, and FAA-RD-74-62 for a more extensive description of all these functions. The functions supported by ARIES are:

1. Locking out the transponder to ATCRBS/All-Call interrogations.
2. Handling synchronous interrogations and replies to support a form of collision avoidance system (Synchro-DABS).
3. Reporting of transponder altitude or identity code.
4. Ground-initiated Comm-B downlink (i.e., the sensor requests information from a particular device attached to the transponder). Note, however, that the downlink message field always contains the same data for all Comm-B replies. No attempt is made to simulate any particular on-board device or vary the data from reply to reply.
5. Uplink message transfer and pilot acknowledgement of uplinks. ARIES ignores the message content except for the first bit which indicates if a pilot acknowledgment is requested. If it is, an acknowledgment will be returned on the next scan, and the sensor can clear the acknowledgment by setting the appropriate uplink bit. The value of the pilot acknowledgment field is determined by the traffic model data.
6. Air-initiated Comm-B downlink. The aircraft can initiate transfer of data to the ground by setting a downlink bit. This is controlled by the traffic model. The ground then requests a Comm-B downlink. The downlink message field will contain the same constant data as for a ground initiated downlink. The sensor sets a particular bit in an interrogation to acknowledge receipt and terminate the request.



7. Alert. The appropriate downlink bit is set under control of the traffic model. This causes the sensor to request the transponder's identity code.
8. VFR/IFR. The appropriate downlink bit is controlled by the traffic model to indicate instrument flight rules (IFR) or visual flight rules (VFR) status.

A final characteristic of the simulated transponders that can be controlled via the traffic model is the reply probability. Each simulated transponder has its own reply probability value, and this can be used to simulate various round-reliability conditions. The hardware random number generator shown in Fig. 1-9 is used to determine whether or not a target will reply.

#### 1.3.3 Radar Report Generation

As targets in the ARIES track file enter the antenna beam, a simulated radar report is prepared for them by the computer. The range and azimuth coordinates used are exactly those used to generate the beacon reports, but converted to the appropriate units. This data is transferred to the radar report generator, which outputs the data to the sensor in a format identical to that of the output interface of a Production Common Digitizer (PCD). The data is transferred to the sensor just after the targets have left the beam.

A system-wide blip/scan ratio can be specified by the operator, and in conjunction with the random number generator is used to determine whether a given target will have a radar report on a given scan.

The interface is capable of transferring over 120 target reports per second.

#### 1.3.4 Fruit Reply Generation

ARIES is capable of generating ATCRBS fruit replies at rates up to about 50000 replies per second. No DABS fruit is generated. These high rates are required to test the performance of the DABS sensor's reply processing circuitry at the interference levels at which it is capable of operating. Because of the high rates required, the entire reply generation process is performed in hardware. The computer specifies to the fruit generator the average fruit rate desired, the fraction of the replies that are to appear in the mainlobe as opposed to the sidelobes, a particular ATCRBS reply code, and the fraction of replies that are to have this code. The latter two items allow a particular code to be emphasized in the otherwise randomly generated stream of reply codes. One likely candidate for emphasis might be, for example, the non-discrete VFR code 1200.

Reply parameters such as offboresight angle, power, and code bits are determined by pseudo-random generators. The term pseudo-random is used to indicate that the sequence of replies can be repeated exactly by reinitializing the fruit generator (i.e., the sequence is deterministic) although the reply statistics are designed to match those of certain random distributions.

The fruit inter-arrival times match those expected from a Poisson process, i.e., the probability that any given inter-arrival interval lies between  $t$  and  $t+dt$  is given by

$$\lambda e^{-\lambda t} dt,$$

where  $\lambda$  is the average arrival rate as specified by the computer. The allowable range for  $\lambda$  is from 1000 to 50000 fruit per second. However, there are only three reply generators, so no more than three overlapping replies can be simulated even though this is a high probability event at a fruit rate of 50000 fruit/second. The number of reply generators provided was chosen to match the DABS sensor's ability to decode a legitimate reply in the presence of up to three overlapping fruit replies.

The monopulse angle of the fruit replies is uniformly distributed across the entire range of offboresight angles that are generated by ARIES.

The fraction of replies specified by the computer appear in the main antenna lobe, the others appearing to be sidelobe replies. Sidelobe replies are simulated by attenuating the  $\Sigma$  and  $\Delta$  signals below the  $\Omega$  signal.

The reply powers of mainlobe replies follow the relation

$$P_{m\ell} = -20 - 20 \log r_{m\ell} \text{ dBm}$$

where  $r_{m\ell}$  is uniformly distributed between 1 and 100 nm. For sidelobe replies the corresponding relation is

$$P_{s\ell} = -55 - 20 \log r_{s\ell} \text{ dBm}$$

where  $r_{s\ell}$  is uniformly distributed between 1 and 32 nm. (These are the apparent sensor RF port power levels. Since the ARIES input is at IF the

actual signal levels differ from these by a constant offset). Note that some sidelobe replies will be generated below the sensor's minimum usable signal level (MUSL) of -79 dBm. This is done in order to simulate the summing of low level fruit replies to give signals that exceed the MUSL.

The reply codes, other than the specified fraction that are generated with a fixed code, are generated according to a distribution that adjusts the probability of each of the ATCRBS data bits roughly according to the likelihood of their being set in a real fruit reply. Thus, not all reply codes are equally likely. For example, certain bit patterns are illegal in mode C replies, and these are therefore less likely to occur, although they are legal mode A patterns. Similarly, patterns corresponding to the mode C replies of very high flying aircraft are less likely than others.

### 1.3.5 The Traffic Model

The traffic model is stored on the system disk. Each logical record specifies the position and velocity of one aircraft at a particular time, as well as such items as its altitude and identity codes and those items of state information mentioned in Section 1.3.2.2 as being controlled by the traffic model. These records are sorted in increasing time order.

The ARIES computer reads these records from the disk. When the current system time matches the time tag of the model record, that record is used to update the track file entry for that target (or to start a new entry if this is the first appearance of the target). Typically, targets are updated periodically by the model at about 4 second intervals. This is not required, however. If a target is not updated by the model, the ARIES computer continues to update its position based on the velocity data in the last model record received for that target. However, since the positions and velocities are specified in  $\rho$ ,  $\theta$ ,  $\dot{\rho}$ ,  $\dot{\theta}$  coordinates, and a simple linear prediction is performed (i.e.,  $\rho \leftarrow \rho + \dot{\rho} \Delta t$ ), the targets will fly spiral paths in X, Y space. Frequent updates from the model can create arbitrary flight paths. In that case the velocity data is used only to adjust the position for the short interval between the model update time and the actual time of interrogation of the target.

A maximum of 400 targets can be simulated by ARIES. Any mix of DABS and ATCRBS targets is possible. In addition to the overall limitation on the number of targets, there are limitations on the degree of bunching in azimuth that is allowed. ARIES is capable of handling at least the degree of bunching specified for the DABS sensor, which is capable of handling:

- a. 50 aircraft in an  $11.25^\circ$  sector, for not more than 8 consecutive sectors.
- b. 16 aircraft in a  $1.2^\circ$  azimuth wedge for up to three contiguous wedges.

The data in the traffic model can be created in a variety of ways. However, for purposes of system tests of the DABS sensor the Los Angeles Basin Standard Traffic Model for 1982 will be used. Report FAA-RD-73-86 and the references mentioned in that document describe the generation and format of this model. A program has been written which will take any model in the format of the Los Angeles model and convert it to the appropriate format for input to ARIES. This program also allows the user to control certain transponder state information (i.e., the downlink request bit, the pilot acknowledgement bits, etc). Options are provided for generating these either randomly or under user control on a target by target basis.

#### 1.3.6 Multi-Site Operation

In order to exercise a network of DABS sensors, an ARIES must be located at each sensor site. These ARIES sites are then connected together by means of telephone links. The links are required for two purposes. First, they are needed in order to start the simulations in synchronism at all sites and to verify that this synchronism is being maintained. Second, DABS transponders have certain states (i.e., lockout) that are controlled by interrogations from the sensor and cannot be pre-computed as part of the traffic model. If a transponder is visible to more than one site it is necessary to assure that this state information is consistent among all sites. Therefore, for example, if a transponder is told to lockout all-call interrogations at one site, the ARIES at that site will inform all other ARIES of this occurrence so that the transponder appears to be locked out at all sites.

The bulk of the traffic information can be pre-computed. A master traffic model in the format of the Los Angeles Basin Standard Traffic Model is run through the model conversion programs once for each ARIES site. Each run produces an ARIES formatted model file that has been adjusted to the geographic coordinates of one of the ARIES sites. However, since all these models are produced from the one master model, the target positions and transponder states are consistent among all sites at all times. This is necessary so that if two sensors attempt to exchange information on a target the positions will correlate after being converted from one  $\rho, \theta$  coordinate system to the other. Also, the data from all sites will be converted to a common X, Y coordinate system at any air traffic control center fed by these sensors, and position reports from all sites for any given target must register.

At any given ARIES site a maximum of 400 targets can be simulated. However, depending on the degree to which the targets visible at each site overlap, up to 800 targets for two sites or 1024 targets for three (or more) sites can be simulated. The limitation of 1024 is determined by the fact that ARIES uniquely identifies each target in the entire network using a 10 bit identity code.

### 1.3.7 Data Recording

ARIES records a limited amount of data in real time, primarily for debugging purposes. However, data is recorded indicating which targets replied to any ATCRBS/All-Call interrogation and whether or not a reply was generated for each discrete interrogation (and if not, why not). More extensive recording is available for a limited number of targets. This includes a record of all targets that were considered for an ATCRBS/All-Call reply and, if they did not reply, an indication of why. For DABS targets all of the interrogation data block except the unused message fields and the entire reply block are recorded. Thus a significant amount of information is available about the interrogations transmitted by the sensor and the replies that were generated by ARIES. This could potentially be used in analyzing the sensor's performance.

This data is normally recorded on magnetic tape. However, there is an option to record on the disk. Due to the limited size of the disk file allocated for this purpose only a few minutes of recording of a dense environment is possible. If the recording process reaches the end of the file it moves back to the beginning of the disk file and begins overwriting the old data.

### 1.3.8 Self Test Capabilities

ARIES is a relatively complex system and is being used to test an even more complex system. It is essential, therefore, that there be some reliable way of testing that the ARIES hardware is functioning correctly. This testing must be independent of the DABS sensor. To provide this capability, each digital device has diagnostic modes available. These by and large fall into the category of writing data to a register or external buffer memory and then reading it back. In some cases, the device control bits affect which data is read and so their functioning can be tested also.

However, it is not possible to test all the digital logic in this fashion, and none of the RF or IF circuitry is tested. Therefore, a self test unit was included in ARIES to provide complete loop tests of the interrogation processing and reply generation circuitry. It is also useful for software checkout purposes. The self test unit consists of two separate parts. The first is capable of generating all the interrogation modes that are processed by the receiver circuitry. These interrogations are generated at 1030 MHz and summed with the signals received from the DABS sensor. Which interrogation is sent and the time of transmission are under computer control. The other part of the self test unit consists of circuitry for demodulating and sampling the  $\Sigma$ ,  $\Delta$ , and  $\Omega$  channels. The computer specifies which channel is to be sampled and whether an ATCRBS or DABS reply is expected. When a pulse is detected by the self test unit an amplitude sample is taken and the data bits are sampled. No decoding of the DABS parity check is performed. This must be done by the software. The amplitude sample and

the data bits are sent to the computer for comparison with the reply data sent to the reply generator under test. Both the controlled reply generator and the fruit reply generator can be tested by this means. In the case of the fruit reply generator this requires the use of a special diagnostic mode whereby the computer specifies the reply parameters and data, rather than having these randomly generated.

Extensive diagnostic software has been written for ARIES to make use of both the simple digital loopback modes and also the capabilities of the self test unit.

## 2.0 DETAILED ARCHITECTURE AND OPERATION

### 2.1 Overall Architecture

Figure 2.1-1 depicts the components of the ARIES system and their interfaces to the DABS sensor, and the DABS antenna.

Components of the ARIES system are controlled and interfaced by a 16-bit mini-computer (Data General, Eclipse S/200), with 32K of core and with high speed I/O interface capability. Peripherals include a teletype for providing operator communications, and a disc cartridge and magnetic tape for storing the system programs and the ARIES controlled traffic model.

The principal special purpose devices which are interfaced to the computer are the uplink receiver and three controlled ARIES targets (CAT's) which are interfaced through a controlled reply controller (CRC).

The three fruit ARIES targets (FAT's) are driven by a similar controller called the fruit controller (FC). Instead of obtaining replies from the computer, as the CRC does, the FC obtains replies from a Random Process Generator (RPG). The RPG generates ATCRBS fruit replies with random power, random offboresight angle, random ATCRBS code and random values of exponentially distributed delay-to-trigger time. Overall fruit characteristics such as fruit rate, mainbeam/sidelobe ratio, and ratio of fixed to random code are controlled by the CPU.

Outputs of the six target generators are combined at IF in the IF combiner network, to provide a single set of IF inputs at 60 MHz to the DABS sensor test input ports.

Names and locations of the ARIES special purpose devices are identified in Fig. 2.1-2 and Table 2.1-1.

The azimuth change pulse decoder/simulator accepts azimuth change pulses, azimuth reference pulses, and front-back signals from the DABS antenna. These signals are fed back to the sensor in a daisy-chain fashion when the sensor is operating with a real DABS antenna. Simulated azimuth change pulses and reference pulses are sent to the DABS sensor when it is operating without an external antenna. The azimuth change pulse decoder simulator also transmits azimuth words to the uplink receiver.

ARIES range timing is achieved with a single 16-MHz clock. Range Counters in each of the controlled ARIES targets and the self test unit are reset each time an All-Call interrogation is decoded by the uplink receiver. This event is indicated by the ATCRBS/All-Call TOA pulse. These counters are not reset by discrete interrogations.

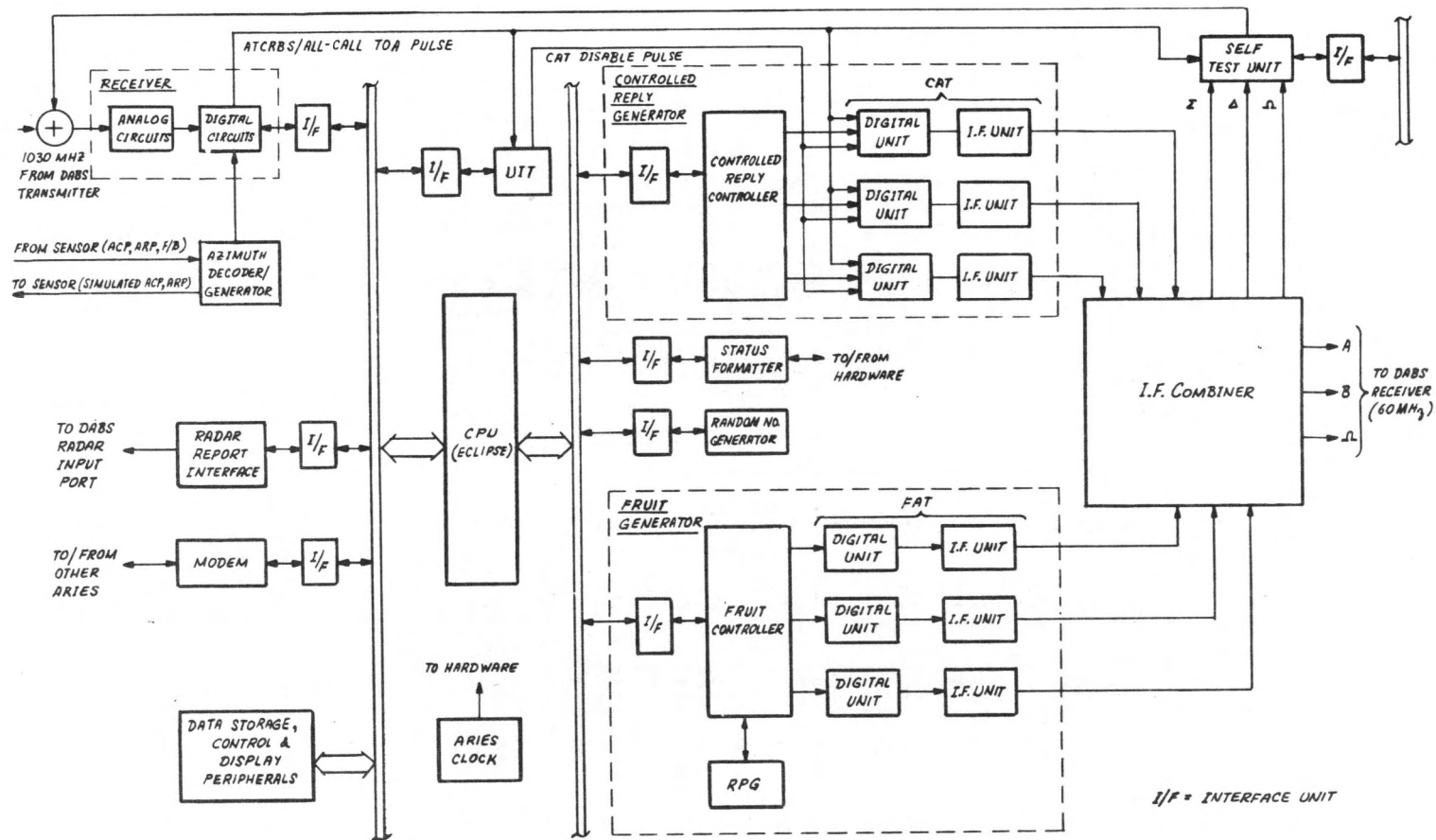


Fig. 2.1-1. ARIES block diagram.



P130-1439

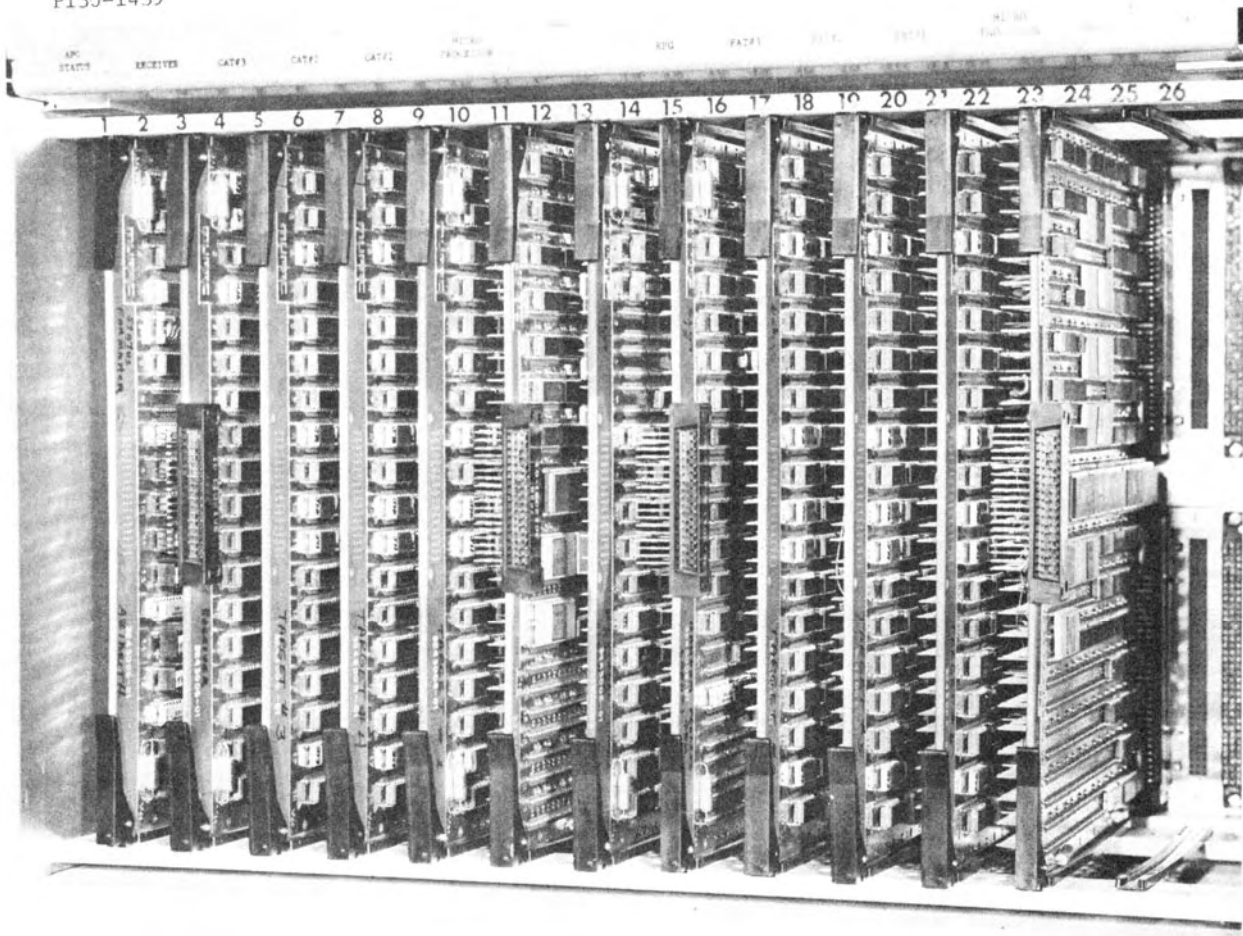


Fig. 2.1-2. Digital subsystem, right view.

TABLE 2.1-1  
ARIES PRINTED CIRCUIT BOARD SLOT ASSIGNMENTS

<u>Assembly</u>	<u>Device</u>	<u>Slot</u>
Eclipse S/200 CPU	TTYI	6M*
	TTYO	6M
	Real Time Clock	6M
	Versatec Printer	8M
	Receiver	10M
	Universal Interval Timer	10M
	Radar Report Interface	12M
	Self Tester	14M
	Status Formatter	14M
	Fruit Reply Generator	9E*
	Random Number Generator	9E
	Controlled Reply Generator	12E
	Magnetic Tape	14E
	Disk	15E
	Synchronous Line Adapter	16E
Digital Subsystem	Clock	13
	Universal Interval Timer (UIT)	13
	Receiver	3
	Azimuth Change Pulse Decoder	1
	Microprocessor	11
	Controlled Reply Generator No. 1 (CAT No. 1)	5
	Controlled Reply Generator No. 2 (CAT No. 2)	7
	Controlled Reply Generator No. 3 (CAT No. 3)	9
	Microprocessor	23
	Fruit Generator No. 1 (FAT No. 1)	21
	Fruit Generator No. 2 (FAT No. 2)	19
	Fruit Generator No. 3 (FAT No. 3)	17
	Random Process Generator (RPG)	13
	Status Formatter	1

\*  
M = Main-Frame  
E = Expansion-Chassis

The ARIES system is required to simulate targets with a minimum range of 1 nmi from the DABS sensor. The round trip delay to a 1 nmi target is about 12  $\mu$ sec. Adding the 3- $\mu$ sec reply delay of an ATCRBS transponder, it is clear that only 15  $\mu$ sec are available for ARIES to respond to an All-Call interrogation when simulating a close-in ATCRBS target. Fifteen  $\mu$ sec is insufficient time to provide for the interrupt latency in the central processing unit. Therefore, it is necessary to predict the time of arrival of ATCRBS interrogations. The universal interval timer (UIT) is used for this purpose. The CPU transfers to the UIT the expected time until the ATCRBS interrogation. During the interval between one All-Call interrogation and the next, the software has time to prepare the entire set of replies for the next predicted interrogation. Approximately 56  $\mu$ sec before the expected arrival time of the next ATCRBS interrogation, the UIT requests an I/O interrupt, thereby indicating that no more DABS interrogations are anticipated and that 56  $\mu$ sec later the controlled ARIES targets must be prepared to reply to an All-Call interrogation.

Other devices shown in Fig. 2.1-1 which are also interfaced to the CPU are the Radar Report Interface, the Random Number Generator, and the Status Formatter. The Radar Report Interface is a DMA device. Its function is to generate radar target reports in Production Common Digitizer (PCD) format for serial transmission to the radar input port of the DABS sensor. The random number generator provides the CPU with a rapid succession of pseudo random numbers; and the Status Formatter enables the CPU to read the state of the hardware and of all manually controllable switches in the system during initialization.

Circuitry is included which may be used to loop-test ARIES under computer control, completely independently of the DABS sensor under test. The self-test unit (STU) provided to do this is capable of testing for:

- correctness of simulated reply data content
- correctness of amplitude in the sum, delta, and omni IF outputs
- time of reply or range readout
- correctness of interrogation decoding.

#### 2.1.1 I/O Interface Conventions

Most of the Lincoln built ARIES devices described in this document are interfaced to a Data General Eclipse computer using Data General I/O protocol.

Thus it is necessary to understand the pertinent Data General conventions in order to completely understand the operation of each piece of hardware described. These conventions are summarized below\*.

Data General interfaces have two modes of data transfer. One is programmed I/O, where the transfer of each word is controlled by the software. The other is data channel I/O, where the software merely specifies to the device the address of a buffer area in memory (via programmed I/O). All transfers to or from that buffer are controlled by the device interface, independently of the software.

Programmed I/O is conducted between a CPU register and one of three "registers" on the device interface, the A, B, and C registers. These may or may not be implemented as unique hardware registers. It is perhaps better to regard A, B, and C as device addresses of particular components of the interface, rather than as actual registers. Output to these registers is done via the DOA, DOB, and DOC instructions, respectively. The corresponding input instructions are DIA, DIB, DIC.

Each device interface has three status flip-flops which are controlled by the software, the BUSY, DONE, and INTERRUPT DISABLE flip-flops. If BUSY is set, the device is active. DONE is always zero in that circumstance. When the device completes its operation, it clears BUSY and sets DONE. This automatically causes an interrupt to be generated at the CPU, unless the INTERRUPT DISABLE flip-flop is set, in which case the interrupt will be inhibited until INTERRUPT DISABLE is cleared. If DONE is set when INTERRUPT DISABLE is cleared, an interrupt will be generated immediately. A device is completely inactive when both BUSY and DONE are zero.

There are several signals available to control these status flip-flops. The START line, when pulsed, will set BUSY and clear DONE. This usually has the effect of activating the device. The CLEAR line, when pulsed, will set both of these to zero. This may be used after receiving an interrupt in order to clear the interrupt. The IORST line is similar to CLEAR, except that the INTERRUPT DISABLE flip-flop is also cleared.

To pulse the START and CLEAR lines the letters S and C respectively are appended to the programmed I/O instruction mnemonics (e.g., DOAS, DIBC), or to a NO I/O TRANSFER instruction (NIOS, NIOC). There is a third control line, called the P(Pulse) line, which is operated similarly by appending a P to the I/O commands. It has no standard usage, and is interpreted differently by each device.

---

\* For complete description of the Data General I/O protocols and conventions refer to "DG Interface Designer's Reference Manual", DG document number 015-000031 and the ARIES "Programmer's Manual", which is included as Vol. 3 of this report.

The IORST line is pulsed by the I/O RESET instruction, or by toggling the RESET switch on the front panel. It is pulsed automatically by the CPU when power is first turned on. Note that the input and output instructions, and any START, CLEAR, or P pulse associated with them, affect only the addressed device, while an IORST affects all devices.

INTERRUPT DISABLE is normally set and cleared by a MASK OUT instruction. This instruction places the contents of a specified CPU register on the data lines, and then sends a special control pulse. Each device looks at only one of the 16 data bits, and loads the value of that bit into its INTERRUPT DISABLE flip-flop on receipt of the pulse.

For more extensive documentation of the operation of the Data General I/O system, see the Data General documentation.

## 2.2 Timing

### 2.2.1 ARIES Clock

The ARIES digital system operates synchronously using timing pulses and gates derived from a 16.0 MHz crystal oscillator in the ARIES clock\*. In its normal or "run" mode the ARIES clock generates continuous 30 ns pulses spaced 62.5 ns apart (16 MHz), and 62.5 ns pulses spaced 250 ns apart (4 MHz). A diagnostic or "single step" mode permits the generation of a single 30 ns pulse on the 16 MHz clock lines (16 MHz single step switch depressed), or a group of four 30 ns pulses on the 16 MHz clock lines and single 62.5 ns pulses on the 4 MHz clock lines (4 MHz single step switch depressed).

Operation of the ARIES clock may be visualized by examining the block diagram in Fig. 2.2-1. The sinusoidal output of the 16 MHz crystal oscillator is shaped to a symmetrical rectangular pulse output. In the run mode the 2:1 mux passes its symmetrical input to flip/flop IC<sub>1</sub> which generates the asymmetrical 30 ns 16 MHz output. Every fourth pulse of the symmetrical 16 MHz waveform is also passed to flip-flop IC<sub>2</sub> to create the 62.5 ns wide 4.0 MHz output.

In the single step mode one or four negative going pulses are selected by the "16 MHz single step" or the "4 MHz single step" flip-flops respectively to drive the output generating D flip-flops IC<sub>1</sub> and IC<sub>2</sub>.

---

\* Shares board space with the system clock and random process generator on the card located in Slot No. 13 (see Fig. 2.1-1) of the digital drawer.

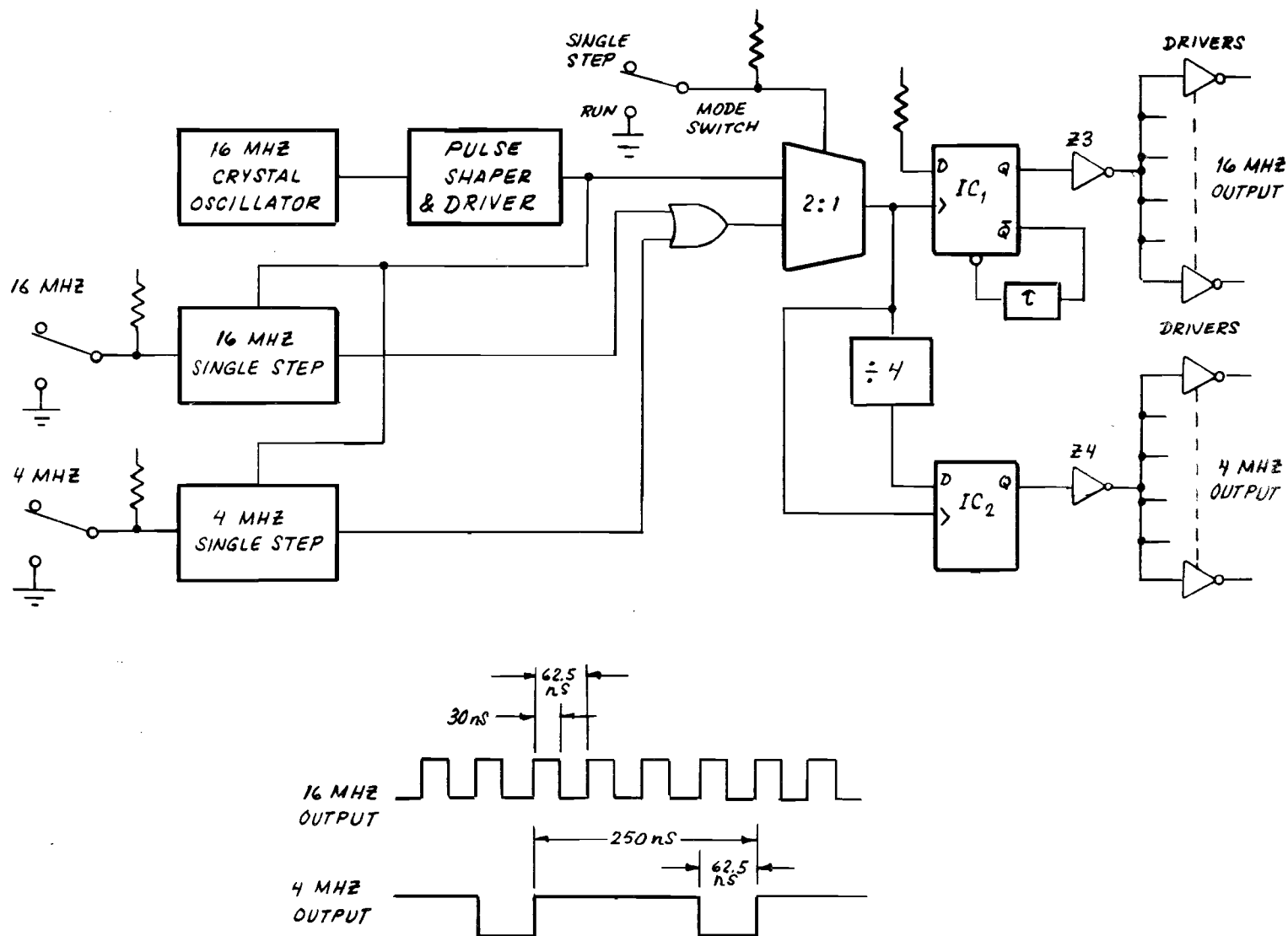


Fig. 2.2-1. ARIES system clock, block diagram and waveforms.

### 2.2.2 Universal Interval Timer (UIT)\*

The UIT consists of (refer to Fig. 2.2-2): (a) a 16-bit counter fed by 1  $\mu$ sec pulses obtained from the system clock -- and therefore capable of counting from 1 to 65535  $\mu$ secs, (b) a 16-bit latch to permit sampling the content of the counter without disturbing its operation, and (c) start, stop, and presetting logic. It receives timing inputs from the 16 MHz system clock, control inputs from the CPU via an 8-bit control bus, and interrogation pulses from the receiver. Its outputs are transmitted (via its interface registers) to the CPU on a 16-bit bidirectional bus which also transmits inputs from the CPU.

Normal operation is controlled via the 8-bit control bus as follows (refer to Table 2.2-1 for the action of each of the control bits):

- (1) Assume that the CPU has already calculated and stored the pulse repetition interval (PRI) of the ATRBS/All-Call interrogations from the previous set of transmissions of the DABS sensor, and that this interval is designated  $T_n$  ( $T_n$  need not be equal for each transmission).

- (2) The CPU then uses the value of  $T_n$  to calculate an initial or "count from" counter setting of

$$(65535 - T_n + 56)$$

which it transmits to the counter on the 16-bit bidirectional bus.

- (3) The stop-on-overflow, start, and load-on-interrogation control gates are set (to logical 1) via the 8-bit control bus.
- (4) Receipt of an interrogation pulse from the receiver causes the counter to load the "count from" value calculated in (2), and the counter to start counting from this preset value.
- (5) When the counter reaches its maximum count (overflows) the counter stops counting (stop-on-overflow gate is reset - to logical 0) and "56  $\mu$ sec early pulses" are sent to the three (3) CAT reply generators, causing them to stop generating any further replies, and to the CPU. The 56  $\mu$ sec early pulse sent to the CPU is referred to as the "UIT interrupt pulse".
- (6) Steps (1) through (5) are repeated.

---

\* In its normal operating mode the UIT provides pulses timed with respect to ATRBS interrogations. However, during ARIES program checkout, troubleshooting, etc., the timer is used to provide precise time intervals relative to other reference pulses - hence the title "Universal".

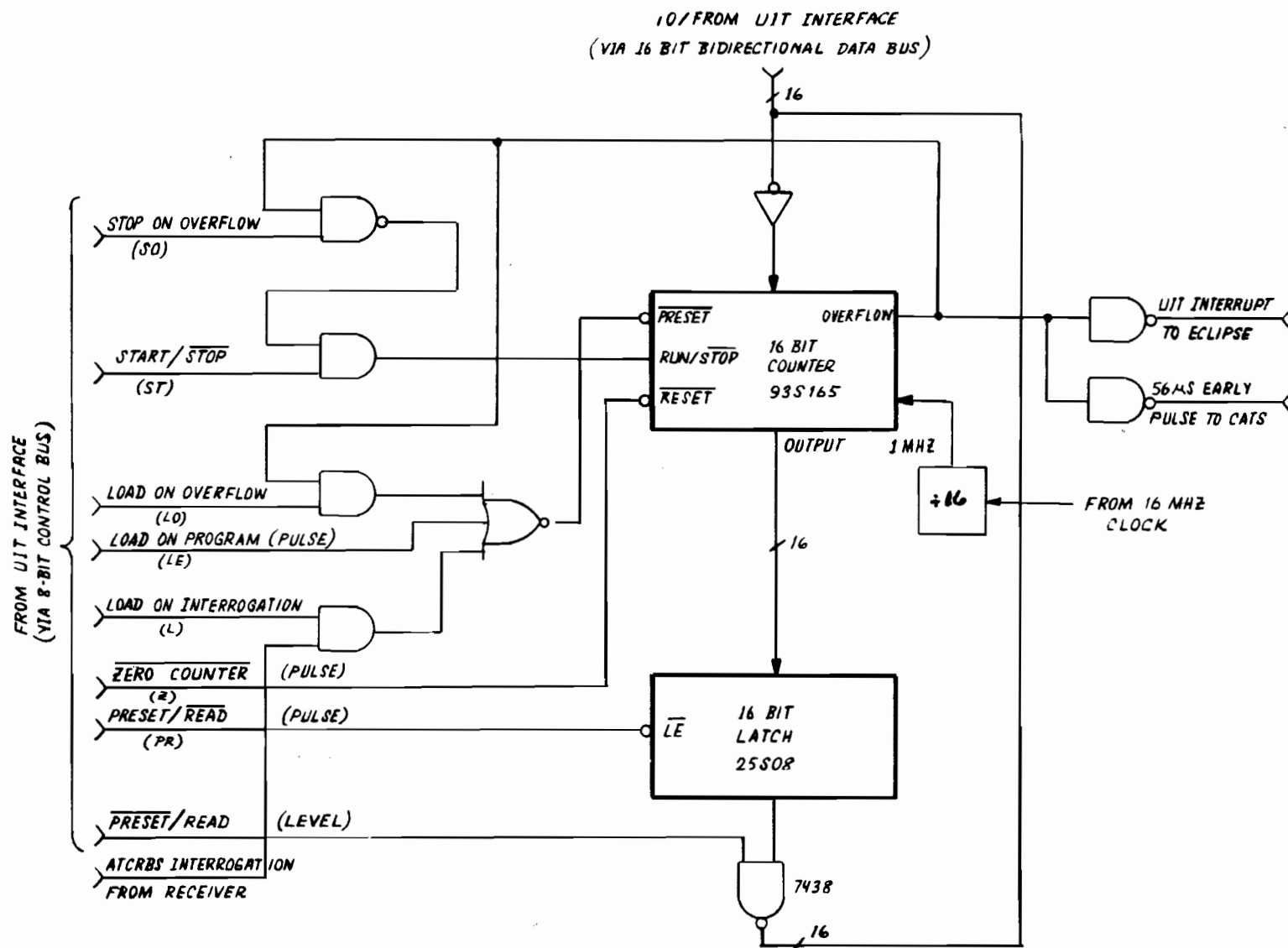


Fig. 2.2-2. Universal interval timer, block diagram.



TABLE 2.2-1

UNIVERSAL INTERVAL TIMER CONTROL BIT OPERATIONS

<u>Bit</u>	<u>Control Bit Title</u>	<u>Setting Bit Causes</u>
1.	Stop-on-overflow	16-bit counter to "freeze" when overflow occurs.
2.	Start/ $\overline{\text{stop}}$	16-bit counter to start.
3.	Load-on-overflow	16-bit counter to load in the preset value when overflow occurs.
4.	Load-on-program (pulse)	Preset value to be loaded in when this pulse occurs.
5.	Load-on-interrogation	Preset value to be loaded in when an ATCRBS or ATCRBS/All Call interrogation arrives.
6.	Zero counter (pulse)	Counter to be cleared.
7.	Preset/ $\overline{\text{read}}$ (pulse)	Counter to be sampled into the 16-bit latch.
8.	$\overline{\text{Preset/read}}$ (level)	The output drivers (7438's) to be enabled.

In normal operation, reading the content of the counter is not necessary. However, for diagnostic purposes the contents may be read by setting the preset/read bit and pulsing the present/read line. This operation will transfer the current value of the 16-bit counter into the 16 bit latch, and at the same time enable the output driver to output the value to the CPU.

Since the UIT interface data bus is bi-directional (used for input and output) it is important that the preset/read level bit not be set during the loading of the 16-bit counter (i.e., if the load-on-program, load-on-overflow, or load-on-interrogation gates are set) or the value loaded by the counter will be erroneous.

It is sometimes desirable to have the UIT generating CPU interrupts at constant intervals. This can be done by setting the load-on-overflow bit, and outputting the value  $(65536 - T)$  to the UIT interface data bus. To initialize this operation, the start bit is set and the load-on-program bit is pulsed. Thereafter, the UIT will generate interrupts at constant interval,  $T$ .

The UIT can be zeroed at any time by pulsing the zero counter bit.

#### 2.2.2.1 Universal Interval Timer Interface

The Universal Interval Timer (UIT) interface provides a means of controlling the operational characteristics of the UIT described in the Section on the UIT.

It consists of two data registers (see Fig. 2.2-3). The 7-bit C-register, which contains the control bits, controls the following seven modes of the UIT:

- (a). LOAD ON INTERROGATION
- (b). PRESET/READ
- (c). LOAD ON OVERFLOW
- (d). LOAD ON PROGRAM
- (e). ZERO
- (f). START/STOP
- (g). STOP ON OVERFLOW

The C-Register can be set by the CPU by executing a DOC instruction. Certain control-bits, (ZERO the UIT, PRESET/READ the UIT and LOAD UIT on PROGRAM) need to be pulsed in order to operate properly. These bits are set by the DOCP instruction. The other register is labeled as the A-Register. This 16-bit register contains the preset-value that will be loaded into the UIT 16-bit counter if the PRESET/READ bit and any one of the following three bits are set: LOAD ON INTERROGATION, LOAD ON OVERFLOW, LOAD ON PROGRAM. The

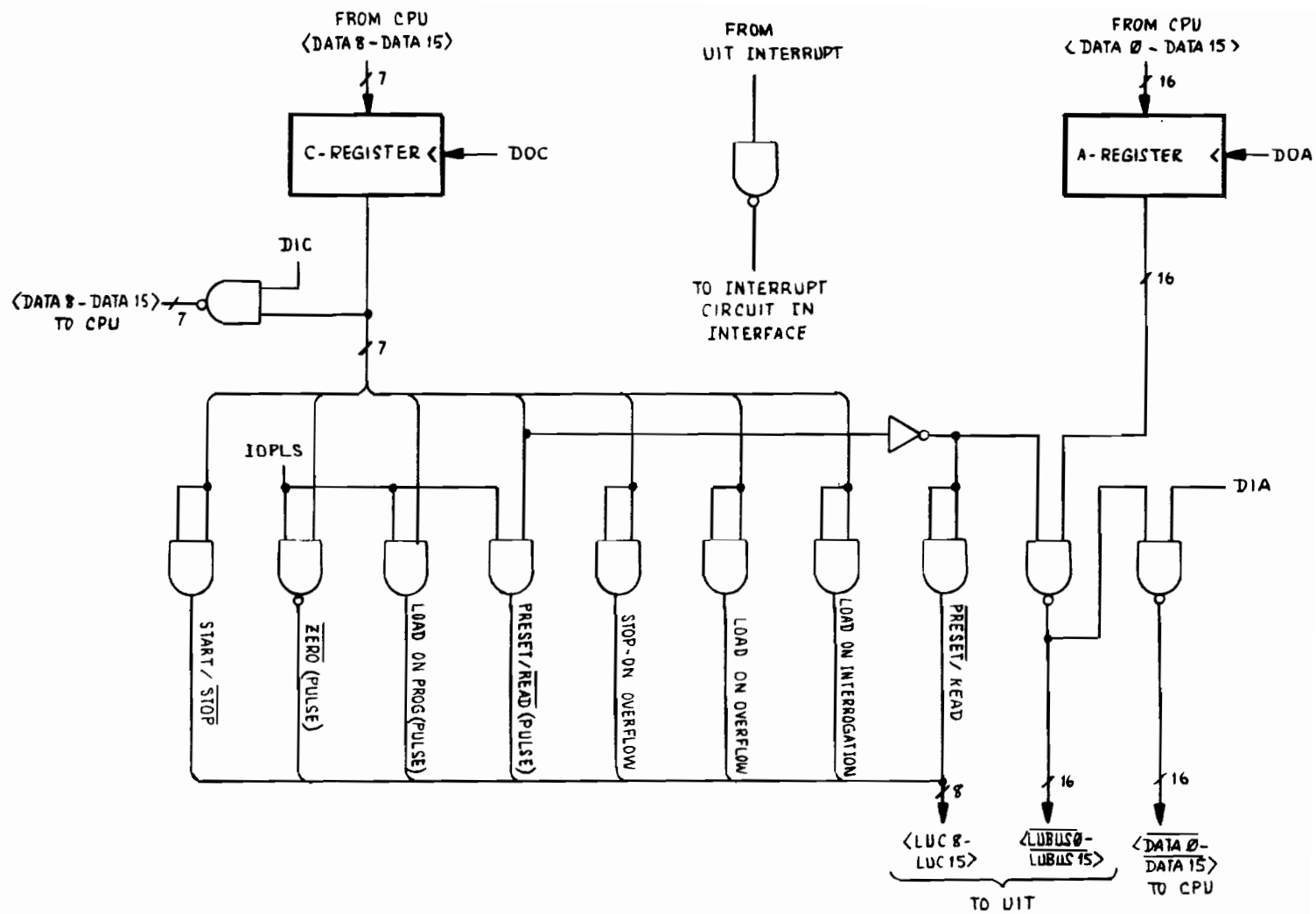


Fig. 2.2-3. Universal interval timer interface, block diagram.

contents of the A-Register can be set by the DOA instruction. For diagnostic purposes, the contents of the C-Register can be read by the CPU by executing a DIC instruction. Also when the Preset/Read bit is set and pulsed, a DIA instruction will return a sampled counter value from the UIT.

Interrupts generated by the UIT are handled by the Data General standard busy-done logic. (Readers not familiar with this protocol should refer to the D.G. Interface Designer's Reference Manual, Data General document number 015-000031).

## 2.3 Receiver

The receiver consists of the analog and digital circuits. The analog circuitry is shown in Fig. 2.3-1.

### 2.3.1 Analog Circuits

The receiver is fed from the ARIES RF port of the DABS sensor at 1030 MHz and at a level of -10/-1 dBm. A 20 dB directional coupler permits the ARIES Tester output to be fed-back to the ARIES receiver input in the self-test mode. After the signal is filtered by the pre-selector, it is then fed to the mixer. The local oscillator, operating at 970 MHz, provides mixer input via a 20 dB coupler. 60 MHz IF is then detected and amplified by the log amplifier. Output from the log amplifier drives both a threshold circuit and a surface acoustic wave demodulator providing PAM and DPSK outputs respectively (Fig. 2.3-2). These are fed to the digital circuits.

### 2.3.2 Digital Circuits

The uplink receiver digital circuitry is similar to that included in a DABS transponder receiver (Fig. 2.3-3). It consists of mode decoding logic, and, for DABS signals, a 24-bit parity decoder. Also included is an azimuth register which accepts azimuth information from the ACP Decoder, and a 20 bit range counter.

#### 2.3.2.1 Mode Decoding

The mode decoding circuitry determines the mode of each interrogation received. As shown in the simplified diagram of Fig. 2.3-4 it consists of a delay element constructed from a series of shift registers. The outputs from the delay element are tapped at 2, 8, 21, and 25  $\mu$ s corresponding to

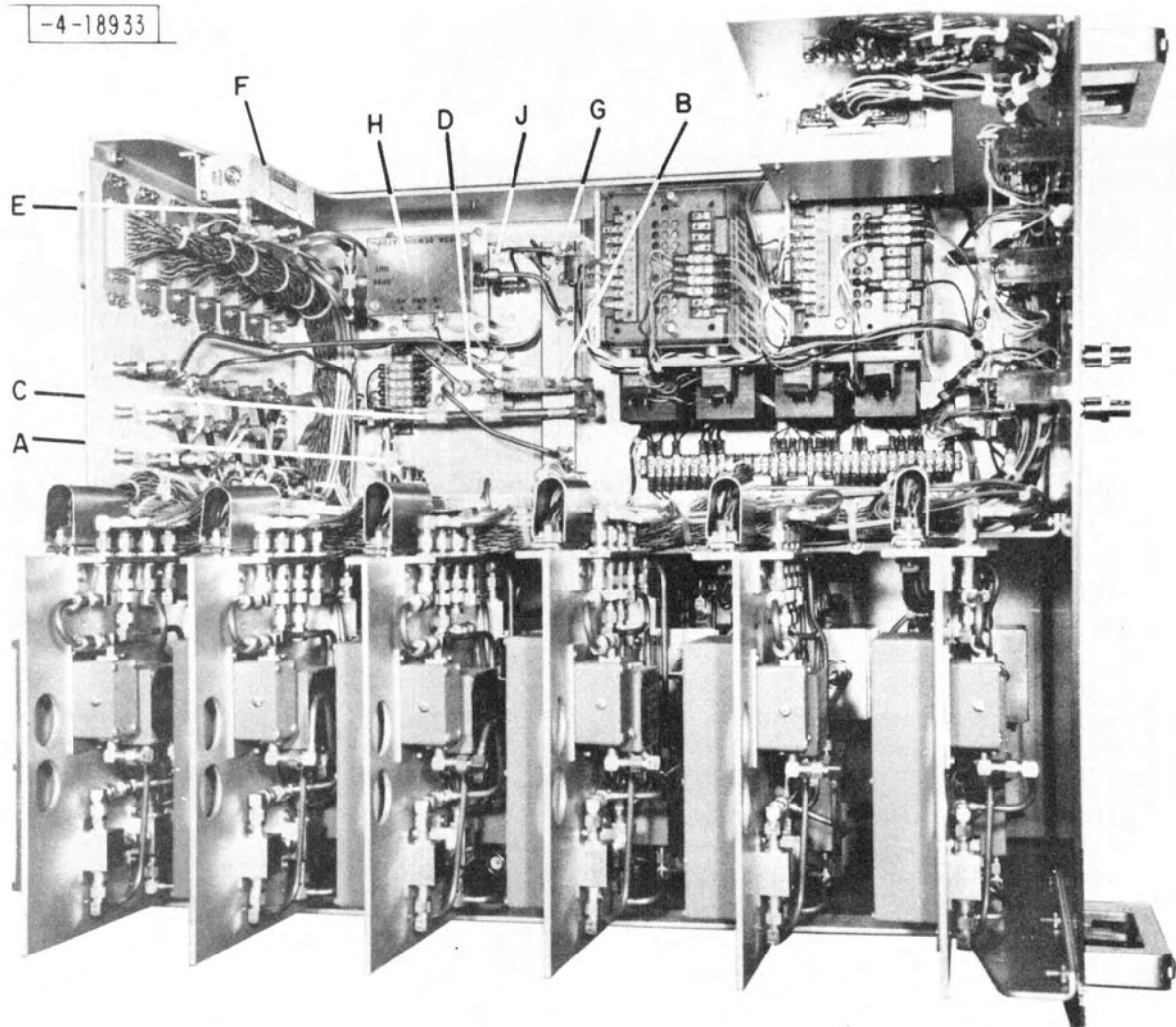


Fig. 2.3-1. Analog drawer (top view) showing receiver components.

ANALOG DRAWER (TOP VIEW) SHOWING RECEIVER COMPONENTS

(Refer to Receiver Section Schematic)

- A    30 dB Attenuator
- B    Directional Coupler
- C    1030 MHz Preselector
- D    Mixer
- E.   Directional Coupler
- F    Local Oscillator
- G    Log Amplifier
- H.   DPSK Demodulator
- J    Threshold Circuit

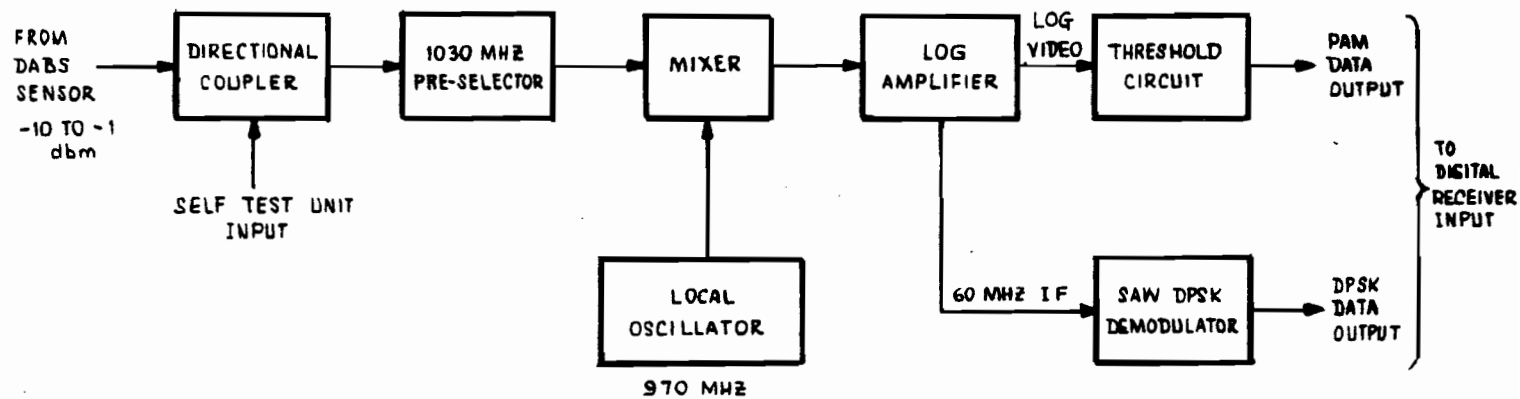


Fig. 2.3-2. Analog receiver, block diagram.

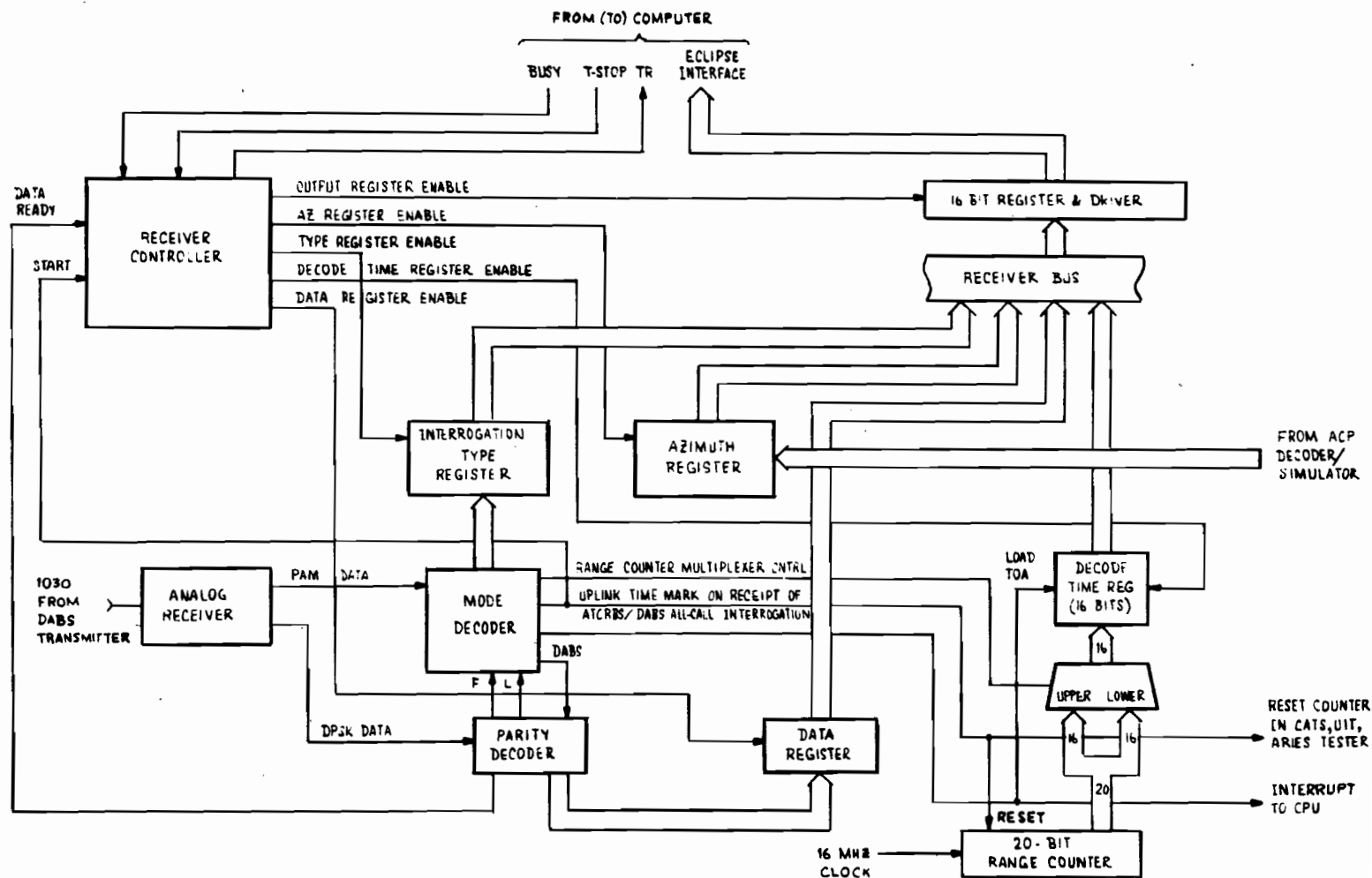


Fig. 2.3-3. Uplink receiver analog and digital circuits, block diagram.



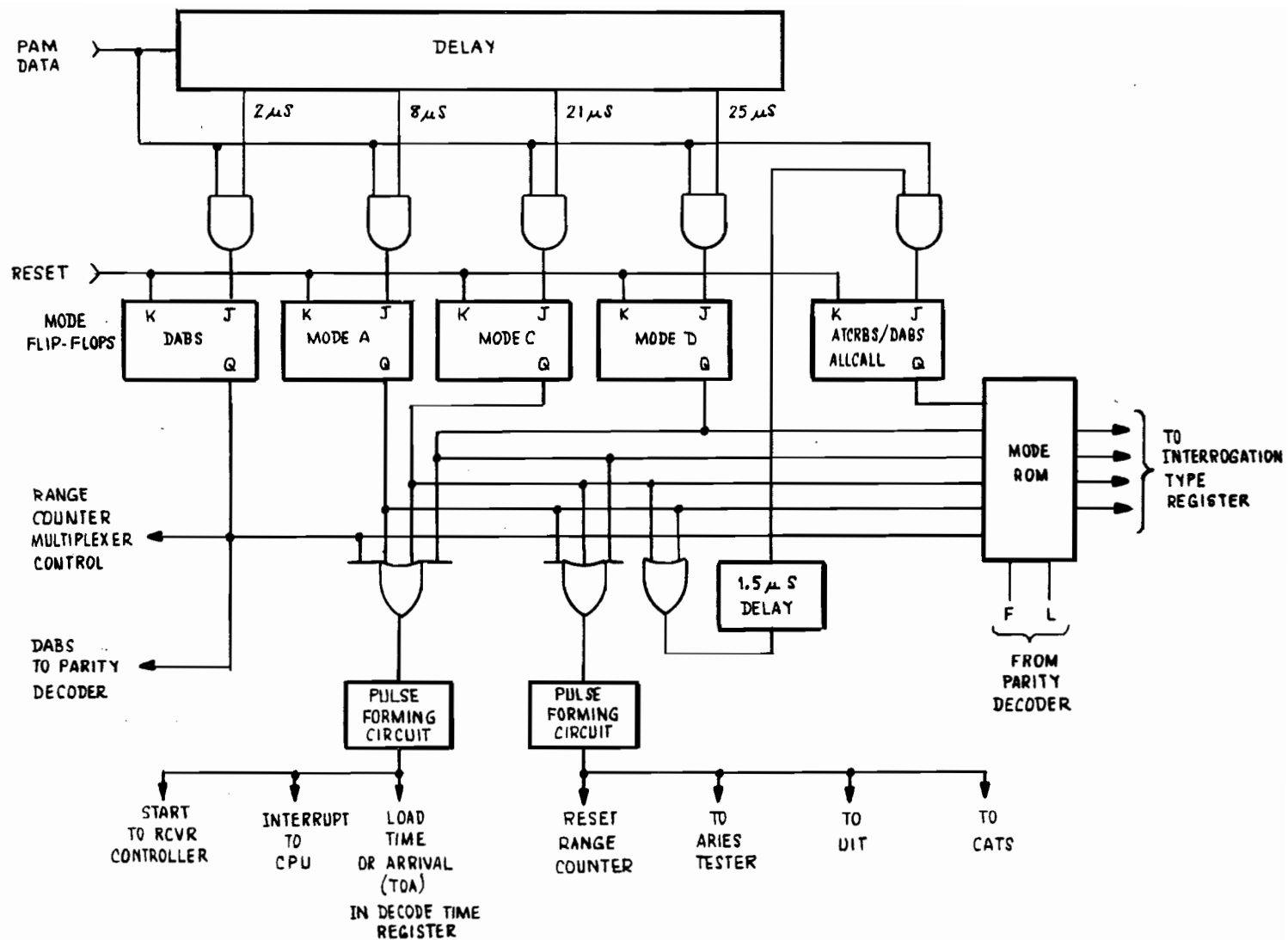


Fig. 2.3-4. Mode decoder, block diagram.

the separation between the  $P_1$  and  $P_2$  pulses for DABS and between the  $P_1$  and  $P_3$  (8, 21, 25  $\mu$ s) pulses<sup>1</sup> for ATCRBS. See Fig. 2.3-5. If a preamble is detected by the AND gate, the corresponding J-K flip-flop will be set. If the interrogation is an ATCRBS/DABS All-Call (Mode A or Mode C), the All-Call flip-flop will also be set. The setting of the mode flip-flop(s) allows the mode ROM to encode the interrogation type, and causes the transmission of the mode type to the mode register. Also, at each interrogation part of the 20 bit range word is transferred to the decode time register to determine time-of-arrival. If the interrogation is an ATCRBS All-Call (Mode A, Mode C or Mode D), the 16 most significant bits are transferred. After these bits are transferred the 20 bit range counter is reset. Each time a DABS interrogation is received, the contents of the 16 least significant bits are transferred, but the range counter is not reset. In this way the reply time for DABS interrogations is referenced to the last time-of-arrival of an All-Call interrogation. This allows 4096 unambiguous microseconds for determining the required reply time to a DABS interrogation. By shifting to the 16 most significant bits of the 20-bit range counter upon receipt of an All-Call interrogation, the ARIES unit is capable of handling, unambiguously, maximum intervals between All-Call interrogations of approximately 60 milliseconds (the normal period between All-Call interrogations is approximately 10 milliseconds).

The loading of time-of-arrival in the decode time register, and the control of range counter multiplexer selection are done by the mode decoding logic. Upon receipt of each interrogation, an interrupt pulse is generated to inform the computer of its arrival, and a 10 word interrogation data block is created in a buffer area of the CPU memory. This block contains the time-of-arrival, the antenna boresight azimuth at arrival, the mode, and (for discrete interrogations only) the data bits. (See section on Data Transfer). Upon receipt of each All-Call interrogation, a pulse is sent to the ARIES tester, the UIT, and the three CAT's to reset their range counters.

The azimuth data from the ACP decoder is also sampled in the azimuth register upon arrival of each interrogation.

#### 2.3.2.2 Parity Decoder\*

In the case of discrete interrogations, the data bits are assembled and the error detection encoding from the DABS address field is removed. This is done by the parity decoding circuitry.

---

\* (For more detail, see "DABS Uplink Coding" by J.T. Barrows, Report No. FAA-RD-75-62, (Lincoln Laboratory, Report No. ATC-49).

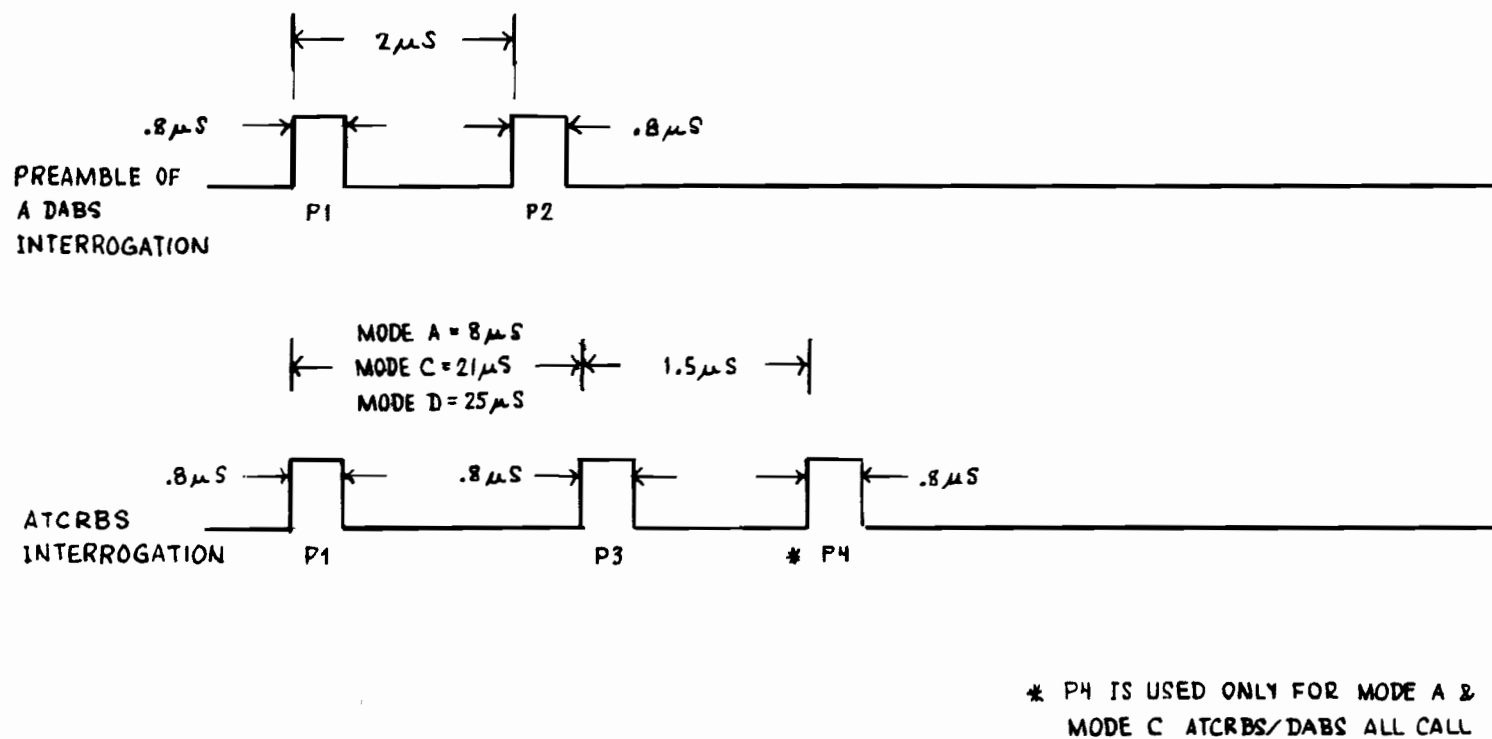


Fig. 2.3-5. Mode decoder waveforms.

The parity decoder (see Fig. 2.3-6) consists of a 24 bit shift register and a sum modulo-2 network connected to the shift register at appropriate points corresponding to the 24-degree polynomial generator. Both the DABS address and parity are decoded. The shift register is first cleared. The received DPSK data is then fed into the shift register. After 32 shifts (DABS All-Call, or surveillance) or 88 shifts (Comm-A), the shift register will contain 24 parity bits. These 24 parity bits are then summed bit by bit, modulo-2, with the remaining 24 bits from the input DPSK data stream to produce the DABS address.

When the mode decoder declares a DABS interrogation, the controller in the parity decoder is reset and ready to begin a decoding sequence. Before the data bits arrive, multiplexer M is selected to allow the DPSK data to be input to the 16-bit shift register without modification. At the same time, the DPSK data is also fed to the parity decoder. As soon as the DPSK data sync phase-reversal is detected, the sample clock will be activated and begin to sample the data at .25  $\mu$ s intervals. After the first 2 data bits (F, L bits) are sampled, the type and length of the DABS interrogation can be determined. These 2 bits allow the controller to select multiplexer M at an appropriate time, so that the last 24 data block bits may be sent to the 16 bit register from the parity decoder. The last 24 bits are the decoded DABS address.

The contents of the 16-bit shift register is parallel loaded into a buffer after each 16 samples, so that the entire 56-bit or 112-bit data block can be transferred to the CPU, 16 bits at a time.

#### 2.3.2.3 Data Transfer Sequence

Since it is not possible to anticipate the time of arrival of DABS interrogations, it is crucial that the received data to be transferred to the computer upon receipt of a DABS interrogation in the minimum possible time. This is accomplished by setting up the interface to the receiver so that it is prepared for a direct memory access (DMA) data transfer immediately following the decoding of a valid DABS interrogation. A similar data transfer technique is used for ATRBS interrogations.

Upon receipt of each interrogation, an interrupt is generated to inform the computer of its arrival. This is done by the mode decoding logic as described previously.

Following the interrupt, a 10 word interrogation block is transferred to memory. This block contains the time of arrival (TOA), the antenna bore-sight azimuth at arrival, the mode, and for (discrete interrogations only) the data bits.

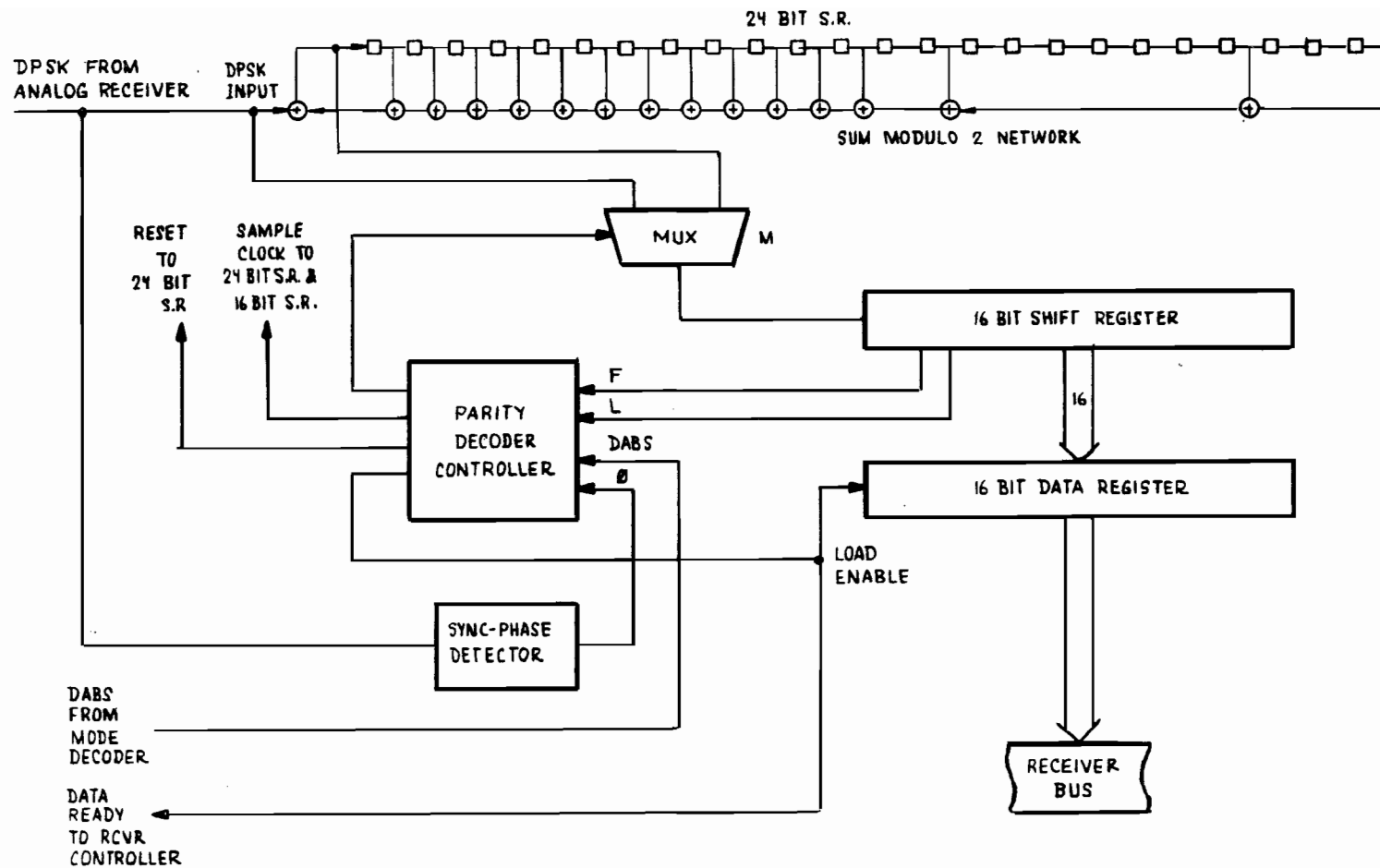


Fig. 2.3-6. Parity decoder, block diagram.

Details, including field definitions, of the above data formats are given in Volume 3 of this report.

The sequence of data transfer is controlled by the receiver controller. A state diagram for the controller is shown in Fig. 2.3-7. Note that in State 10, the controller will loop on the status of the busy-bit coming from the busy-done logic of the receiver interface; thus the CPU has a direct control over the operation of the digital receiver.

Once the busy bit is set, the controller will go to State 0 and wait for the arrival of the next interrogation. Upon arrival, its TOA, azimuth and type will be sent by the controller as shown in States 1, 3, and 5.

The data rate for the DABS uplink interrogation is 4 MBPS. Therefore, in order for the 16-bit shift register in the parity decoder to assemble a 16 bit word, 4  $\mu$ sec are required. This is the purpose of State 6. When the data are ready from the 16-bit shift register, they are transferred to the receiver interface. States 6, 7, 8, and 9 will be repeated until the end of the data block. Since ATCRBS interrogations include no data field; all zero's will be transferred as data.

The receiver is then reset and the above process repeated for the next interrogation.

Each of the 10 words transferred involves two "hand-shaking" signals between the receiver and its interface. These two signals are the transfer request (TR) from the receiver, and transfer stop (TSTOP) from the interface. Each time, the receiver is ready to send a word, a TR signal is sent to the interface, and at the same time the appropriate register is enabled (e.g., decode time register), allowing it to output to the receiver interface buffer. If the buffer can accept more data words, TSTOP will be set to zero; otherwise TSTOP will be set to one, causing the controller to enter a wait state (States 2, 4, and 7). It will remain in this state until the buffer can accept another word. The controller is shown in Fig. 2.3-3.

The interrogation type, azimuth, decode time and data registers are of tri-state type. They are all tied together to form the internal receiver bus. When each register is enabled by the receiver's controller, its content will be transferred to the 16-bit register and driver, and subsequently to the CPU.

### 2.3.3 Receiver Interface

The Receiver Interface uses the direct memory access (DMA) or data channel control technique to transfer data from the receiver to the CPU. This method has a speed advantage over the conventional programmed I/O

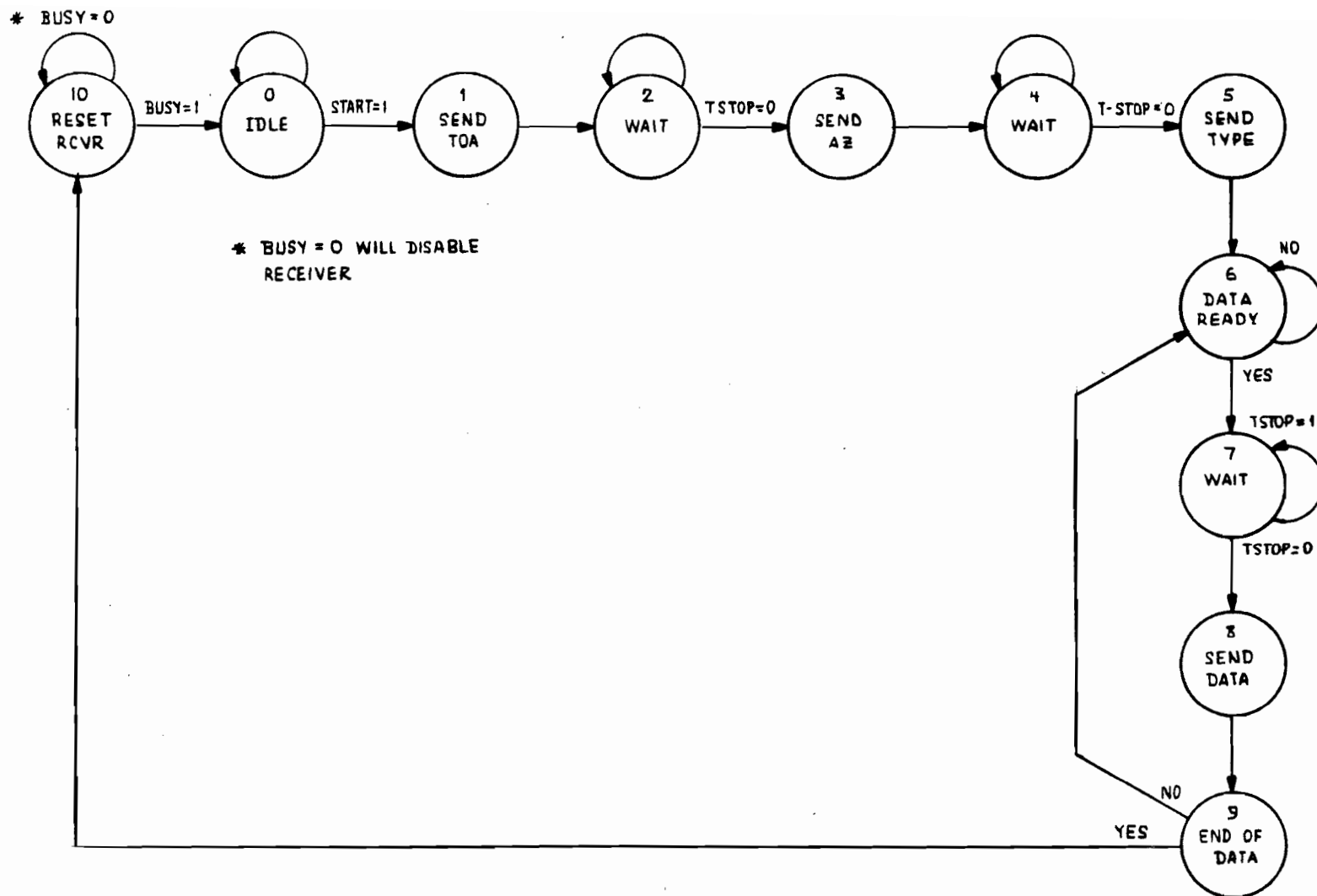


Fig. 2.3-7. Data transfer state diagram.

technique since it is software independent once the initial memory address has been specified (via programmed I/O). The protocols used are similar to all Data General DMA controllers\*.

A block diagram of the Receiver Interface is shown in Fig. 2.3-8. It consists of a register file and control circuit, data channel control logic and a memory address counter.

Each time the register file and control circuit is fed a word from the receiver, the word available bit will be set, causing the data channel control logic to generate a data channel request (DCHR) to the CPU. In return, the CPU issues a DCHA and DCHI pulse to the interface. The DCHA pulse has two functions: it allows the memory address counter circuit to output the address, and enables a control signal DCHMO that specifies the type of transfer (input or output) to the CPU. The DCHI pulse which occurs immediately after DCHA allows the register file and control circuit to output the data word from the interface to the CPU.

#### 2.3.3.1 Data Channel Control Logic

DCHA SYNC EARLY, DCHA SYNC LATE,  $\overline{\text{DCHA.SEL}}$ , and  $\overline{\text{DCHI.SEL}}$ , DCHI.SYNC are derivatives of DCHA and DCHI respectively. They are control signals for the memory address counter, the register file circuit, and the data control logic.

Operation of the data channel control logic may be understood by considering a DMA sequence as follows (refer to Fig. 2.3-9).

When an interrogation is received by the receiver, an interrupt pulse is generated and sent to the interface. This pulse sets the done bit in the interface thereby informing the CPU of the arrival of an interrogation, and initializes the interface for the up-coming DMA transfer. When a word from the receiver arrives, the interface DCH SYNC flip-flop in the data channel control logic will be set. The state of DCH SYNC is then clocked into the data channel request flip-flop (DCH REQ) by the request enable line RQENB, causing a data channel request (DCHR) to be setn to the CPU.

When the CPU sees  $\overline{\text{DCHR}}$  asserted, it pauses at the next convenient point in its operation, and issues an acknowledge signal, DCHA. Since this signal goes to all devices, the receiver interface will respond to it only if other devices having higher data channel priority are not currently requesting DMA (i.e., data channel priority in (DCHP-IN) to the receiver interface must be low).

---

\* The reader not familiar with this protocol should refer to the Data General Interface Designer's Manual, DG015-000031).



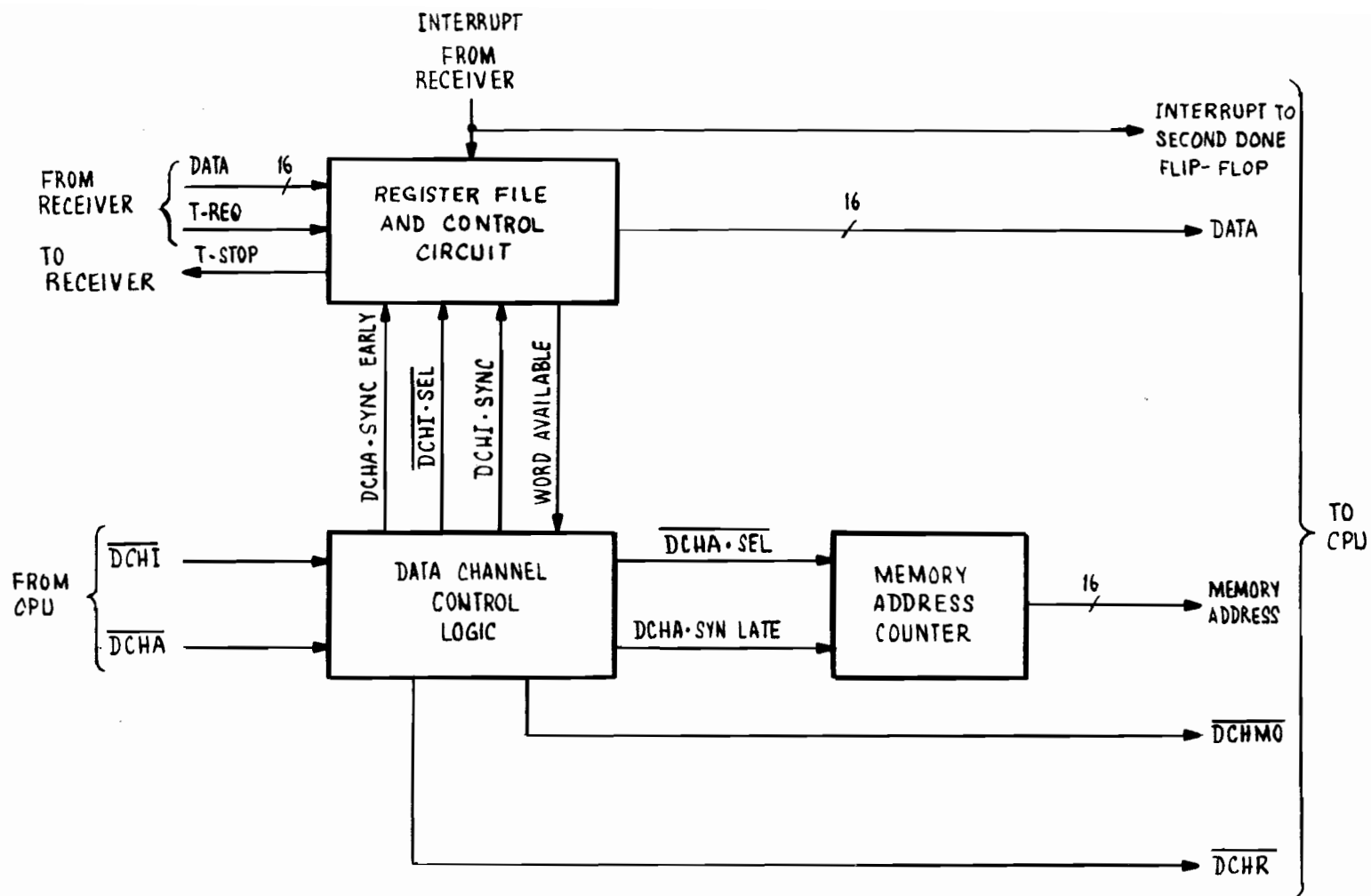


Fig. 2.3-8. Receiver interface, block diagram.

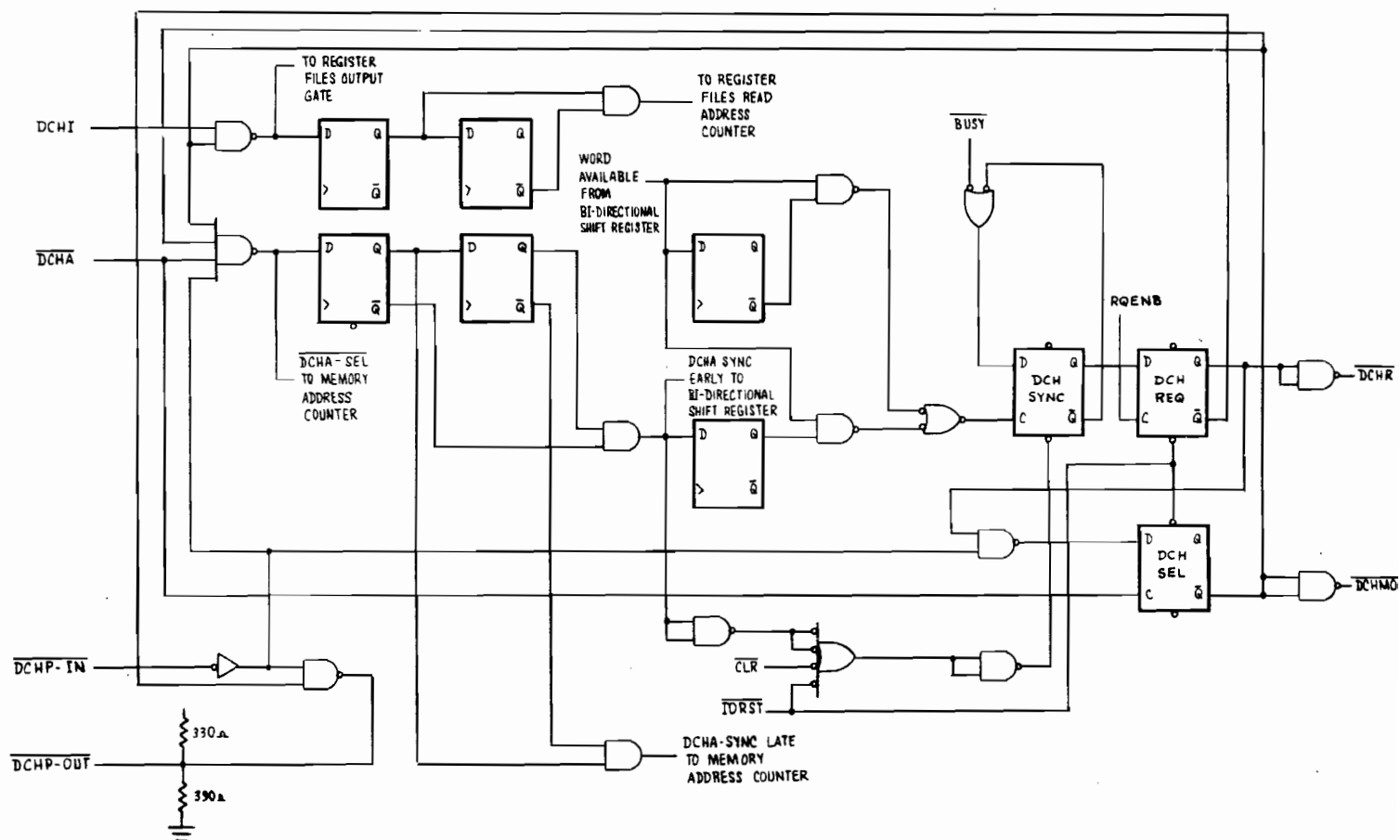


Fig. 2.3-9. Data channel control network.

The receiver interface then outputs two pieces of information to the CPU: type of transfer flag, DCHMO (from the data channel control logic) and the address (from the memory address counter) of the memory location that will be accessed. (Logical low for DCHMO signifies an input operation to the memory. This is the only type of transfer which the receiver can provide).

Following DCHA, the CPU issues a data channel input control signal DCHI to the receiver interface, causing the interface to output a data word (from a register file) to the addressed memory location.

If another word is received from the receiver before the end of the above operation, and there is no other higher priority device requesting DMA, the DCHA, DCHI sequence will be repeated. This burst mode operation will repeat until all available words from the receiver interface are transferred to the memory. The interface then removes the DCHR signal, thus suspending further receiver DMA operation until another word is received.

#### 2.3.3.2 Memory Address Counter

Each time the CPU issues a DCHA, it expects from the interface two pieces of information: the type of transfer and the memory address. The memory address counter provides the latter.

Before DMA is attempted, the CPU should initialize the counter with a starting address. This is done by means of a DOA instruction. Since a 640 word buffer is required, the address counter is capable of counting from the initial address A to A + 639. When A + 639 is reached, the counter will continue at A. Fig. 2.3-10 shows the memory address counter circuitry used in the receiver interface. Note that DCHA.SEL and DCHA.SYNC.LATE from the data channel control are used to output the memory address to the CPU and to increment the counter respectively; also when preset of the counter is logical low, Q will be preset on the next rising edge of the clock (i.e., DOA or DCHA-Sync-Late).

#### 2.3.3.3 Register File and Control

Assuming no other device in the system is currently requesting DMA, the CPU can accept data from the receiver interface at a maximum rate of 800 ns per word. This is not fast enough for the first three words which are transferred at a rate of 750 ns/word from the receiver to the interface. For this reason, a register file is provided in the interface. It can stack up to 4 words from the receiver, providing the DMA channel with ample time to input the data. After the first three words, the remaining seven words from the receiver are transferred at a slower rate of 4  $\mu$ sec/word, so that in this case the timing constraint is not as critical.

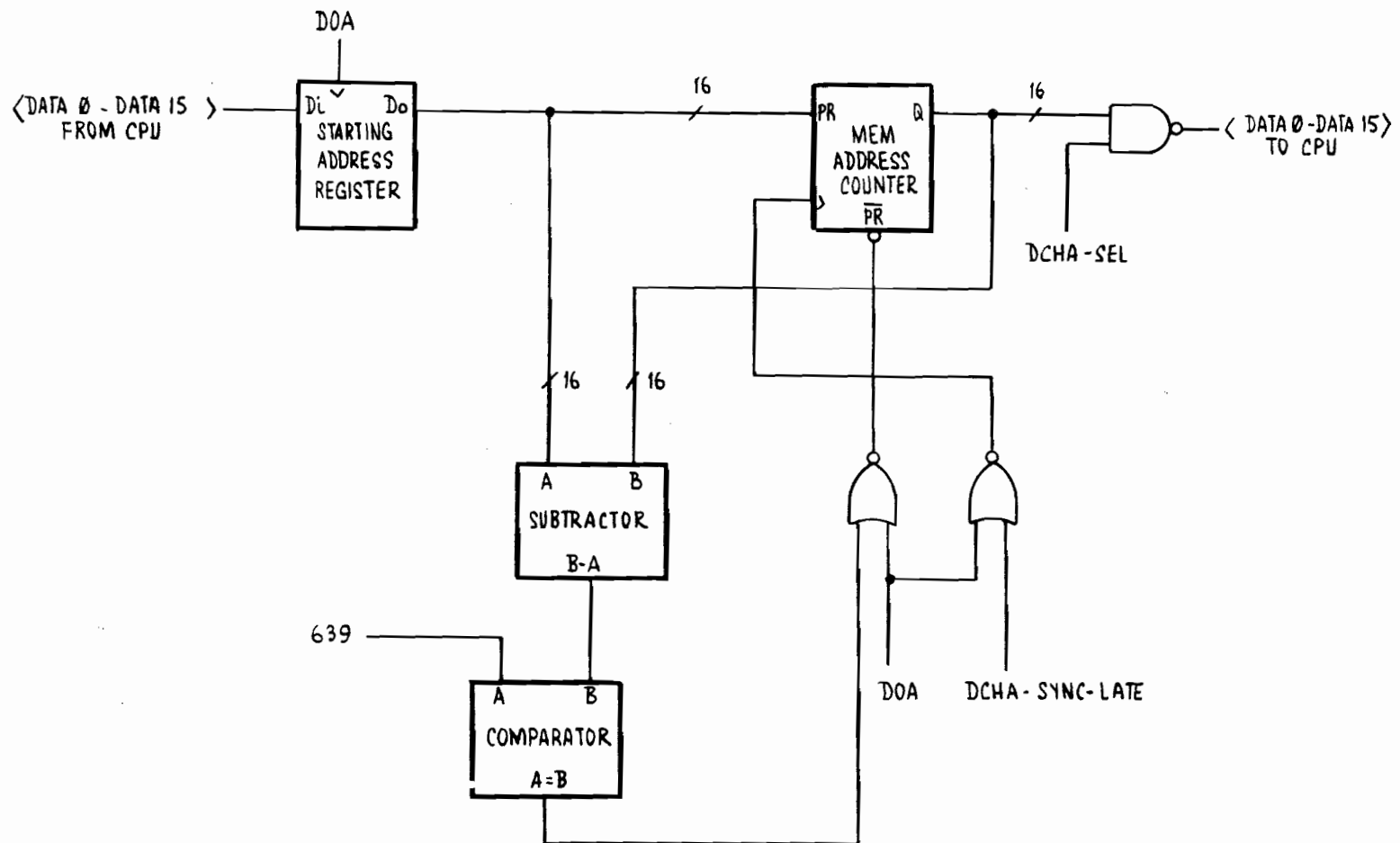


Fig. 2.3-10. Memory address counter, block diagram.

The register file and control circuits consist of a read address counter, a write address counter, an interrogation counter, a 4 bit bi-directional shift register, and a 16 bit x 4 word register file. The circuits are shown in Fig. 2.3-11.

After a word is output to the interface from the receiver a transfer request (T-request) is also sent. The purpose of this pulse is to write the data word into the register file; and after the data is written, to cause the write address counter to increment and a "one" to be right shifted into the bi-directional shift register. This shift register is a status indicator, with the leftmost bit connected as the word available bit, and the second right-most bit connected as the transfer stop bit (T-STOP).

Upon receipt of an interrupt from the receiver, indicating an arrival of an interrogation, the shift register will be cleared to zero. Each time a word is stored in the register files, a "one" will be right shifted into the shift register setting the word available bit. If a word is taken out of the files, a "zero" will be left shifted into the register.

The setting of the word available bit will cause the data channel control logic to generate a DMA request, and a normal sequence of data transfer will follow once the channel priority is granted.

If four words are stored in the register file before a data channel transfer is granted, the transfer stop bit (T-stop) will be set, causing the receiver to stop transferring any further words.

When a word is transferred to the CPU, the read address counter for the register file will be incremented, and a "zero" left shifted into the bi-directional shift register. This is done by the DCHI SYNC and DCHA SYNC pulses respectively.

The read and write address counters are zeroed by the interrogation interrupt.

#### 2.3.3.4 Interrogation Counter

The purpose of the interrogation counter is to maintain a count of interrogation data blocks that have been completed. It is a 6-bit up/down counter, and can be incremented at the end of each 10 word transfer, and decremented by the CPU via a DOC instruction after an interrogation block is processed. The counter can be read by the CPU by means of a DIC instruction.

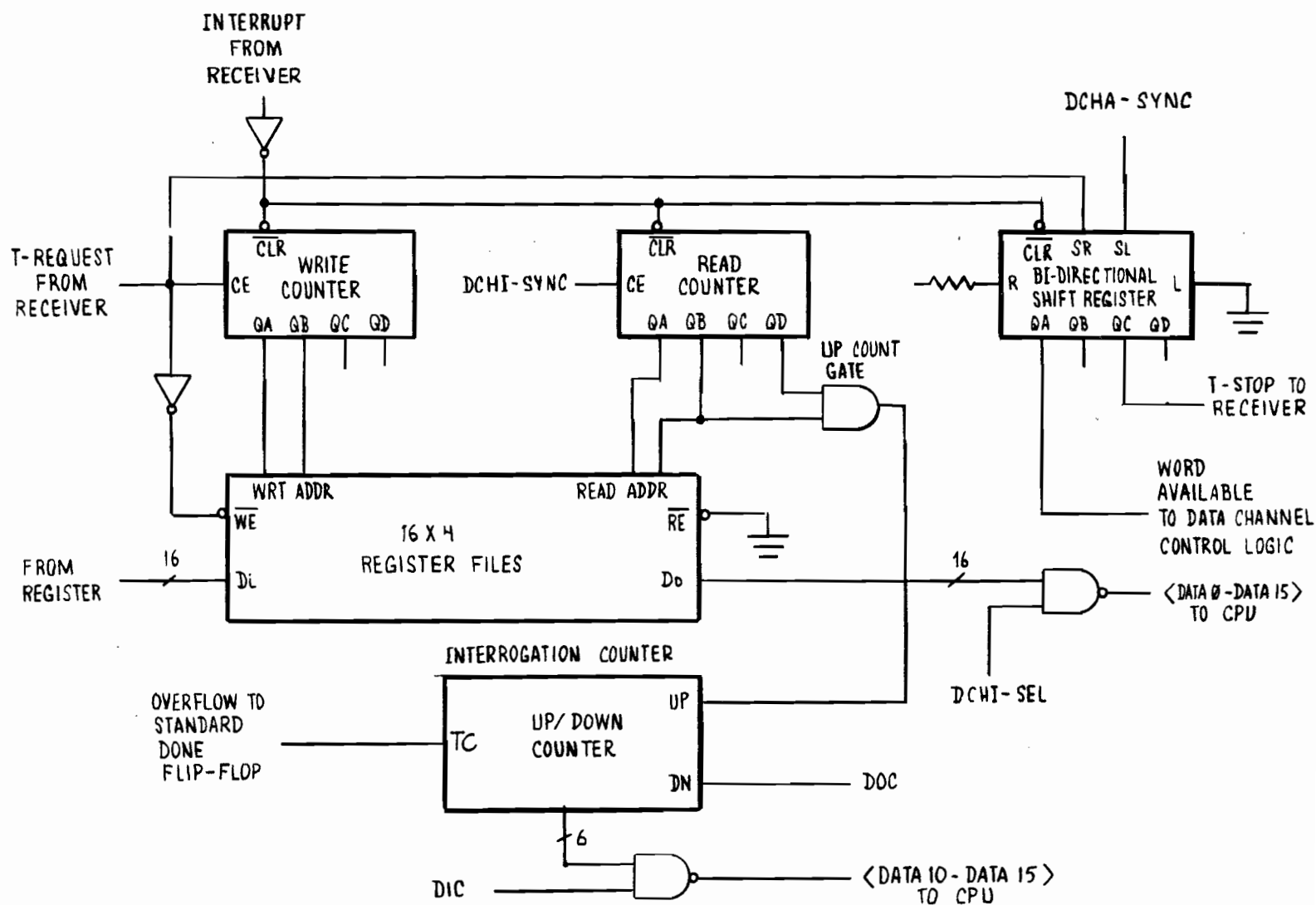


Fig. 2.3-11. Register file and control circuits, block diagram.

If this counter overflows, indicating 64 unprocessed interrogations (i.e., buffer overflow condition) an interrupt will be generated and sent to the CPU.

#### 2.3.3.5 Interrupt Logic

The interrupt logic for this interface is a modification of the Data General standard. Two DONE flip-flops are used to indicate different conditions. The standard DONE, which is tested by the I/O SKIP instruction, is set only upon buffer overflow. As in the Data General standard, the BUSY flip-flop is simultaneously reset, disabling the interface. This occurrence should be treated as an error condition.

The second DONE flip-flop cannot be tested by the I/O SKIP, but does cause an interrupt to be generated. The BUSY flip-flop is not reset. This is used to indicate the arrival of an interrogation. When either of the DONE flip-flops is set, an interrupt is generated, and the INTA instruction will indicate that the interrupt is from the receiver. The I/O SKIP instruction can be used to distinguish the two cases by testing the standard DONE flip-flop. See Fig. 2.3-12.

#### 2.4 Reply and Fruit Generation

Controlled replies, both DABS and ATCRBS, and fruit replies (only ATCRBS) are generated by two groups of components referred to as the Controlled Reply Generator (CRG) and the Fruit Generator (FG) (see Fig. 2.1-1). Each group is under the control of the CPU via its own interface, and each uses its own microprocessor for routing and constructing the replies. The CRG receives reply data from the CPU and the FG from the Random Process Generator. Each directs, via microprocessor control, its set of three ARIES target generators (controlled ARIES Targets (CAT's) or Fruit ARIES Targets (FATS), to create properly formatted and sequenced modulation control signals. These signals modulate 60 MHz carriers generated in the IF Unit of each CAT or FAT. The modulated outputs of all six ARIES targets are then combined in the IF Combiner and sent to the sensor.

In the descriptions which follow the details of operation of the component group referred to as the Controlled Reply Generator (2.4.1) precede the details of operation of the component group referred to as the Fruit Generator (2.4.2). However, each group includes a "controller" and each group includes three "reply generators", the operation of which is very similar. In many cases they are identical except for the "controlled reply" and "fruit" modifying terminology or stored control programs. The hardware components of the controlled reply and fruit controller microprocessors are identical and the hardware comprising the controlled reply and fruit reply generators are identical. Descriptions of the identical modules are not repeated but are given once under the first component group addressed, i.e., the Controller Reply Generator, and differences only in Section 2.4.2, Fruit Generator.

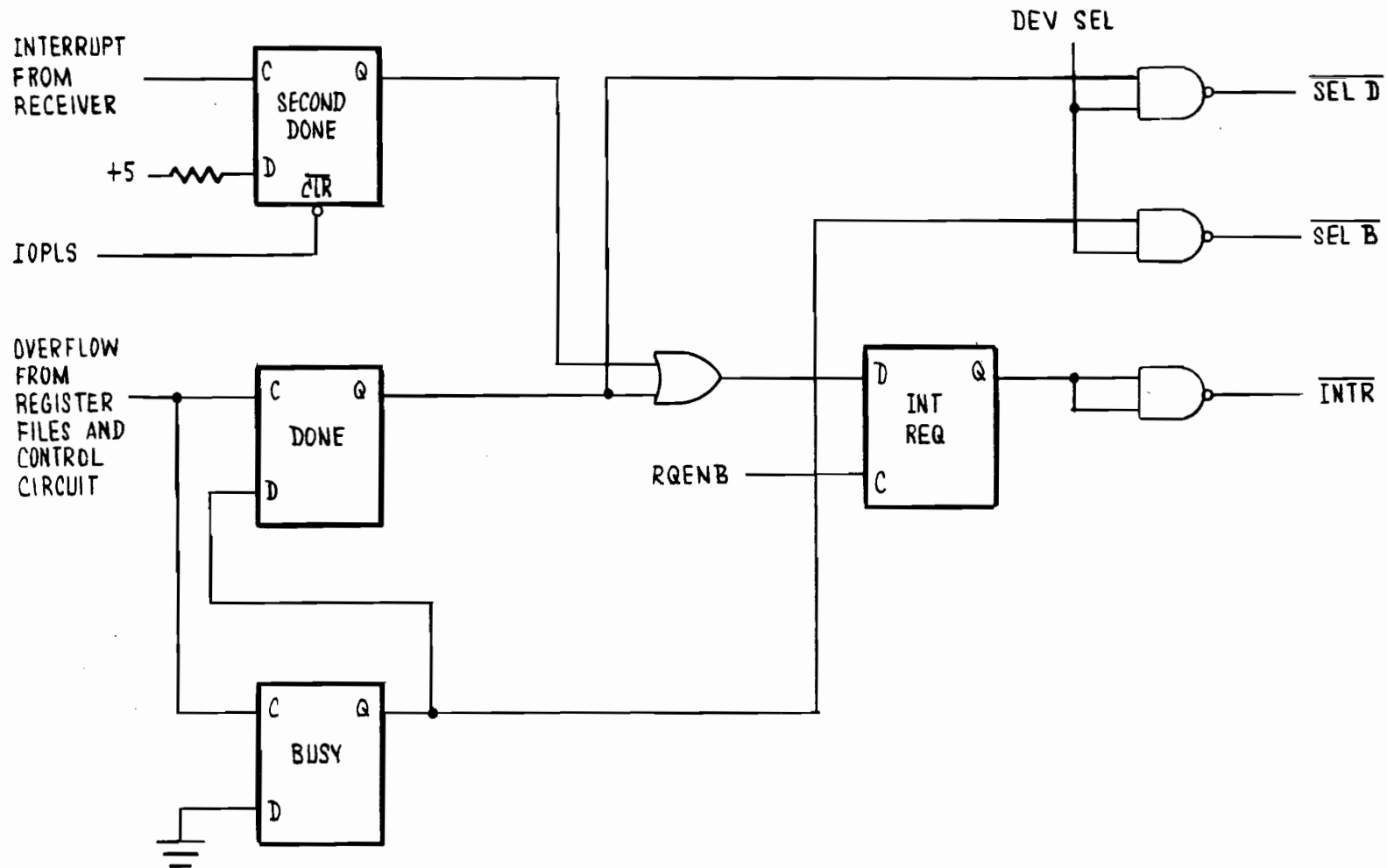


Fig. 2.3-12. Modified data general interrupt logic, block diagram.



#### 2.4.1 Controlled Reply Generator (CRG)

The Controlled Reply Generator consists of the CRG Controller, three controlled ARIES Targets (CAT's) and the CRG Interface.

##### 2.4.1.1 Controlled Reply Controller (CRC)

The primary function of the CRC, a high speed microprocessor (identical to the Fruit Reply Controller microprocessor), is to determine which CAT units are available for operation, and to transfer blocks of data from the CRG Interface to the available CATs. The microprocessor maintains a large portion of its control program in a programmable read-only memory (PROM), i.e., is microprogrammed. This permits higher speed operation and the possibility of changing the microprocessor instruction set (by altering the PROM) without modifying hardware.

Understanding of the operation of the CRC is best understood by examining its block diagram as shown in Fig. 2.4-1. There are six functionally related sub-blocks; the (1) microprocessor, (2) branch control logic, (3) program control logic, (4) microprogram storage and pipeline register, (5) input circuits, and (6) output circuits. Description of the operations of each of these blocks are given following a description of the controller overall architecture.

##### 2.4.1.1.1 Architecture

The Controlled Reply Controller consists of a number of 2900 bipolar microprocessor elements: the 2901 4-bit cascable microprocessor, the 2902 look-ahead carry generator, the 2909 4-bit cascable program sequencer, and related supporting circuit elements.

Four 2901's were used to form a 16-bit word microprocessor, with a worst case cycle time of about 200 ns. These LSI chips consist of a 16 word by 4-bit two port RAM, a high speed ALU, and associated shifting, decoding, and multiplexing circuitry. Next address selection and program control use three 2909's and the branch control logic. Together they can point to any one of 512 locations in the PROM. The input circuit enables the 2901's to multiplex data from the D-Bus or PROM. And the output circuit allows the output data from the 2901's to be placed on either the Y-Bus or the C-Bus.

The program for the controller is stored in five 512 word x 8 bit programmable read only memories (INTEL3624). Each 40 bit micro-instruction contains all control signal for the 2901's and 2909's. In the branch and load immediate instructions, the ROM also contains the branch address and load immediate constant, respectively.

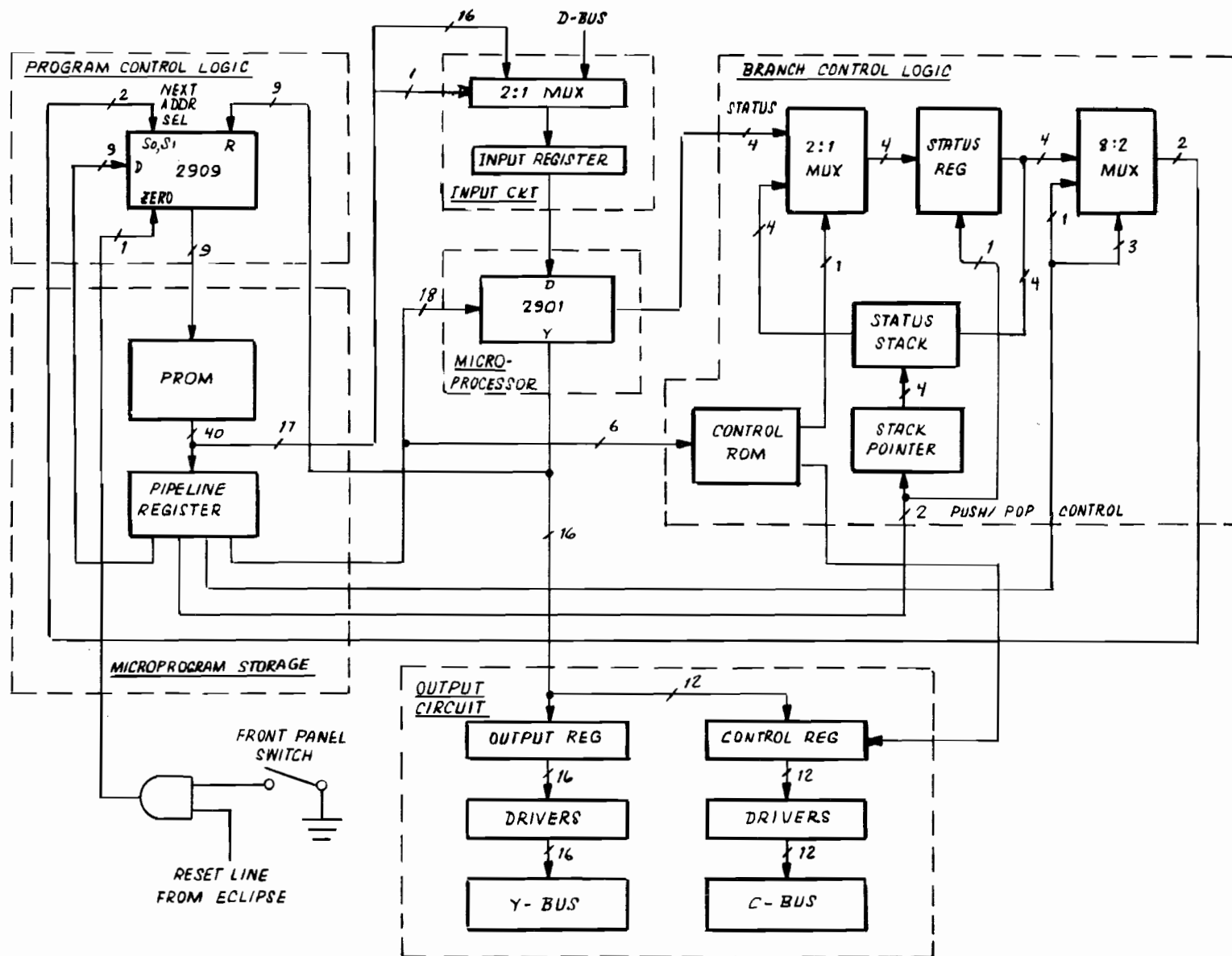


Fig. 2.4-1. Controlled reply controller, block diagram.

The output of the read only memory is buffered by a register known as the pipeline-register. See Fig. 2.4-2. This pipeline register permits the microinstruction fetch to occur in parallel with the data operation, effectively doubling the clock frequency. While the 2901's are executing the current instruction, the next address selected by the Program Control Logic is sent to the ROM, so that the next instruction will be strobed into the pipeline register beginning on the next clock edge.

During the execution of the Load Immediate instruction, the 16 bit constant must be available from the input register. To accomplish this, the constant is loaded into the input register at the same time that the Load Immediate instruction is strobed into the pipeline register. A control bit is programmed in the ROM to allow the input register to receive data from the memory.

#### 2.4.1.1.2 Program Control Logic

The program control logic (see Fig. 2.4-3) consists of three cascaded 2909's. This provides an addressing capability of 4096 memory locations (only 9 bits were used in the design). As shown in Fig. 2.4-4 the 2909 contains a four-input multiplexer to select one of four address sources. These sources are: 1) direct, 2) address register, 3) stack register, and 4) program counter. To select one of these sources,  $S_1 S_0$  must be coded as follows:

$S_1$	$S_0$	Source
0	0	Program counter
0	1	Address register
1	0	Stack register
1	1	Direct input

$S_0 S_1$  logic code is generated by the branch control logic (to be explained in Section 2.4.1.1.3).

The following explanation treats the three cascaded 2909's together. The address register consists of nine D-type flip-flops. Each pin RE is grounded, so these registers are always enabled, allowing the data from the nine least-significant-bits of the Y-bus to latch in on every clock cycle. This allows the microprocessor to execute a computed GO TO instruction, (i.e., address of the next instruction set by the result of the current instruction).

The direct input is a 9-bit field of inputs to the multiplexer. These nine-bits are programmable to provide the address during the execution of a BRANCH instruction.

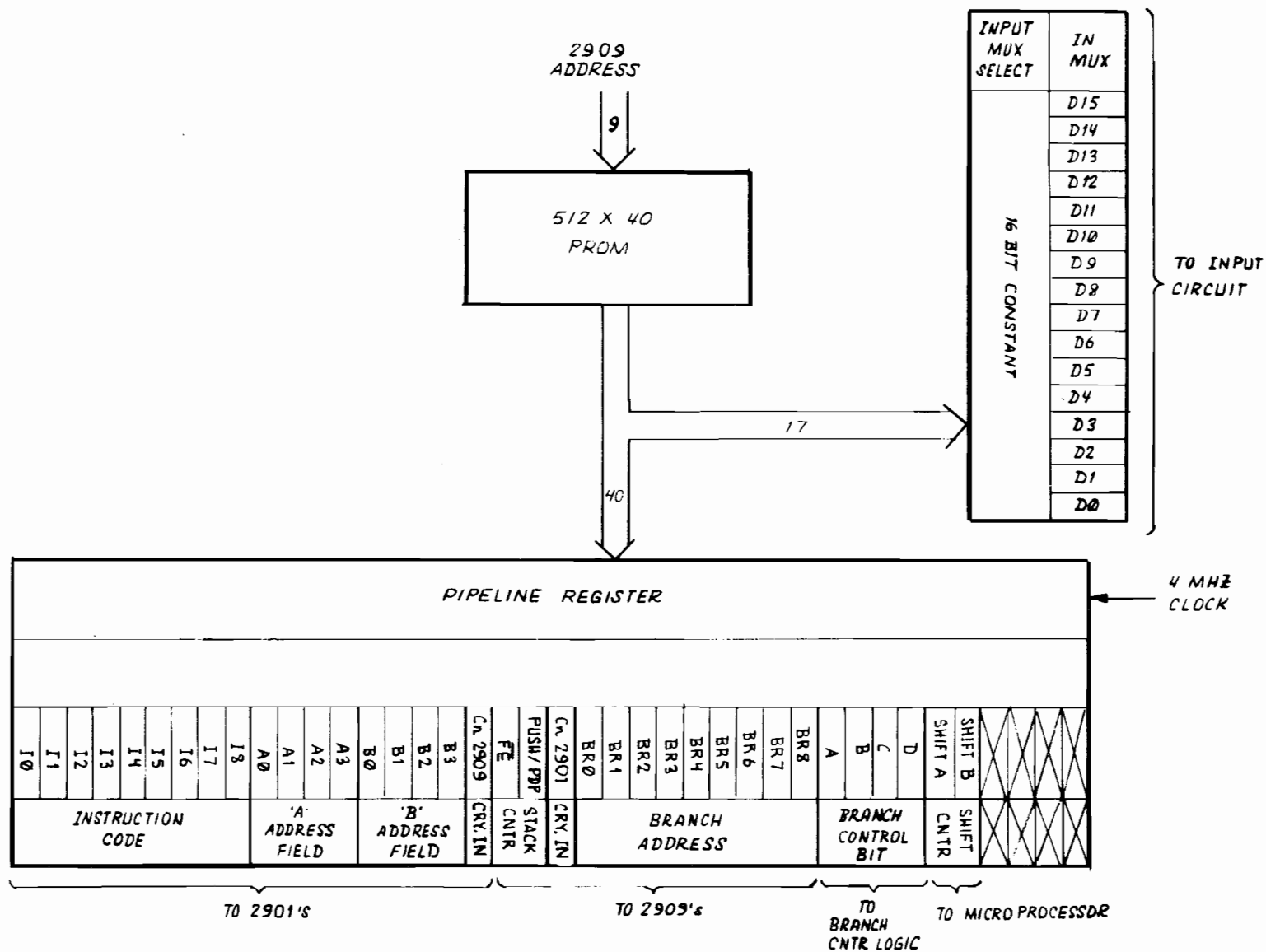


Fig. 2.4-2. Microprogram storage and pipeline register.

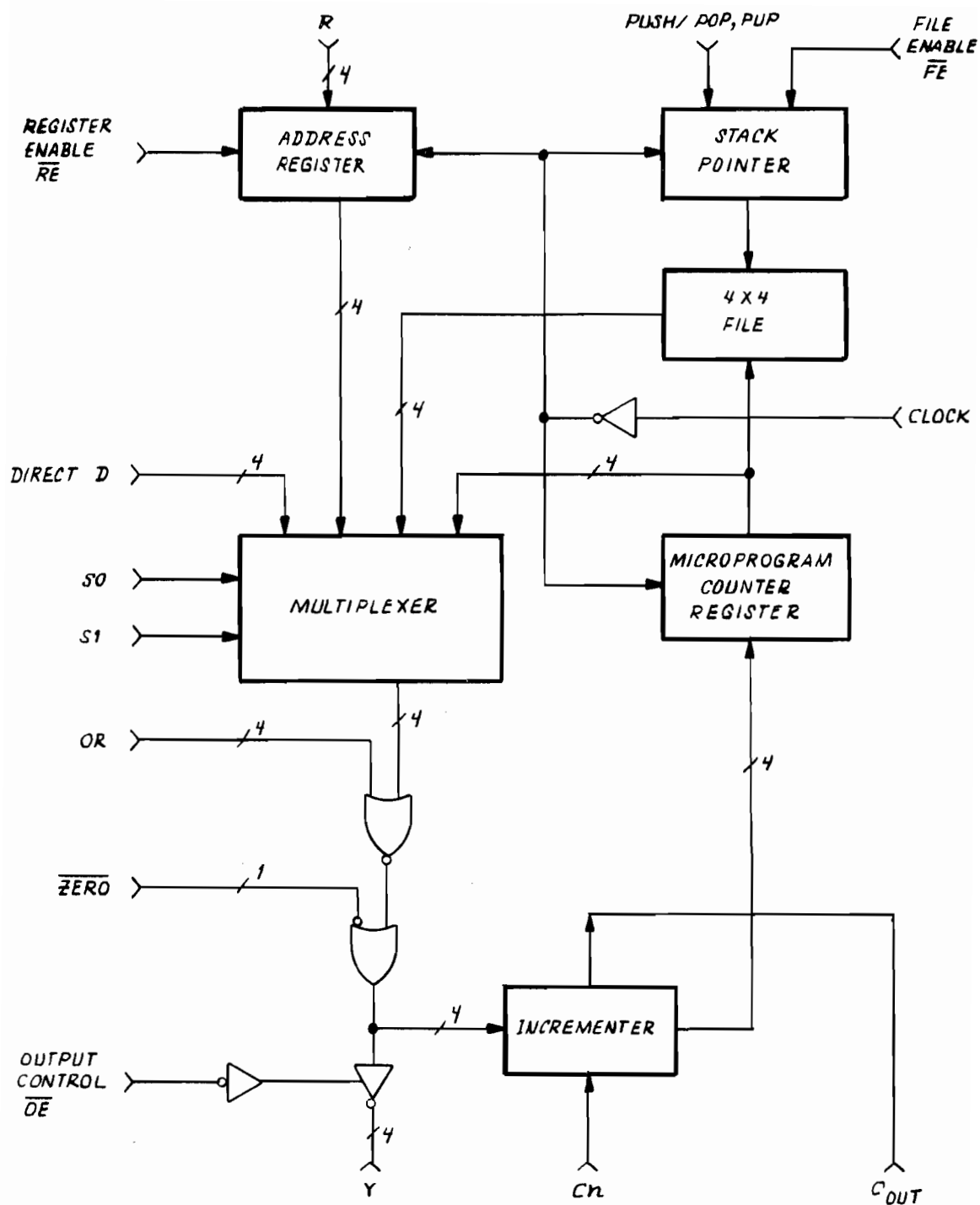


Fig. 2.4-3. Block diagram of a 2909.

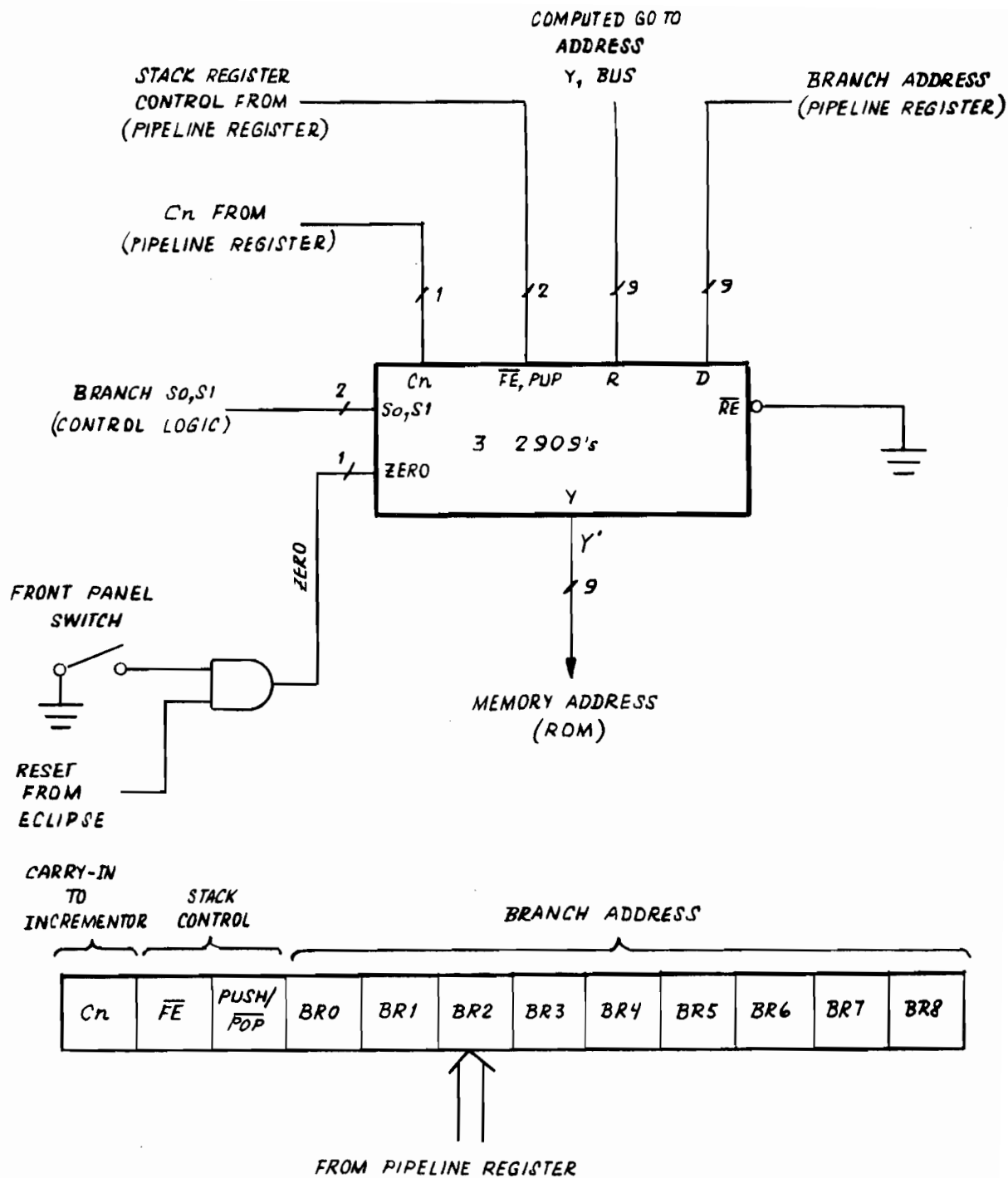


Fig. 2.4-4. Program control logic.

To provide a means of returning from a sub-routine call, a stack file and stack pointer are provided in the 2909's. The 4-word stack file allows four-deep nesting of subroutines. The stack pointer, always pointing to the last file word written, operates as an up/down counter. Its mode of operation is programmable by controlling the PUSH/POP and file enable ( $\overline{FE}$ ) input. To execute a jump-to-subroutine instruction (JSR), PUSH/POP is set to one and file enable ( $\overline{FE}$ ) is set to zero. This causes the current address in the microprogram register to be written in the file and the stack pointer to increment.

To return from a sub-routine call, both the file enable ( $\overline{FE}$ ) and the PUSH/POP control inputs are programmed low, and the stack register is selected as the source for the multiplexer. With the file enable and PUSH/POP set low, the stack pointer will decrement on the next clock cycle.

Stack operation is summarized as follows:

<u>Stack Enable (<math>\overline{FE}</math>)</u>	<u>Push/Pop (PUP)</u>	<u>Stack Register Change</u>
H	X	No change
L	H	Increment stack pointer; push current program counter (PC) to stack register
L	L	Decrement stack pointer

The 9-bit microprogram counter consists of a 9-bit incrementer followed by a 9-bit register. To execute a sequential instruction the Cn input from the program control logic is programmed high. This allows the current address plus one to be loaded into the 9-bit register on the next clock cycle. However, if Cn is programmed low, the incrementer passes the current address unmodified, and the microprogram register is loaded with the same address on the next cycle. Thus the same instruction can be executed any number of times.

To reset the microprogram counter to zero, the ZERO input of the 2909 can be grounded. This is accomplished by a front panel switch, or via a reset line (IORST) from the CPU (Eclipse).

#### 2.4.1.1.3 Branch Control Logic

The Branch Control Logic (see Fig. 2.4-5) consists of a 2:1 multiplexer, a 4-bit status register, two 8:1 conditional code multiplexers, a 16 word x 4 bit RAM, a bi-directional shift register used as a memory pointer, and a control ROM.

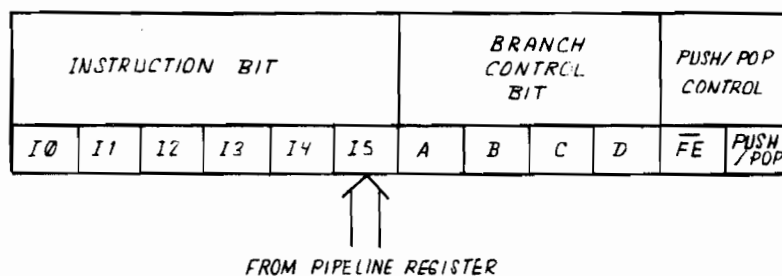
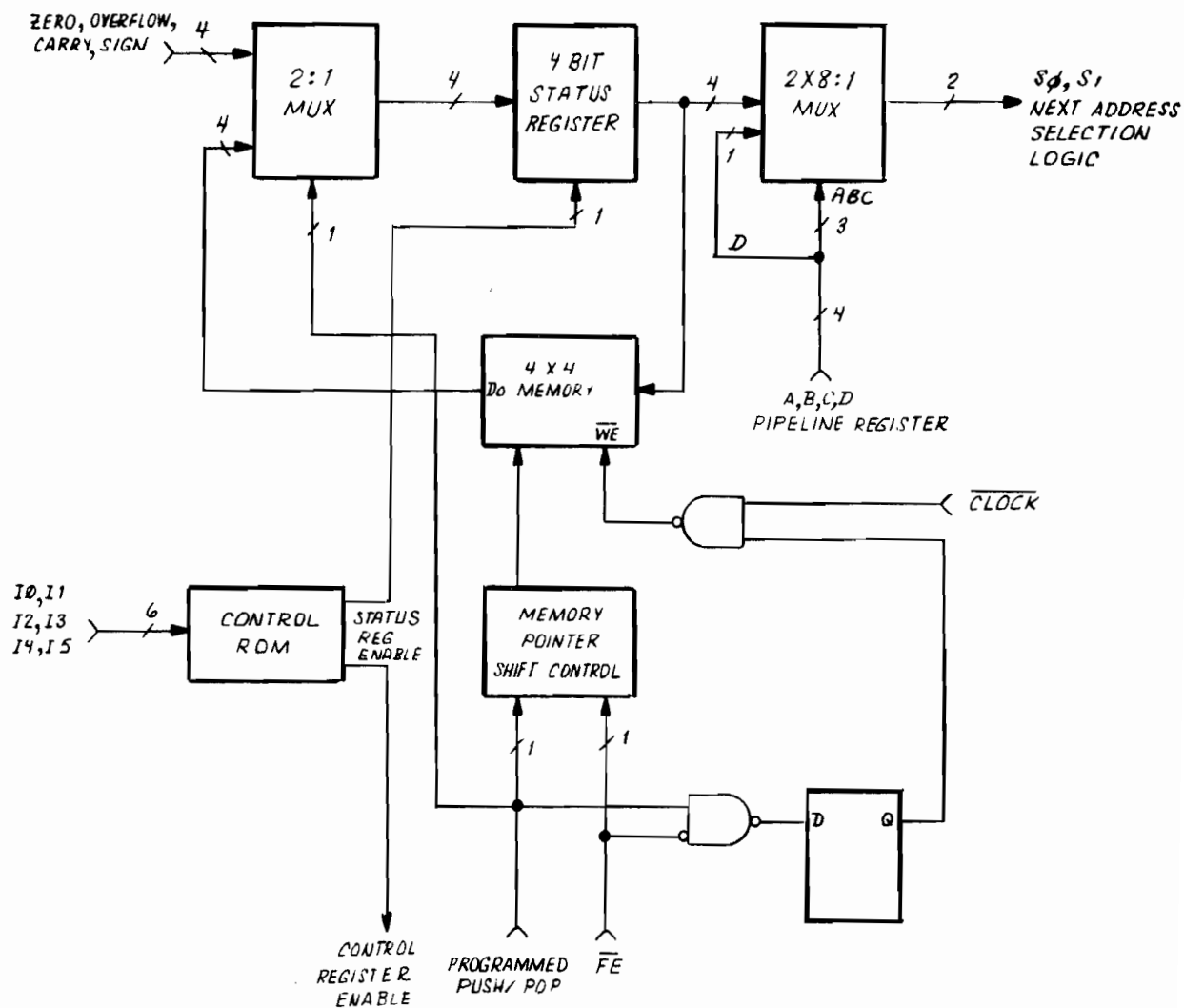


Fig. 2.4-5. Branch control logic.



The 2:1 multiplexer is normally programmed to allow the status register to receive four current status bits from the 2901's. The status register is enabled by the control ROM for those instructions selected by the six bits ( $I_0$  through  $I_5$ ) that will cause the status to change (i.e., arithmetic, logical, and shift operations). The status register is also enabled during a software POP operation, in which the 2:1 multiplexer is also selected to allow the register to restore the previous status.

The heart of the branch control logic are two 8:1 condition code multiplexers (see Fig. 2.4-6). These multiplexers receive their 3-bit address and 2 of the eight inputs from the pipeline register. The other six inputs come from the 4-bit status register which contains four status conditions. They are: 1) zero, if the result of the arithmetic, logical or shift operation is zero 2) sign, resulting from a negative (i.e., MSB = 1) arithmetic, logical or shift operation 3) carry, the result of an arithmetic, logical or shift operation in which a carry out is generated and 4) overflow, the result of an arithmetic operation which exceeds the available two's complement number range.

The value of the four branch control bits A, B, C, and D, depend on the instruction that is currently executing. If the instruction is a conditional branch, address 0 through 5 may be selected (see Table 2.4-1).

If the selected condition is true i.e., logical 1, the two branch control bits will be set. As described in the previous section,  $S_0 S_1 = (1, 1)$  will cause the 2909's to address from the D-inputs -- causing a conditional jump. However, if the condition is false,  $S_0 S_1$  will be (0, 0), which will cause the 2909's to execute the next sequential instruction. If the current instruction is not a conditional branch, it is necessary to control the branch control bit directly from the program. Address 6 and 7, together with bit D will generate all four possible combinations of  $S_0 S_1$ : (1) (0, 0) - must be programmed for all non-branch (sequential) instructions; (2) (0, 1) - programmed in the return from subroutine instruction, RTN (i.e., return address is located in the stack register; (3) (1, 0) - jump to the address located in the address register of the 2909's; (4) (1, 1) - unconditional jump instruction.

To save the status during an external hardware interrupt, or software PUSH operation, or to retrieve the status during the software POP operation, a status stack is provided. It is a 16 x 4 memory in which only 4 locations are used (addresses 1, 3, 7, 15). The address for the memory is provided by a bi-directional shift register. During a hardware interrupt or software PUSH operation, a "one" is left shifted into the shift register. On the following clock cycle, the status is written into the memory. To retrieve the status, as in the software POP operation, the 2:1 mux is selected to allow the previous status to be stored again in the status register, and a "zero" is right shifted into the shift register. After four consecutive PUSH operations the shift register will contain four "one's". Therefore, any further PUSH operations will cause the status to be written in the same memory location (1111). Simi-

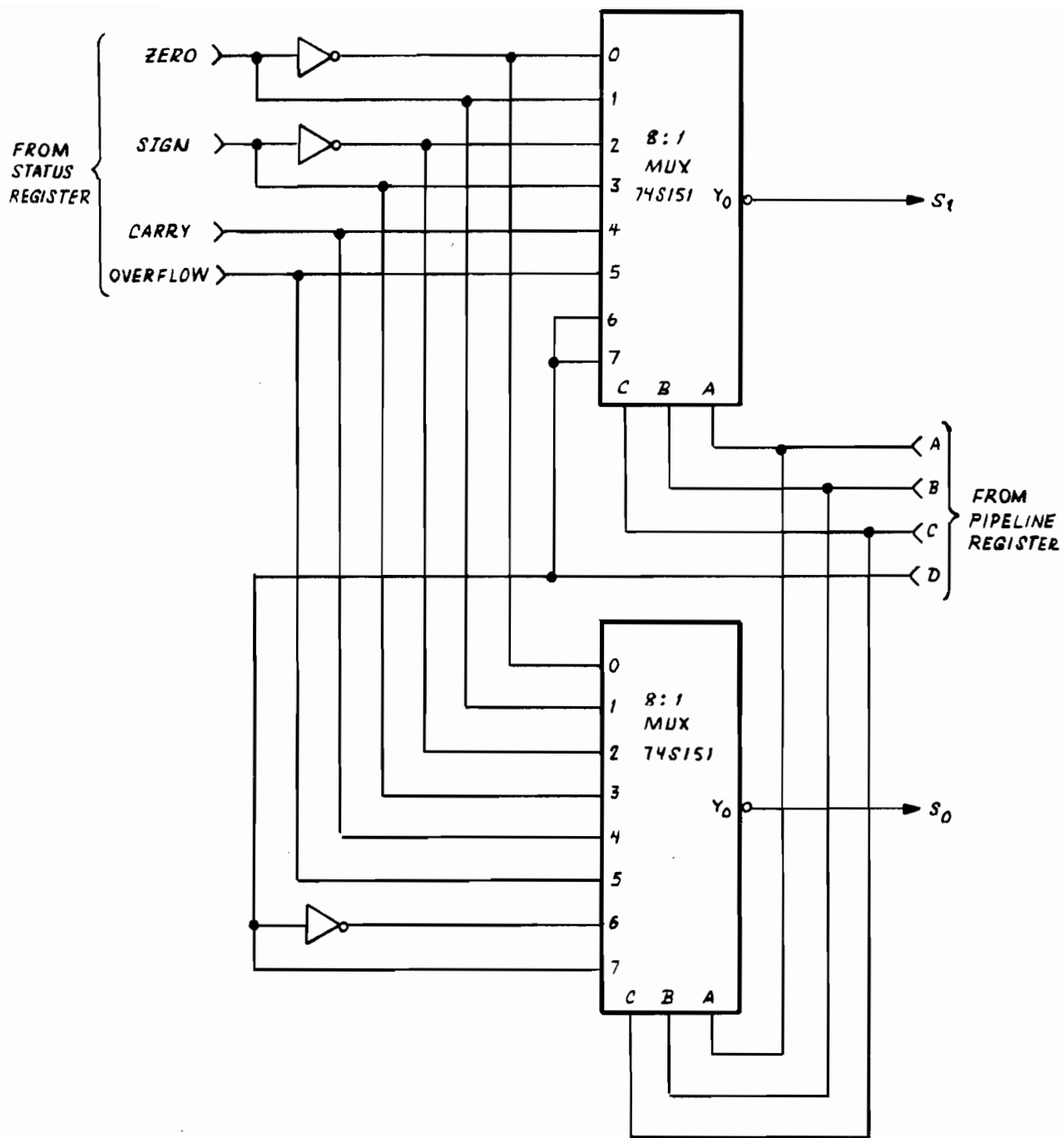


Fig. 2.4-6. Condition code multiplexer.

TABLE 2.4 -1

<u>INSTRUCTION</u>	<u>ADDRESS</u>				<u>BRANCH CONTROL BITS</u>		
	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>S<sub>0</sub></u>	<u>S<sub>1</sub></u>	
BRANCH ON NOT ZERO (BNZ)	0	0	0	X	0	0	$\overline{\text{zero}} = 0$
					1	1	$\overline{\text{zero}} = 1$
BRANCH ON ZERO (BRZ)	0	0	1	X	0	0	$\text{zero} = 0$
					1	1	$\text{zero} = 1$
BRANCH ON NOT NEGATIVE (BNN)	0	1	0	X	0	0	$\overline{\text{sign}} = 0$
					1	1	$\overline{\text{sign}} = 1$
BRANCH ON NEGATIVE (BRN)	0	1	1	X	0	0	$\text{sign} = 0$
					0	0	$\text{sign} = 0$
BRANCH ON CARRY (BRC)	1	0	0	X	0	0	$\text{carry} = 0$
					1	1	$\text{carry} = 1$
BRANCH ON OVERFLOW (BRO)	1	0	1	X	0	0	$\text{overflow} = 0$
					1	1	$\text{overflow} = 1$
RETURN THIS STACK REGISTER (RTN)	1	1	0	0	0	1	
ADDRESS REGISTER (JAR)	1	1	0	1	1	0	
UNCONDITIONAL JUMP (JMP)	1	1	1	0	1	1	
<b>SEQUENTIAL</b>	1	1	1	1	0	0	

larly, more than four consecutive POP operations will cause the status to be written into the same memory location (0000). It is the responsibility of the programmer not to allow more than four PUSH or POP operations. The operation of the shift register is summarized in Table 2.4-2. The content of the shift register is initialized to (0000) during hardware reset.

TABLE 2.4-2

<u>FE</u>	<u>PUSH/POP</u>	<u>Shift register</u>	<u>Function</u>
1	1	0 0 0 0	Do-nothing
0	1	0 0 0 1	PUSH
0	1	0 0 1 1	PUSH
0	1	0 1 1 1	PUSH
0	1	1 1 1 1	PUSH
0	0	0 1 1 1	POP
0	0	0 0 1 1	POP
0	0	0 0 0 1	POP
0	0	0 0 0 0	POP

#### 2.4.1.1.4 Microprocessor

The heart of the ARIES Target Controllers is the 2901 microprocessor. This four bit device, as shown in Fig. 2.4-7, consists of a 16 word x 4 bit two port RAM, a high speed ALU, and associated shifting decoding and multiplexing circuitry.

Four cascaded 2901's implement the 16 bit microprocessor. Other support circuitries used are the 2902 look ahead carry generator for high speed arithmetic operation, and two 25LS253 dual four-input multiplexers connected to provide four shift modes (see Fig. 2.4-8).

Two key elements in each of the 2901's are the 16 word memory and the 8-function ALU.

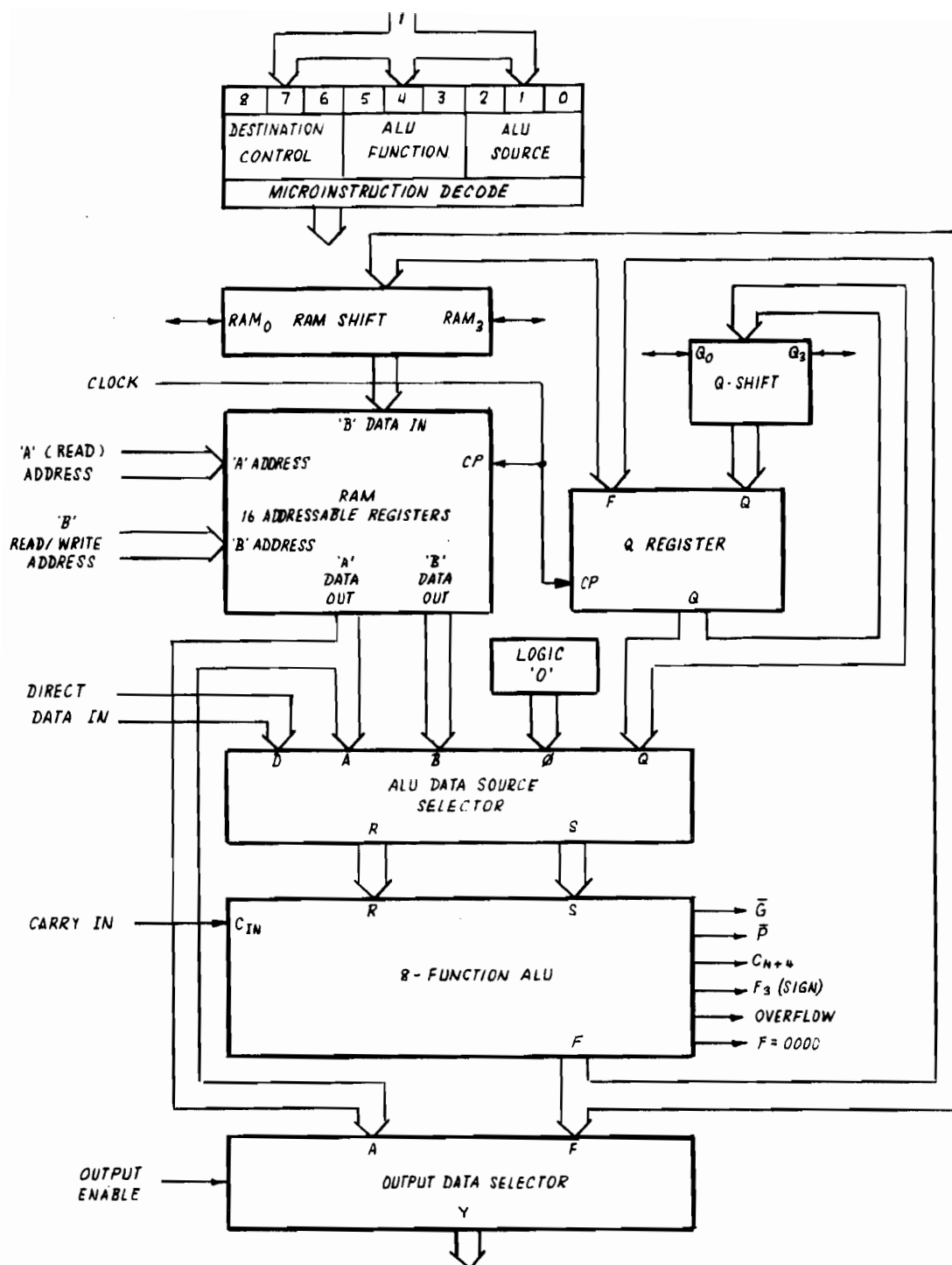
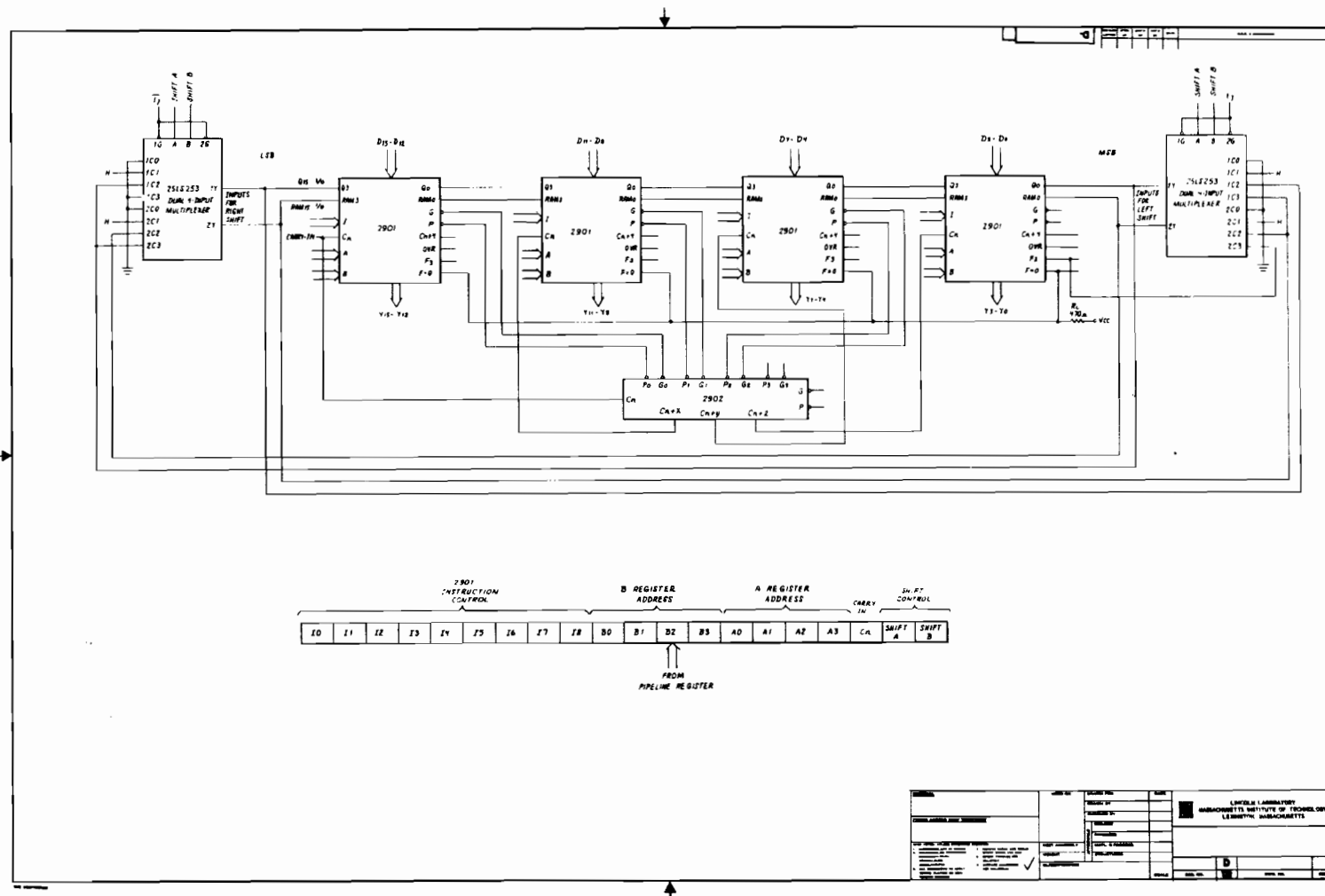


Fig. 2.4-7. Microprocessor slice, block diagram.



Data in any of the 16 words of Random Access (RAM) memory can be read from the A port of the RAM as controlled by the 4-bit A-address field input ( $A_0 - A_3$ ). Similarly, data in any of the 16 words of RAM as controlled by the 4-bit B-address field input ( $B_0 - B_3$ ) can be read simultaneously, from the B port. If the same address is applied to ( $A_0 - A_3$ ) and ( $B_0 - B_3$ ), identical data will appear at both the A and the B output ports.

New data can be written into any of the 16 word locations by specifying the B address field only. During the process of writing, the A address field will be ignored. The data input of the RAM is driven by a 3-input multiplexer, (RAM SHIFT) which allows the input data to the memory to be shifted up one-bit, shifted down one bit, or not shifted in either direction.

There is one additional register called the Q-Register. It has its own 3-input multiplexer. Its primary function is for multiplication and division, but it can be used like the A and B registers.

The ALU can perform 8 functions on its two input RAMs - five logical and three arithmetic. The control lines  $I_3$ ,  $I_4$  and  $I_5$  select the function to be performed as shown in Table 2.4-3.

TABLE 2.4-3

MICRO CODE				ALU Function	Symbol
$I_5$	$I_4$	$I_3$	Octal Code		
L	L	L	0	R Plus S	R+S
L	L	H	1	S Minus R	S-R
L	H	L	2	R Minus S	R-S
L	H	H	3	R or S	RvS
H	L	L	4	R And S	RAS
H	L	H	5	$\overline{R}$ And S	$\overline{R}AS$
H	H	L	6	R Ex-or S	R $\nabla$ S
H	H	H	7	R Ex-Nor S	R $\nabla$ S

ALU Function Control

The R and S inputs of the ALU are driven by 2 and 3 input multiplexers respectively. This allows 8 functions to be performed on any combination of the two inputs: A port, B port, external data D, Q registers and zero. Selection of the ALU source operand is controlled by lines  $I_0$ ,  $I_1$  and  $I_2$  as shown in Table 2.4-4.

TABLE 2.4-4

MICRO CODE				ALU SOURCE OPERANDS	
$I_2$	$I_1$	$I_0$	Octal Code	R	S
L	L	L	0	A	0
L	L	H	1	A	B
L	H	L	2	0	0
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	0
H	H	H	7	D	0

D inputs allow external data to be entered into the microprocessor. Such data could be a load immediate constant from the ROM to any of the 17 registers, or external data from the D-Bus. (See Section 2.4.1.1.5 Input Circuits, for more detail).

One additional input to the ALU is the carry-in bit,  $C_n$ . It is used in conjunction with the carry generate and carry propagate outputs of the ALU to run the 2902 look ahead carry generator. This arrangement allows the anticipation of carries across the 2901's, thereby eliminating the lengthy carry propagation time from LSB to MSB. The  $C_n$  of the least significant 2901 comes from the pipeline register. It is normally programmed low, and is programmed high only for the incrementing instruction.



The ALU has four status oriented outputs. These are sign (F3), overflow (OVR), carry out (Cn +4), and Zero (F=1).

The sign bit is the most significant bit of the ALU. It is used in two's complement arithmetic to indicate sign (i.e., positive or negative). The overflow bit is used to flag arithmetic operations that exceed the available two's complement range. A carry out bit is set whenever an operation, either logical or arithmetic, exceeds the 16-bit range, and zero output is used for zero detect.

The output of the ALU is routed to several destinations. It can be the output of the device, or it can be input to the RAM and Q registers. Eight possible combinations, as controlled by the lines I<sub>6</sub>, I<sub>7</sub>, and I<sub>8</sub> are illustrated in Table 2.4-5.

TABLE 2.4-5  
ALU DESTINATION CONTROL

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I <sub>8</sub>	I <sub>7</sub>	I <sub>6</sub>	Octal Code	Shift	Load	Shift	Load		RAM <sub>3</sub>	RAM <sub>0</sub>	Q <sub>3</sub>	Q <sub>0</sub>
L	L	L	0	X	NONE	NONE	F→Q	F	X	X	X	X
L	L	H	1	X	NONE	X	NONE	F	X	X	X	X
L	H	L	2	NONE	F→B	X	NONE	A	X	X	X	X
L	H	H	3	NONE	F→B	X	NONE	F	X	X	X	X
H	L	L	4	DOWN	F/2→B	DOWN	Q/2→Q	F	F <sub>3</sub>	IN <sub>0</sub>	Q <sub>3</sub>	IN <sub>0</sub>
H	L	H	5	DOWN	F/2→B	X	NONE	F	F <sub>3</sub>	IN <sub>0</sub>	Q <sub>3</sub>	X
H	H	L	6	UP	2F→B	UP	2Q→Q	F	IN <sub>3</sub>	F <sub>3</sub>	IN <sub>3</sub>	Q <sub>0</sub>
H	H	H	7	UP	2F→B	X	NONE	F	IN <sub>3</sub>	F <sub>0</sub>	X	Q <sub>0</sub>

X = Don't care.

B = Register Addressed by B inputs.

UP is toward MSB, Down is toward LSB.

The control lines  $I_6$ ,  $I_7$ , and  $I_8$  also determine whether the data bits from the ALU will be shifted up one or shifted down one before they are loaded into the RAM or Q register.

The Q register and RAM left/right shift-data transfers occur between devices over bi-directional tri-state lines ( $Q_n$ ,  $RAM_n$ ). A dual tri-state, four-input, multiplexer (25LS253) is connected at the most significant and least significant device to select what new input should be presented to the register during shifting (see Fig. 2.4-8). Control line  $I_7$ , used for controlling up-shift or down-shift, is also used to enable either the up-shift multiplexer, or down-shift multiplexer. Two shift control bits, A and B, are used to select the shift modes (zero, one, rotate, or arithmetic) as shown in the following Table 2.4-6.

TABLE 2.4-6

	Shift Control		Source of New Data					
	B	A	Q	Q	RAM	RAM		
I <sub>7</sub>			LSB	MSB	LSB	MSB	Shift	Type
1	0	0	0	Q <sub>1</sub>	0	F <sub>1</sub>	} UP	ZERO
1	0	1	1	Q <sub>1</sub>	1	F <sub>1</sub>		ONE
1	1	0	Q <sub>0</sub>	Q <sub>1</sub>	F <sub>0</sub>	F <sub>1</sub>		ROTATE
1	1	1	0	Q	Q <sub>0</sub>	F <sub>1</sub>		ARITHMETIC
0	0	0	Q <sub>14</sub>	0	F <sub>14</sub>	0	} DOWN	ZERO
0	0	1	Q <sub>14</sub>	1	F <sub>14</sub>	1		ONE
0	1	0	Q <sub>14</sub>	Q <sub>15</sub>	F <sub>14</sub>	F <sub>15</sub>		ROTATE
0	1	1	Q <sub>14</sub>	F <sub>15</sub>	F <sub>14</sub>	RAM <sub>0</sub> = F <sub>0</sub> RAM <sub>1</sub> = F <sub>1</sub>		ARITHMETIC

#### 2.4.1.1.5 Input Circuits

The input circuits (see Fig. 2.4-9) provide data to the D-inputs of the microprocessor 2901's from a 16-bit input register and a two-input multiplexer. The D-Bus input is normally selected. This allows data from the external de-

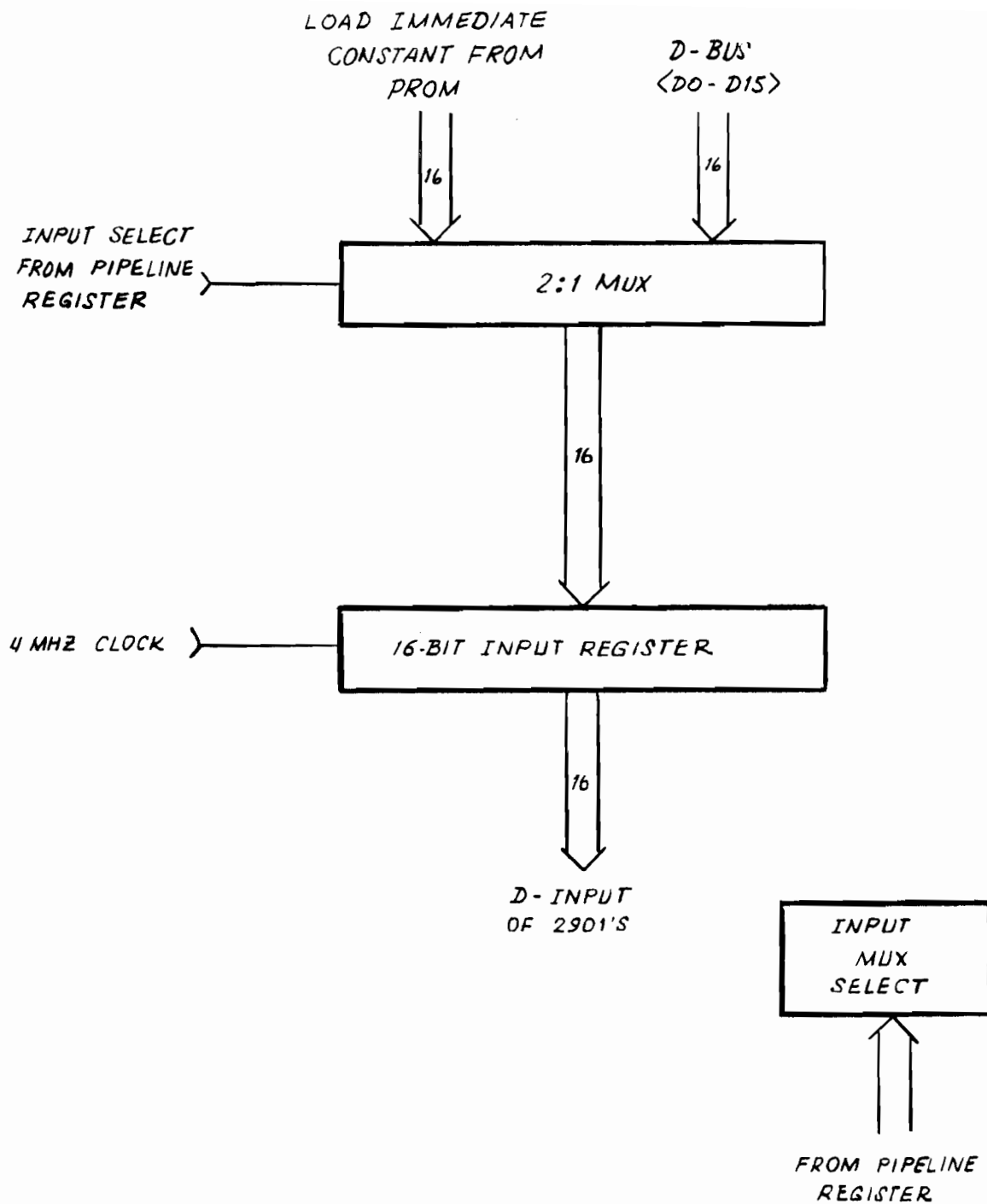


Fig. 2.4-9. Input circuits.

vices to enter the controller. Since the D-Bus is an open collector bus tied to many devices, only one device, addressed by Controller via the C-Bus, is allowed to output data (see Output Circuits). During execution of a load immediate instruction, the multiplexer is selected so that a known constant can be input from the PROM.

#### 2.4.1.1.6 Output Circuits

The outputs of the 2901's Y Bus drive two sets of registers located in the output circuits (see Fig. 2.4-10). Note that one is labeled as the output register, and the other is labeled as the control register. The 16-bit output register, which is always enabled, serves as a data link between the target controller and the rest of the system. The Y-Bus is connected to many devices. Therefore, in order to address each device properly, another register is needed. This is the purpose of the 12 bit control register. When enabled (by the control ROM in "branch control logic",) the control register will be loaded with a device code and an OP-code, so that only one device will respond to the OP-code and the data on the Y-Bus. This same device code and OP-code will remain in the control register until the data transfer is completed, or a particular operation is done (i.e., reset, read, etc.). The operations and their codes are shown in Table 2.4-7.

#### 2.4.1.2 Controlled ARIES Targets (CAT's)

As noted previously, ARIES generates two types of replies: controlled replies generated by the three CAT's as commanded by the ARIES computer via the CRC just described, and fruit replies generated by the three FAT's using control inputs originating in the Random Process Generator module as controlled by the Fruit Controller. The type of reply is determined by the subsystem that drives these identical CAT and FAT modules. These modules are referred to as "reply generators" in the following sections.

The reply generators accept four or ten word message blocks (four words for an ATCRBS reply or ten words for a DABS reply) and each reply generator module has buffer space for 5 complete ATCRBS or 2 DABS replies so that a consecutive string of replies can follow one another as rapidly as possible without controller intervention. Each message block contains the following data fields:

- a. trigger time (range)
- b. reply code for ATCRBS; message block for DABS
- c. power level

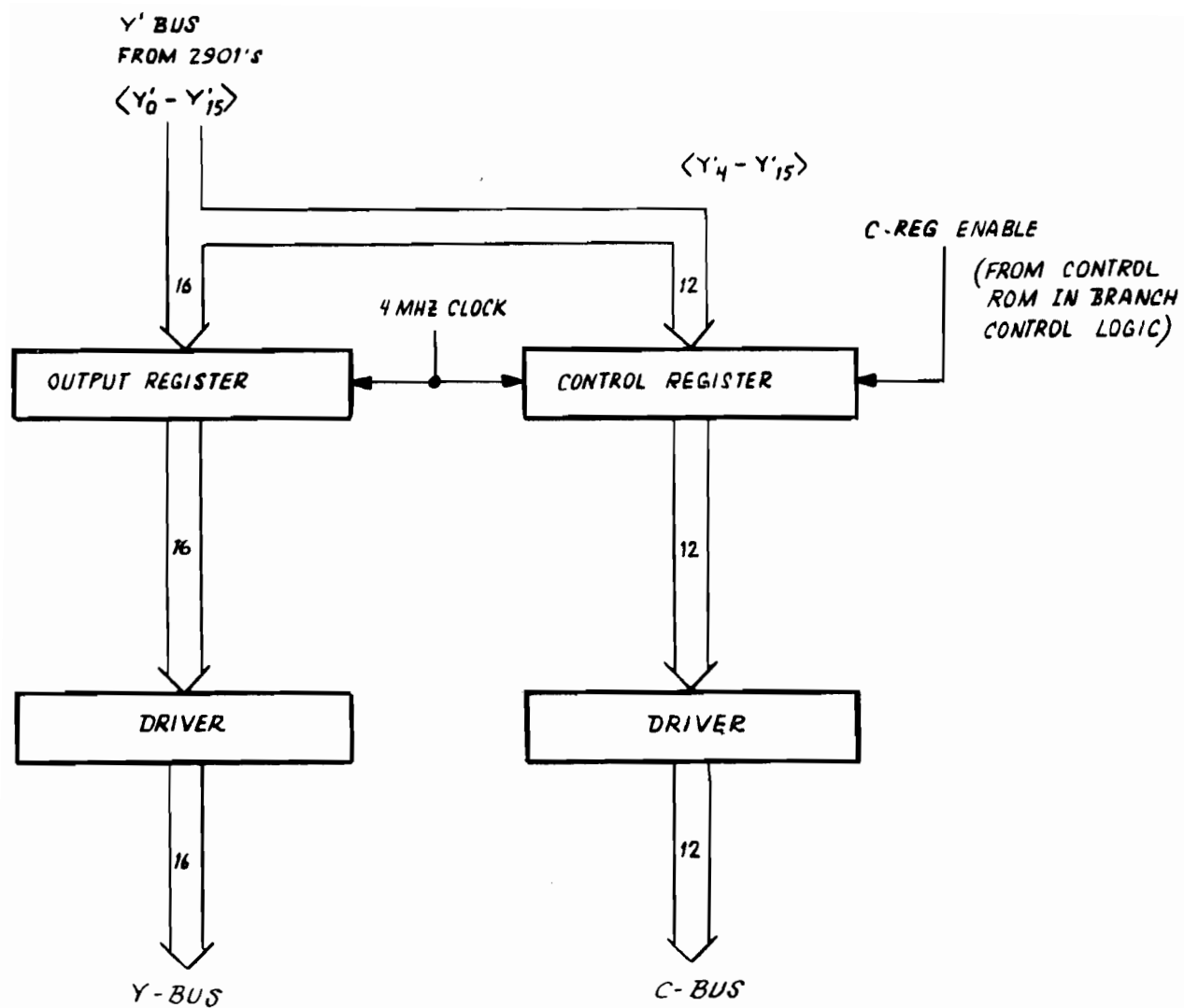


Fig. 2.4-10. Output circuits.

TABLE 2.4-7

OPERATION		DEVICE CODE AND OP-CODE (OCTAL)
ARE YOU BUSY CAT (FAT)-1		0201
ARE YOU BUSY CAT (FAT)-2		0401
ARE YOU BUSY CAT (FAT)-3		1001
RESET CAT (FAT)-1		0202
RESET CAT (FAT)-2		0402
RESET CAT (FAT)-3		1002
LOAD 1-4 CAT (FAT)-1		0210
LOAD 1-4 CAT (FAT)-2		0410
LOAD 1-4 CAT (FAT)-3		1010
LOAD 5-10 CAT (FAT)-1		0220
LOAD 5-10 CAT (FAT)-2		0420
LOAD 5-10 CAT (FAT)-3		1020
RESET RPG		0102
READ RPG		0104
LOAD RPG		0110
SEND INTERRUPT TO FAT INTERFACE		4001
READ STATUS IN FAT INTERFACE		6000
SEND DATA TO FAT INTERFACE		5000
RECEIVE DATA FROM FAT INTERFACE		4400
CLEAR RESET FLIP-FLOP CAT INTERFACE		4100
READ 1-4 FROM CAT INTERFACE		4200
READ 5-10 FROM CAT INTERFACE		4400
READ STATUS FROM CAT INTERFACE		5000
DECREMENT REPLY COUNTER IN CAT INTERFACE		6000

- d. monopulse off-boresight angle
- e. mainbeam or sidelobe reply (CAT replies are always mainbeam; FAT replies occur randomly as mainbeam or sidelobe replies at a computer controlled average ratio).

Reply generator action is initiated when the 16-bit trigger-time word is coincident with a 16 MHz reply trigger-time counter. The target generator modulator then produces a pulse amplitude modulation (PAM) data stream for ATCRBS replies and a pulse position modulation (PPM) data stream for DABS replies. The modulating waveform (either PAM or PPM) plus the power level, monopulse offboresight angle, and mainbeam or sidelobe control signals are sent to the reply generator IF unit which produces the actual IF reply at 60 MHz.

#### 2.4.1.2.1 Reply Generator Data Bus Structure

Each reply generator receives its data via the Y-Bus and its control commands from the controller via the C-bus. The generator can report its status (i.e., buffer full condition) to the controller using the D-Bus. As shown in Fig. 2.4-11, three reply generators share the same buses and each generator can receive (send) data from (to) the bus only if the control commands in the C-Bus are addressed to it. The device addresses and op codes used with the four control commands are shown in Table 2.4-8; definitions of these commands are as follows:

ARE YOU BUSY?	Can the buffer in the reply generator accommodate 10 words?
RESET:	Clear buffer, and initialize the reply generator.
LOAD 1-4:	Load four words of an ATCRBS reply, or the first four words of a DABS reply, into the buffer.
LOAD 5-10:	Load the remaining 6 words of a DABS reply.

The normal sequence of reply loading is as follows: the controller first sends out an ARE YOU BUSY? command to a reply generator. Depending on the status of the buffer, the reply generator will respond with a "1" (not busy) or "0" (busy) on the D-bus. When a "1" is received by the controller, it will output a LOAD 1-4 command and four reply words. If the reply is a DABS type, a LOAD 5-10 command and the remaining six words will also be sent. The controller then polls the next reply generator likewise. The controller will continue this round-robin process as long as reply data are available from the CPU.

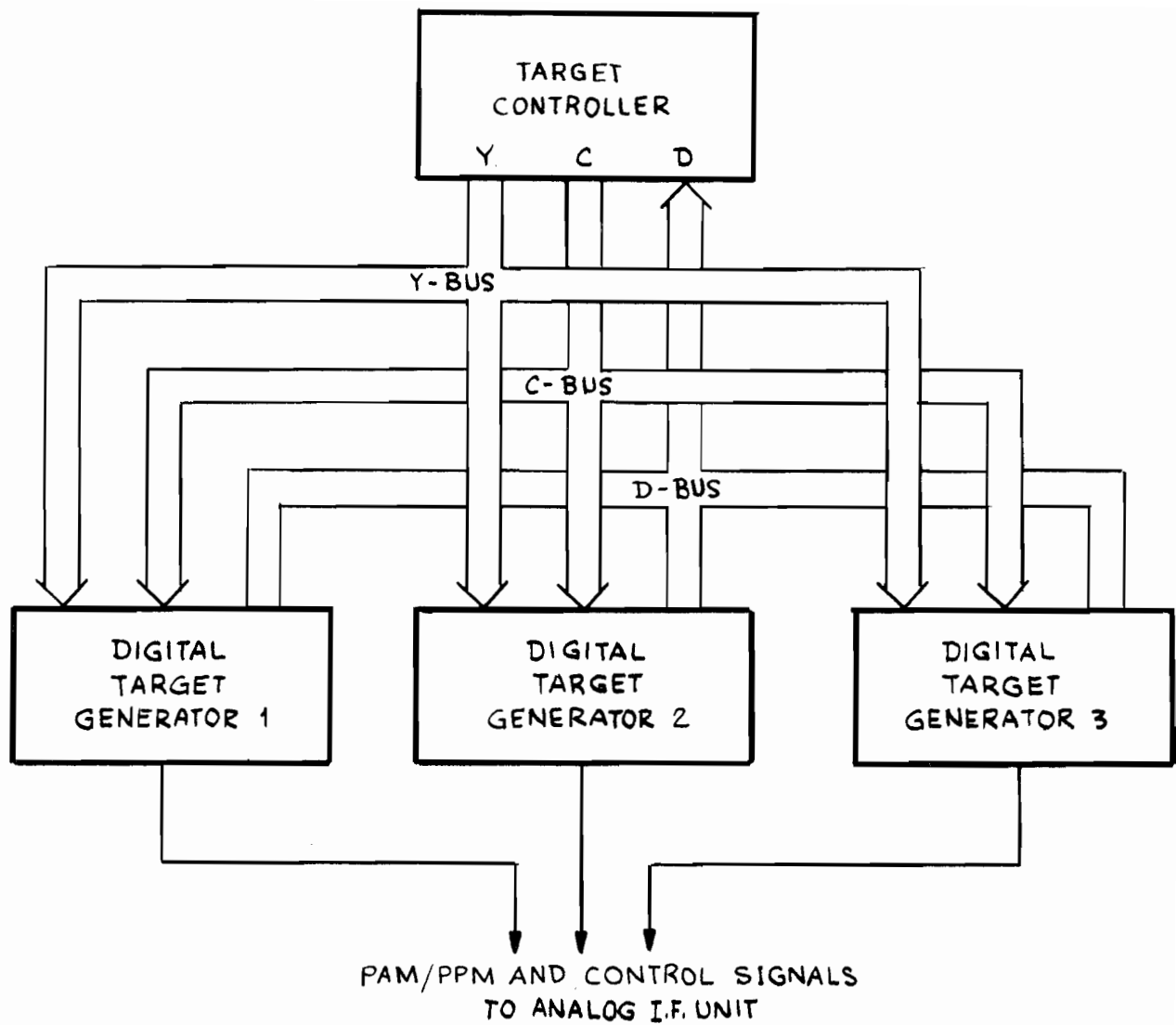


Fig.2.4-11. Data bus interconnecting the controller and its target generators.



TABLE 2.4-8  
DEVICE ADDRESS AND OP CODE FOR TARGET GENERATOR

	Device Address						OP Code							
ARE YOU BUSY?	1	1	1	1	0	1	1	1	1	1	1	0		
RESET	1	1	1	1	0	1	1	1	1	1	0	1		
SPARE	1	1	1	1	0	1	1	1	1	0	1	1		
LOAD 1-4	1	1	1	1	0	1	1	1	0	1	1	1		
LOAD 5-10	1	1	1	1	0	1	1	0	1	1	1	1		
	MSB	0	1	2	3	4	5	6	7	8	9	10	11	LSB

Target #1

	Device Address						OP Code							
ARE YOU BUSY?	1	1	1	0	1	1	1	1	1	1	1	0		
RESET	1	1	1	0	1	1	1	1	1	1	0	1		
SPARE	1	1	1	0	1	1	1	1	1	0	1	1		
LOAD 1-4	1	1	1	0	1	1	1	1	0	1	1	1		
LOAD 5-10	1	1	1	0	1	1	1	0	1	1	1	1		
	MSB	0	1	2	3	4	5	6	7	8	9	10	11	LSB

Target #2

	Device Address						OP Code							
ARE YOU BUSY?	1	1	0	1	1	1	1	1	1	1	0			
RESET	1	1	0	1	1	1	1	1	1	0	1			
SPARE	1	1	0	1	1	1	1	1	0	1	1			
LOAD 1-4	1	1	0	1	1	1	1	1	0	1	1			
LOAD 5-10	1	1	0	1	1	1	1	0	1	1	1			
	MSB	0	1	2	3	4	5	6	7	8	9	10	11	LSB

Target #3

The reply generators are reset by the RESET command during power up or when an IORST\* is executed from the CPU.

#### 2.4.1.2.2 Major Elements of the Reply Generator

Figure 2.4-12 illustrates the major elements of the reply generator. It consists of: (1) Input Buffer and Registers, (2) Delay-to-Trigger Timing Circuit, and (3) Reply Assembler.

The reply generator receives two types of data from the reply controller; the reply data to be included in the ATCRBS and DABS reply, and the reply control data: reply time; power level of the reply; offboresight angle of the reply; and the mainbeam to sidelobe (M/S) control bit.

The reply words are stored in the Input Buffer and Registers. They are then routed so that the reply data\*\* are forwarded to the Reply Assembler, the reply time to the Delay-to-Trigger Timing Circuit, and power level, offboresight angle, and M/S bit to the IF unit.

The reply data are transferred to the reply assembler 16 bits at a time. The assembler is then pulsed serially to control the PPM/PAM modulation, thus simulating the actual DABS/ATCRBS reply pulse stream.

To simulate range, the reply time is compared with 16-bit range clock in the Delay-to-Trigger Circuit.

When the comparator indicates that coincidence has occurred a reply trigger is generated causing the reply assembler to transmit the modulated reply. The other three control words of the reply are sent to the IF unit. These remain effective during the entire reply transmission period so that the required characteristics of the reply are simulated.

#### 2.4.1.2.3 Operation of the Reply Generator Elements

Refer to Fig. 2.4-13 for a Detailed Block Diagram of the Reply Generator.

##### 2.4.1.2.3.1 Input Buffer and Register

Note that the Input Buffer and Register consists of:

---

\* IORST is a reset instruction of the Eclipse CPU. It can be executed either from the software or by means of a console switch.

\*\* ATCRBS reply data contains only one 16-bit word.

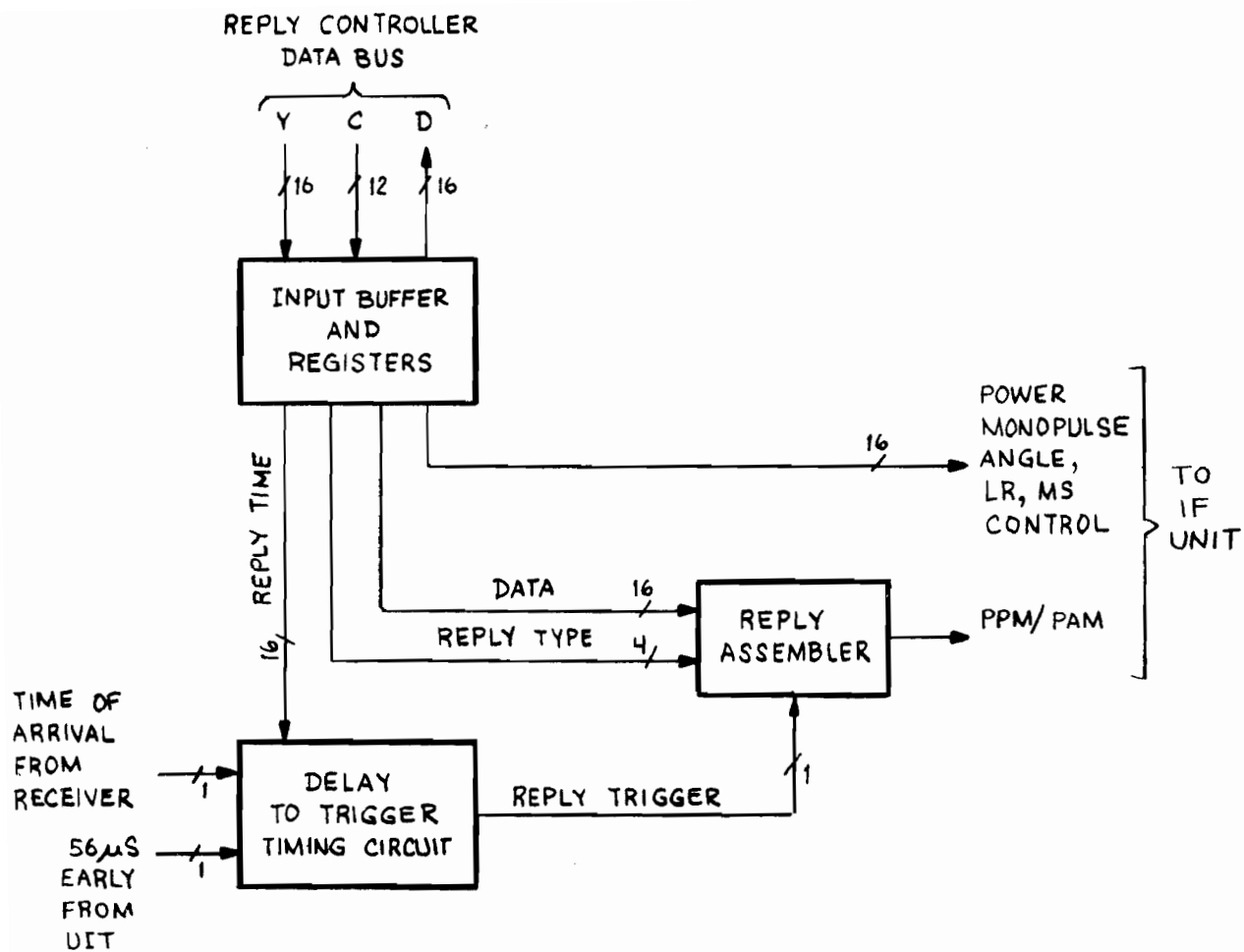


Fig. 2.4-12. Reply generator, block diagram.

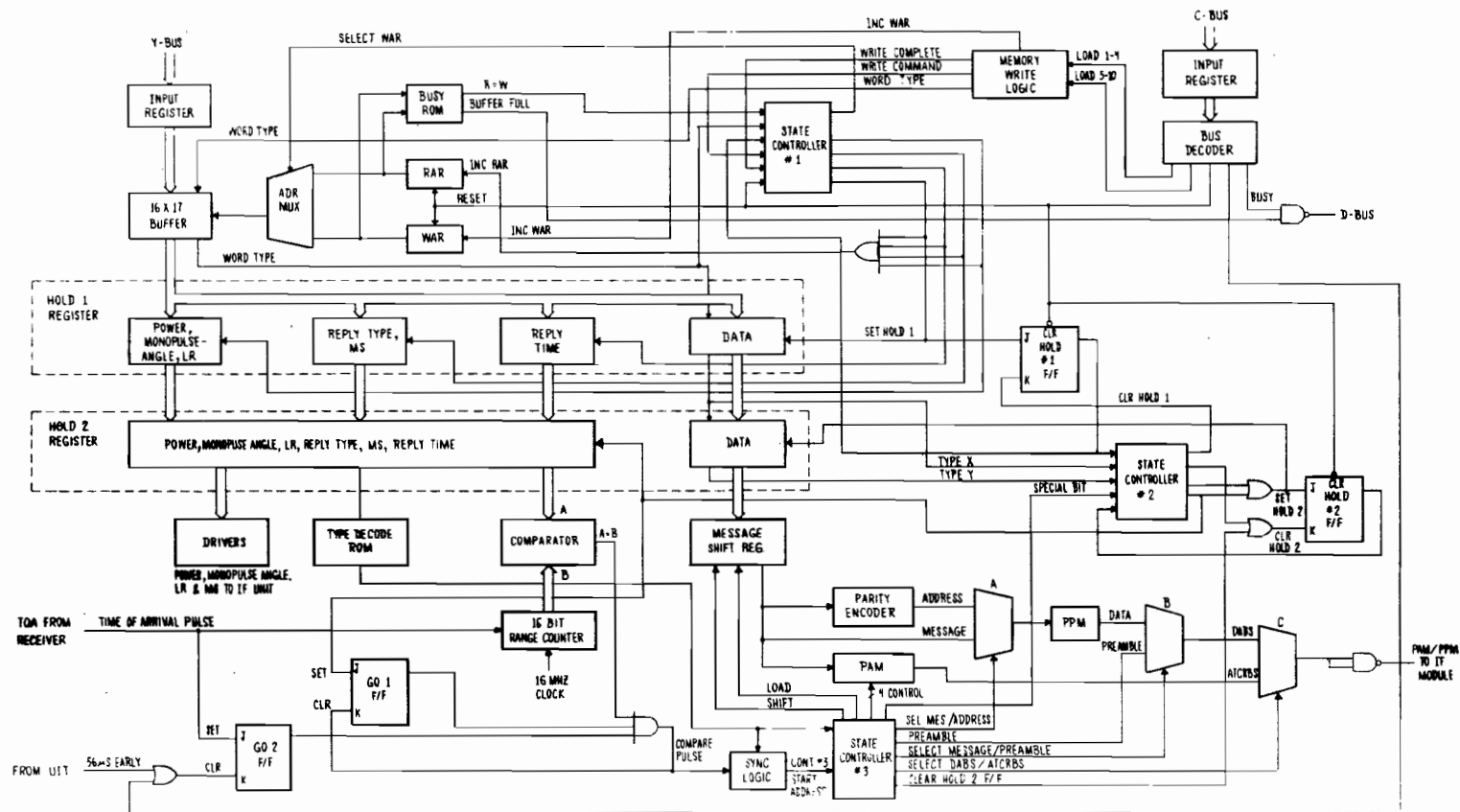


Fig. 2.4-13. Reply generator, detailed block diagram.

- (a) 17-bit x 16 word buffer and its read/write address registers.
- (b) Hold-1 register and Hold-1 flip-flop.
- (c) Hold-2 register and Hold-2 flip-flop.
- (d) C-Bus decoder.
- (e) State controllers (No. 1 and No. 2).

The 17-bit x 16 word buffer is for the purpose of stacking replies so that they can follow one another without controller intervention. Before the reply data are sent to the reply assembler and control data to the IF unit, output from the buffer is further buffered by the Hold-1 and Hold-2 registers. (The purpose of these two registers will become apparent later in the discussion). To indicate the status of the Hold-1 register, Hold-1 flip-flop is set (or reset) each time a new set of data words are transferred to (from) the register. Hold-2 flip-flop performs similarly for the Hold-2 register. The writing of new data into the buffer and the transferring of data from the buffer to the Hold-1 register are controlled by State Controller No. 1. Transferral of data from the Hold-1 register to the Hold-2 register is controlled by State Controller No. 2. State diagrams for the two state controllers are shown in Fig. 2.4-14 and Fig. 2.4-15.

Consider the following Input Buffer and Register action sequence occurring during transfer of data from the controller (power up condition is assumed to have just occurred):

- (1) The reply generator (CAT) is reset (by the reply controller via the C-Bus, the RESET command being decoded by the Bus Decoder - see Fig. 2.4-13), causing the read address register (RAR) and write address register (WAR) of the buffer to be zeroed, Hold-1 and Hold-2 flip-flops to be cleared and State Controller No. 1 and No. 2 to be reset to their idle states. The reply generator will remain in the idle state as long as no reply words are received.
- (2) When the controller is ready to send the reply words, it will first check the status of the buffer by issuing an ARE YOU BUSY? command. This command is decoded by the Bus Decoder, enabling the status (a bit from the Busy ROM) to be output to the D-Bus.
- (3) The Busy ROM determines the status of the buffer by comparing the 4-bit outputs from the read address and the write address registers. If there are 9 or fewer locations left for writing,

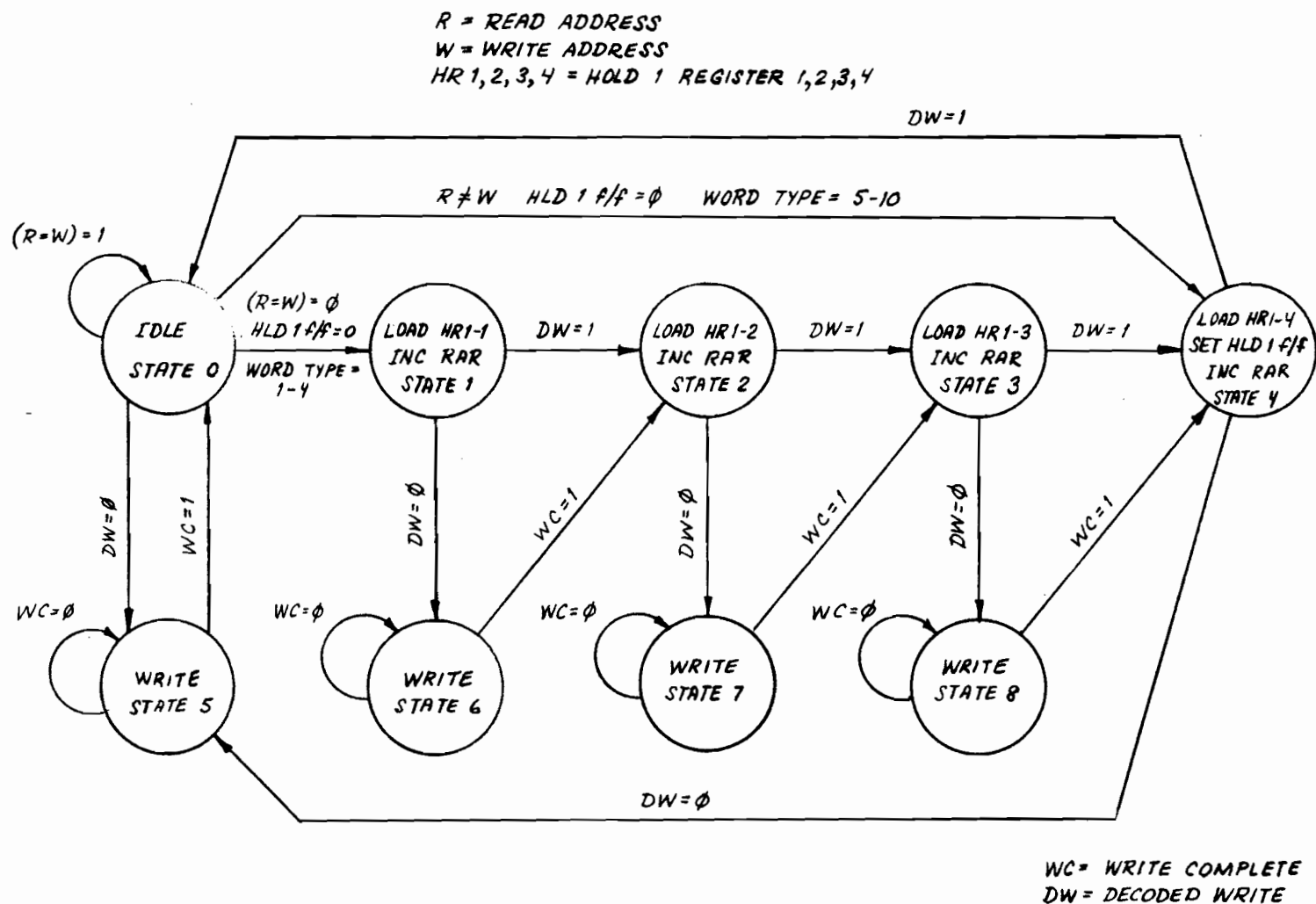


Fig. 2.4-14. State diagram for controller No.1.

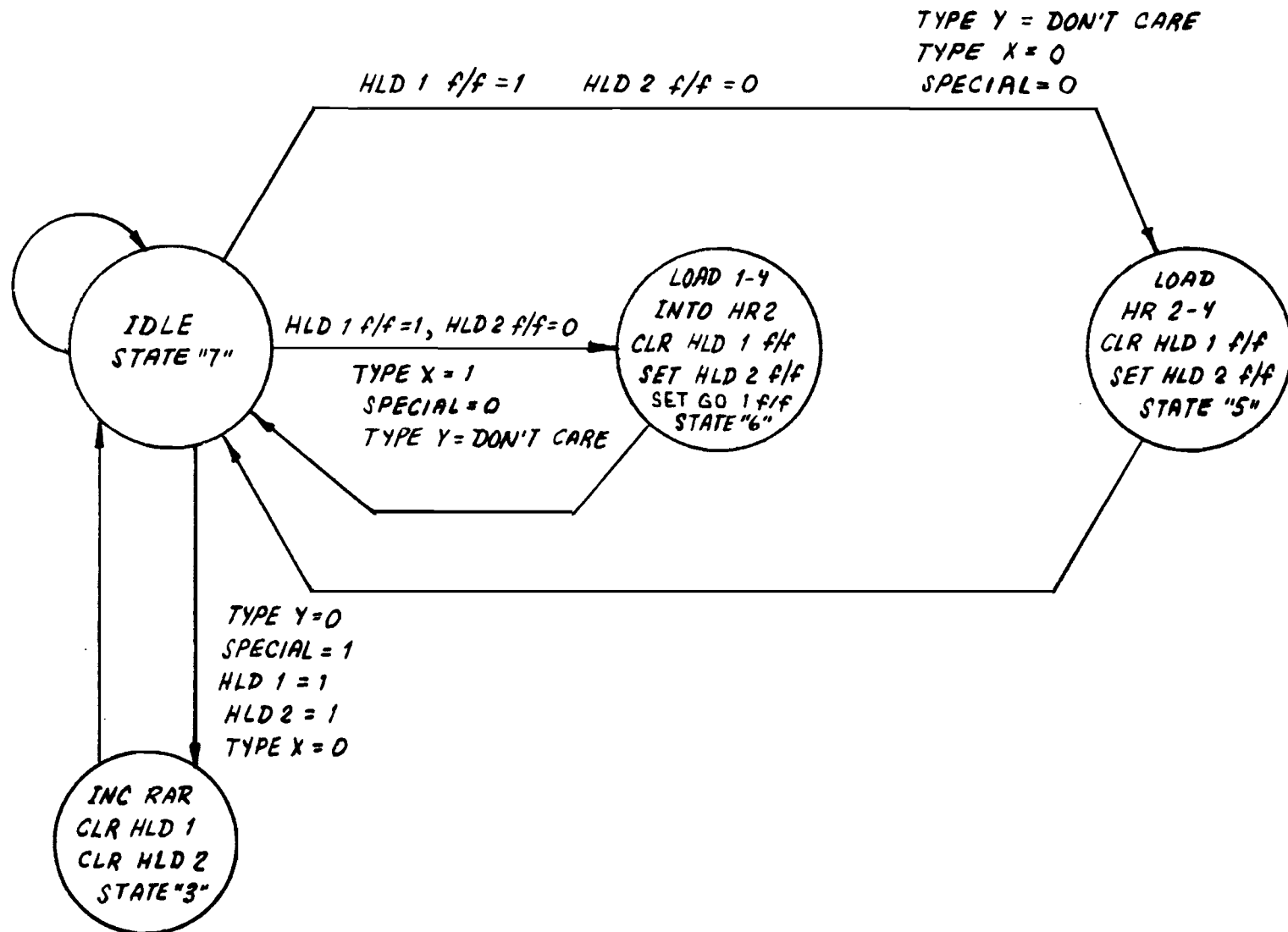


Fig. 2.4-15. State diagram for controller No.2.

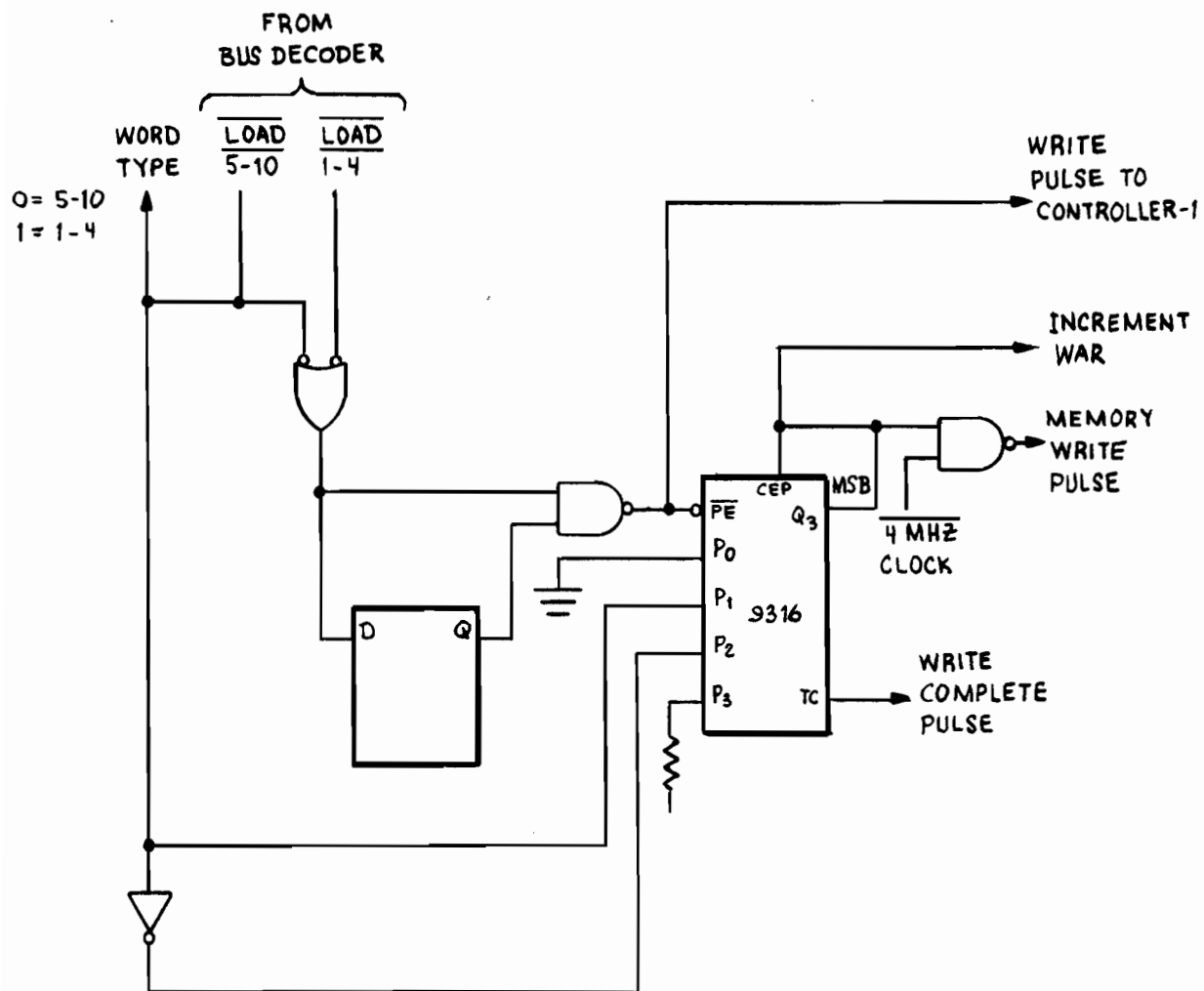
a busy bit from the ROM will be set indicating "buffer full" status. The buffer can thus accept at least a full 10 word DABS reply when it is not full.

- (4) Another bit from the Busy ROM signifies the "buffer empty" condition. It is set when the read address is equal to the write address. This "R=W" bit is used by State Controller No. 1 for reasons to be explained below.
- (5) Since the reply Generator has just been powered up, the "R=W" bit is set. This condition causes State Controller No. 1 to loop around the Idle State (State 0) until a write command from the controller is received. (See Fig. 2.4-14).
- (6) It is important at this point to notice how the ATCRBS and DABS replies are transferred. An ATCRBS reply is transferred by a LOAD 1-4 command accompanied by 4 data words.

The first 4 words of a DABS reply are transferred in a similar way using a LOAD 1-4 command, however, a LOAD 5-10 command is used for the remaining six. The LOAD 1-4 or LOAD 5-10 command remains on the C-Bus during the period of transferral so that each word can be identified as a 1-4 or 5-10 type by the reply generator.

- (7) When a LOAD command is decoded by the Bus Decoder, a "write" pulse will be generated by the Memory Write Logic to Controller No. 1, causing four words to be written into the buffer if it is a LOAD 1-4 command and six words, if it is a LOAD 5-10 command. Fig. 2.4-16 and Fig. 2.4-17 illustrate the Memory Write Logic, and a waveform for the LOAD 1-4 sequence.
- (8) At the completion of writing, a "write complete" pulse from the Memory Write Logic will cause Controller No. 1 to return to State 0 from State 5. Refer to the state diagram for Controller No. 1, Fig. 2.4-14. Since the write address has been incremented, the read address is no longer equal to the write address, (R≠W). Also at this time the Hold-1 flip-flop is in its 0 state as the Hold-1 register is empty, and the transfer was a 1-4 type. These three conditions will cause the received four words in the buffer to be loaded into the "reply time" register (HR-1), "power, monopulse, LR" register (HR-2), "reply time" register (HR-3), and the "DATA" register (HR-4) of the Hold-1 register, as indicated by States 1, 2, 3, and 4 respectively. At the end of State 4, the Hold-1 flip-flop will be set signifying a "Hold-1 register full" condition, and State 0 is re-entered. Note that during States





LOAD 5-10	LOAD 1-4	9316 COUNTER
1	1	DO NOTHING
0	1	PARALLEL LOAD $10_{10}$
1	0	PARALLEL LOAD $12_{10}$
0	0	NOT DEFINED

Fig. 2.4-16. Memory writer logic.

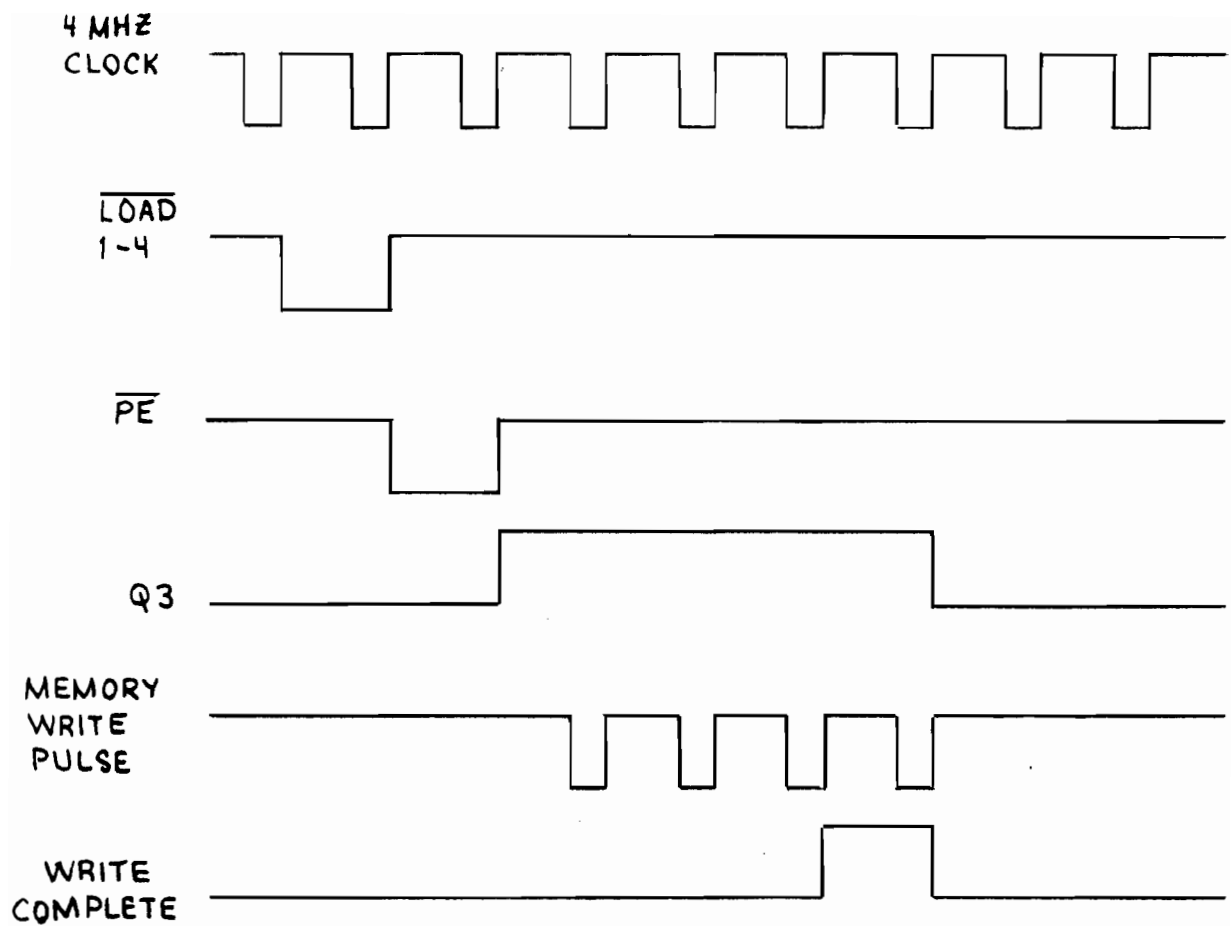


Fig. 2.4-17. Waveforms for the memory write logic during LOAD 1-4 sequence.

0, 1, 2, 3, and 4, if a "write" pulse is decoded, Controller No. 1 will process the buffer write sequence before going into the next state. Also, note that State 0 can go directly to State 4 if the received data is a 5-10 type. This is due to the fact that word 5 through 10 of a DABS reply contains only the reply data. State 4 will enable these words to be loaded into the "DATA" register so that the reply data will be encoded and assembled by the Reply Assembler (see Hold-1 Register in Fig. 2.4-13.).

There are three conditions which must be met before the reply words can be transferred from the Hold-1 register to the Hold-2 register. These conditions are: hold-1 register is full (Hold-1 flip-flop = 1), Hold-2 register is empty (Hold-2 flip-flop = 0), and the special bit is "0" (the special bit comes from Controller No. 3; it is set at the end of the reply.) Two other bits also affect the transfer. They are the type X and type Y bits. A type X or type Y bit is an additional bit attached to each word of the reply. Its sole purpose is to indicate whether the reply word is a 1-4 or 5-10 type as discussed previously. Type X comes from the Hold-1 register, and when it is transferred to the Hold-2 register, it becomes a type Y.

Referring to Fig. 2.4-15, if the type X bit is equal to a "1", indicating that the word in Hold-1 register is a 1-4 type, Controller No. 2 will cause the contents of the entire Hold-1 register to be parallel loaded into Hold-2 register (State 6). On the other hand, if type X is equal to a "0", signifying a 5-10 type, only the contents of the "data" register will be transferred (State 5). In this way, after a 1-4 transfer, the Hold-2 register will contain the reply control words and a reply data word. If the reply is a DABS type, the remaining 6 words of reply data will be transferred as a 5-10 type. During the 5-10 transfer only the "data" register is affected. The control words will remain unchanged in the Hold-2 register during the entire reply transmission.

Each time data of either type is transferred to the Hold-2 register, the Hold-2 flip-flop will be set and the Hold-1 flip-flop will be cleared by Controller No. 2.

Resetting the Hold-1 flip-flop will cause additional data, if any, stored in the buffer to be immediately transferred into the Hold-1 register; and subsequently into the Hold-2 register when the Hold-2 flip-flop is cleared.

The purpose of the two stage buffering done by the Hold-1 and Hold-2 registers is to enable replies to follow one another with minimum separation in time. This is effected by having the Hold-2 register and the reply assembler processing the last reply data word for the current reply concurrently with the next reply being loaded into the Hold-1 register. When the current reply is terminated, the next reply is loaded into the Hold-2 register and Reply Assembler for immediate processing.

ARIES can simulate three type of DABS replies: all-call, surveillance and Comm-B. The latter is 112 bits long as compared to 56 bits for the other two. For simplicity, all DABS replies are transferred to the target generator in a ten word block. Therefore, there will be three "don't care" words in the all-call and surveillance replies (for data formats, see Volume 3 of this document).

The reply generator ignores these three words in the two types of short replies. This is done by State 3 in Fig. 2.4-15. To understand how this is accomplished, assume that the reply assembler is processing the 7th (the last) word of a short reply, and that therefore, the 8th, 9th, and 10th word of the same reply are stored in the Hold-2 register, Hold-1 register and the current read location of the buffer respectively. Normally, at the end of transmission of each word the Hold-2 flip-flop is cleared by Controller #3. This allows the next data word to be propagated to the Hold-2 register. However, at the end of the 7th word of a short reply, the Hold-2 flip-flop is not cleared, therefore the Hold-1 and Hold-2 flip-flops remain set and the type X and type Y bits are zero since the 8th and 9th words are 5-10 type. These conditions signify that the current reply is either all-call or surveillance. When the last bit of the current reply is finally transmitted, as indicated by the Special Bit = 1, State 3 will be entered, causing the Hold-1 and Hold-2 flip-flops to be cleared, and the current read location of the buffer to be skipped.

#### 2.4.1.2.3.2 Delay-To-Trigger Timing Circuit

The delay to trigger timing circuit consists of:

- (a) a 16-bit range counter
- (b) a comparator, and
- (c) the Go-1 and Go-2 flip-flops

The first control word in a reply, either ATCRBS or DABS, is a 16-bit reply time. It specifies the precise moment that a reply should be transmitted by the reply assembler. The reply time is held in the Hold-2 register (along with other control words) and is continuously compared to the 16-bit range clock in the delay-to-trigger circuit. When the value of the reply time and range clock are equal, a trigger pulse is generated by the comparator, which causes the reply to be transmitted.

The Go-1 and Go-2 flip-flops must also be set before a trigger pulse can be forwarded to the Reply Assembler. The reason that these two flip-flops are needed is as follows:

Since the comparator continuously compares the content of the range clock and the reply time register whether reply data is present or not, unwanted trigger pulses may be generated. To prevent this from happening, gating logic is required. This is the purpose of the Go-1 flip-flop. It is set whenever a "1-4" is loaded into the Hold-2 register, and reset when a compare pulse is generated. The Go-1 flip-flop remains reset until another "1-4" is loaded.

There is a period of at least 56  $\mu$ s (dead zone) between the beginning of the last DABS reply and the next ATCRBS/DABS All-Call interrogation when no reply should be generated. (A DABS reply generated in this 56  $\mu$ sec interval would overlap the interrogation, and the interrogation scheduling software prevents this).

During this period, ATCRBS replies for the upcoming interrogation are being loaded into the reply generator. Following the usual loading sequence, the reply may be in the Hold-2 register waiting to be transmitted. If the reply time happens to be equal to the range counter at this time, the reply will be triggered prematurely before the arrival of the All-Call interrogation. The Go-2 flip-flop prevents this from happening. 56  $\mu$ s before the next All-Call interrogation, a "56- $\mu$ s" pulse is generated by the UIT, (see Section 2.2.2), causing the Go-2 flip-flop to be reset and thus preventing a compare pulse from being sent. When the All-Call interrogation is finally decoded by the Receiver, the Time of Arrival (TOA) pulse from the Receiver will set the Go-2 flip-flop. TOA also causes the range counter to be zeroed, synchronizing the reply generator with the Receiver.

#### 2.4.1.2.3.3 Reply Assembler

The reply assembler is the heart of the reply generator. It takes reply data from the "data register" of the Hold-2 register and converts it to serial pulses simulating the PAM/PPM modulation format of the ATCRBS/DABS replies.

The reply assembler consists of:

- (a) a shift register that performs parallel to serial conversion,
- (b) a parity encoder for DABS replies,
- (c) PAM modulator for ATCRBS replies,
- (d) PPM modulator for DABS replies,
- (e) Sync logic for timing,
- (f) State Controller No. 3 for controlling the reply assembler, and
- (g) multiplexers for selecting DABS or ATCRBS replies.

State Controller No. 3 consists of a 512 word x 23 bit ROM, and a multiplexing circuit shown in Fig. 2.4-18.

The ROM is coded to generate all necessary signals to control the shift register, PPM modulator, and PAM modulator. The control signals for each type of reply (ATCRBS, LONG DABS and SHORT DABS) are available from the ROM at all times. The multiplexing circuits select the correct type for the current reply. The multiplexing circuits are controlled by the least significant four bits of the third control word of a reply (see the CRG in Volume 3 of this document).

To generate a reply, the reply assembler must first receive a trigger pulse from the Delay-to-Trigger Timing Circuit. Depending upon the reply type (from the Hold-2 Register) the trigger pulse will preset the Sync Logic to a starting address for the Controller No. 3 ROM. All replies, regardless of type, will terminate at location 512 of the ROM (last location). Only the starting addresses are different as each type of reply (ATCRBS, SHORT DABS and LONG DABS) has a different time period. Table 2.4-9 shows the starting address for each reply and Fig. 2.4-19 shows the Sync Logic. Note that the Sync Logic disables itself after the end of each reply and can start only after a trigger pulse is received.

#### 2.4.1.2.4 Reply Simulation

##### 2.4.1.2.4.1 DABS Replies

To understand how a DABS reply is simulated, consider the following sequences. Assume that the reply to be transmitted is a Comm-B type (long) as specified by the reply type in Hold-2 register, that Multiplexer C (Fig. 2.4-13) will be selected for the DABS reply, and also that control signals for long replies from the Controller No. 3 ROM are selected. (See Fig. 2.4-18). As soon as a trigger pulse is received, the Sync Logic will preset the Controller No. 3 ROM address to 23.

Fig. 2.4-20 shows the control signal waveforms for the long DABS reply. Beginning at address 23, the DABS preamble pulses are the first to be sent, followed by clearing of the shift register and parity encoder register for the up-coming reply. The first 16-bits of reply data are then loaded into the shift register from the Hold-2 register. The shift register is clocked every 1  $\mu$ sec so that the data are shifted out at a rate corresponding to the DABS reply data rate (1 MBPS). Loading and shifting for the shift register are controlled by the Shift/Load control line. Note that every time a new set of 16-bit data is loaded into the shift register, the Hold-2 flip-flop will be cleared by the CLEAR Hold-2 control line, so that consecutive 16-bit data words may be propagated from the memory to the Hold-1 register and to the Hold-2 register as described previously.

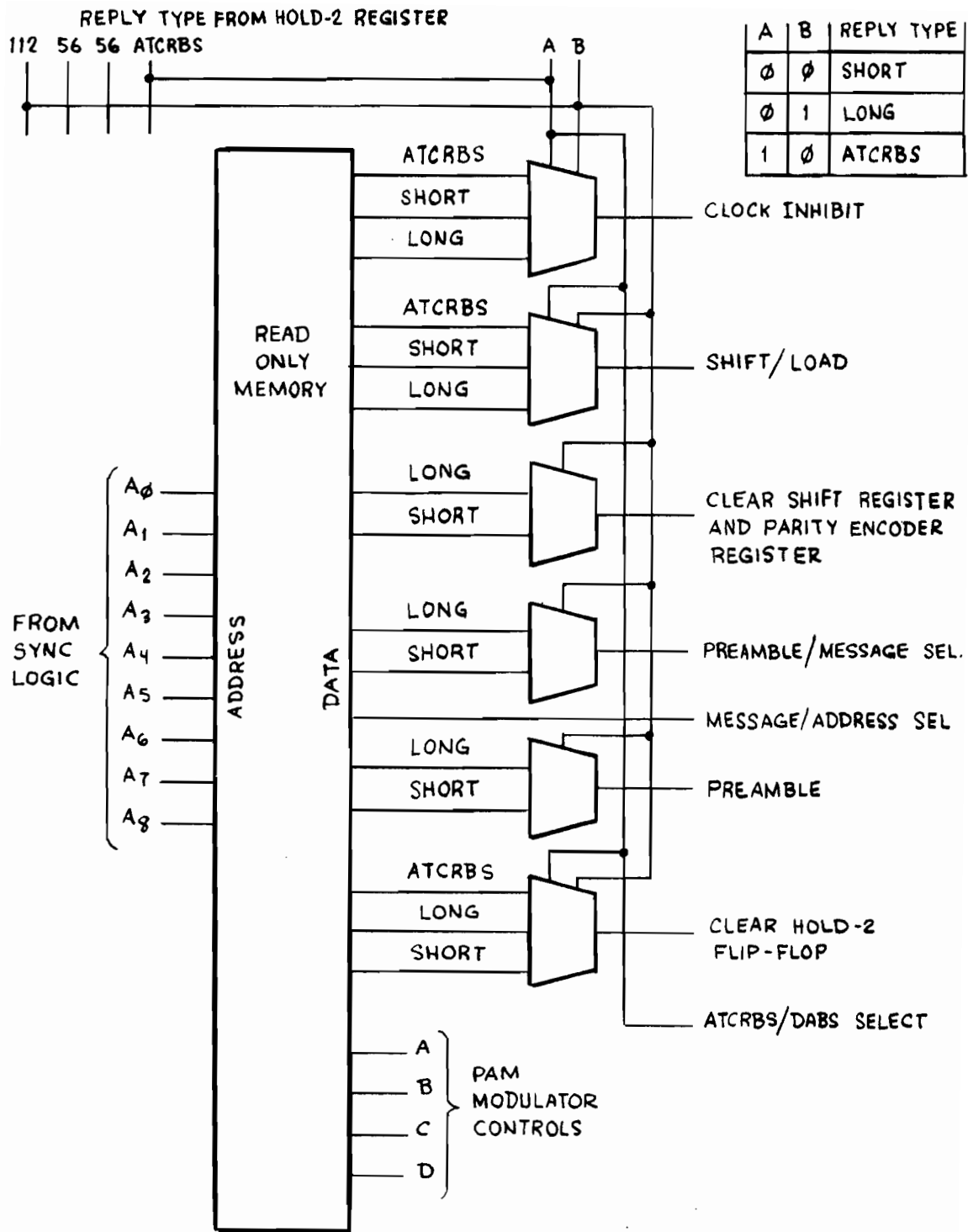


Fig. 2.4-18. Controller No.3.

TABLE 2.4-9

<u>Reply Type</u>	<u>Starting Address of ROM (Decimal)</u>
LONG (Comm-B)	23
SHORT (All/Call)	247
SHORT (Surveillance)	247
ATCRBS With SPI	407
ATCRBS Without SPI	425



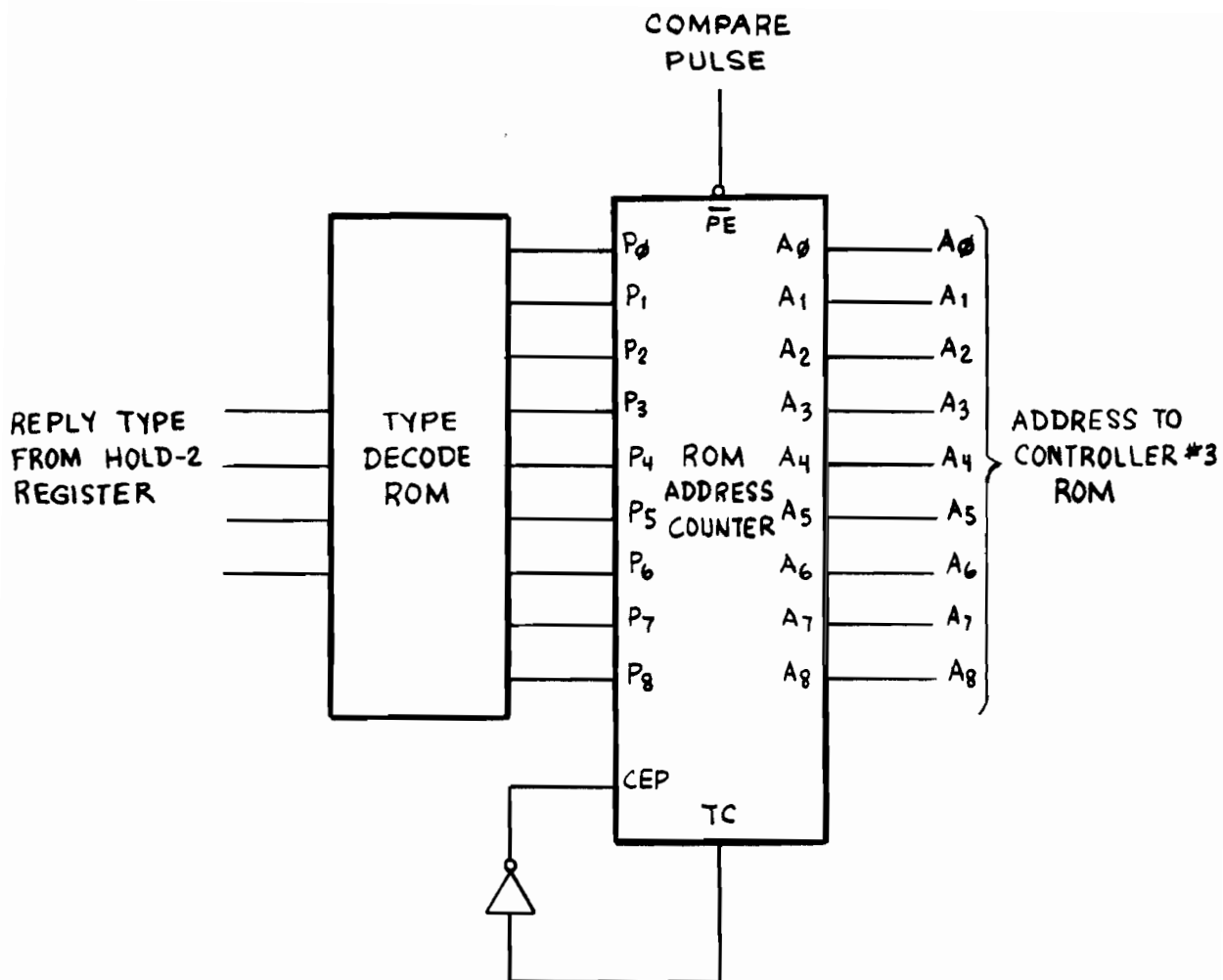


Fig. 2.4-19. Sync logic.

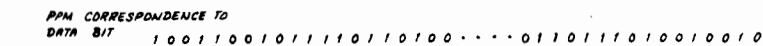


Fig. 2.4-20. Control signal waveforms, DABS reply (long).

Also shown in Fig. 2.4-20 (lower half) are the PPM waveforms corresponding to the sample reply data bit as generated by the PPM modulator. The sources of the waveform can be identified in Fig. 2.4-21.

Each reply generator is designed with a 24 bit parity encoder. Its purpose is to take the message portion of the reply and convert it to 24 parity check bits and to add the remaining 24 address bits to these parity check bits in modulo-two fashion. The 24 parity check bits are computed and stored in the parity encoder register as seen in Fig. 2.4-22. (For more detail on DABS downlink encoding see "DABS Downlink Coding" by J.T. Barrows, FAA-RD-75-61, Lincoln Laboratory Report No. ATC-48). At the end of the message bits (88 for a DABS long reply), Multiplexer A will be selected so that the last 24 bits of the reply will be coming directly from the parity encoder rather than from the shift register. Multiplexer A is controlled by the message/address control line.

Short DABS replies (all-call and surveillance) are simulated in a similar way, except that the Controller No. 3 ROM is started in location 247, and since there are only 32 message bits, Multiplexer A will output the address from the parity encoder after 32 shifts.

#### 2.4.1.2.4.2 ATCRBS Replies

The ATCRBS reply format (see Fig. 2.4-23) consists of 16 pulses: two bracket pulses,  $F_1$  and  $F_2$ , 12 data bits, an X bit, and a SPI bit. The width of each pulse is 450 ns, and pulse separation is 1000 ns except between  $F_2$  and SPI which is 3900 ns.

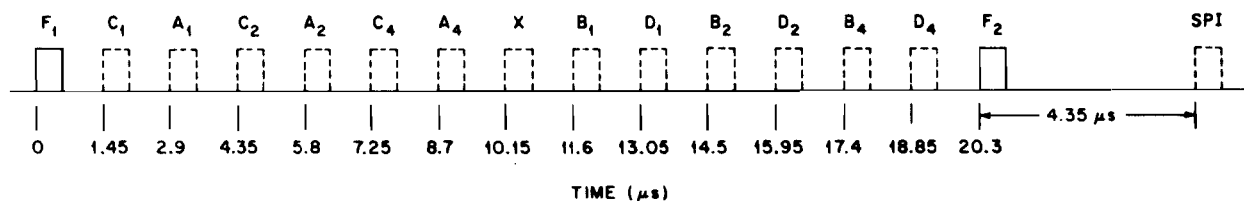


Fig. 2.4-23. ATCRBS reply.

To simulate an ATCRBS reply, output from the message shift register is modulated by the PAM modulator, and Multiplexer C selects the ATCRBS replies.

To understand how the PAM modulator operates, refer to Table 2.4-10. Columns 1, 2, and 3 of the Table show each ATCRBS reply pulse and its timing relative to  $F_1$ . Since the ARLES 4 MHz clock period is 250 ns, it is not possible to match the timing of the leading edges of these pulses directly, and a delay scheme must therefore be used. This scheme performs as follows: Controller No.3 ROM generates a 500 ns wide pulse with a leading edge that is as close to that of the actual pulse as the resolution of the 4 MHz clock allows, but which does not occur later than the desired leading edge. Leading edge delays for these pulses are shown in Column 4. Delay is then added to set the leading edge of the simulated pulse at its proper value. These delays vary from 0 to 200 ns in 50 ns increments as shown in Column 5.

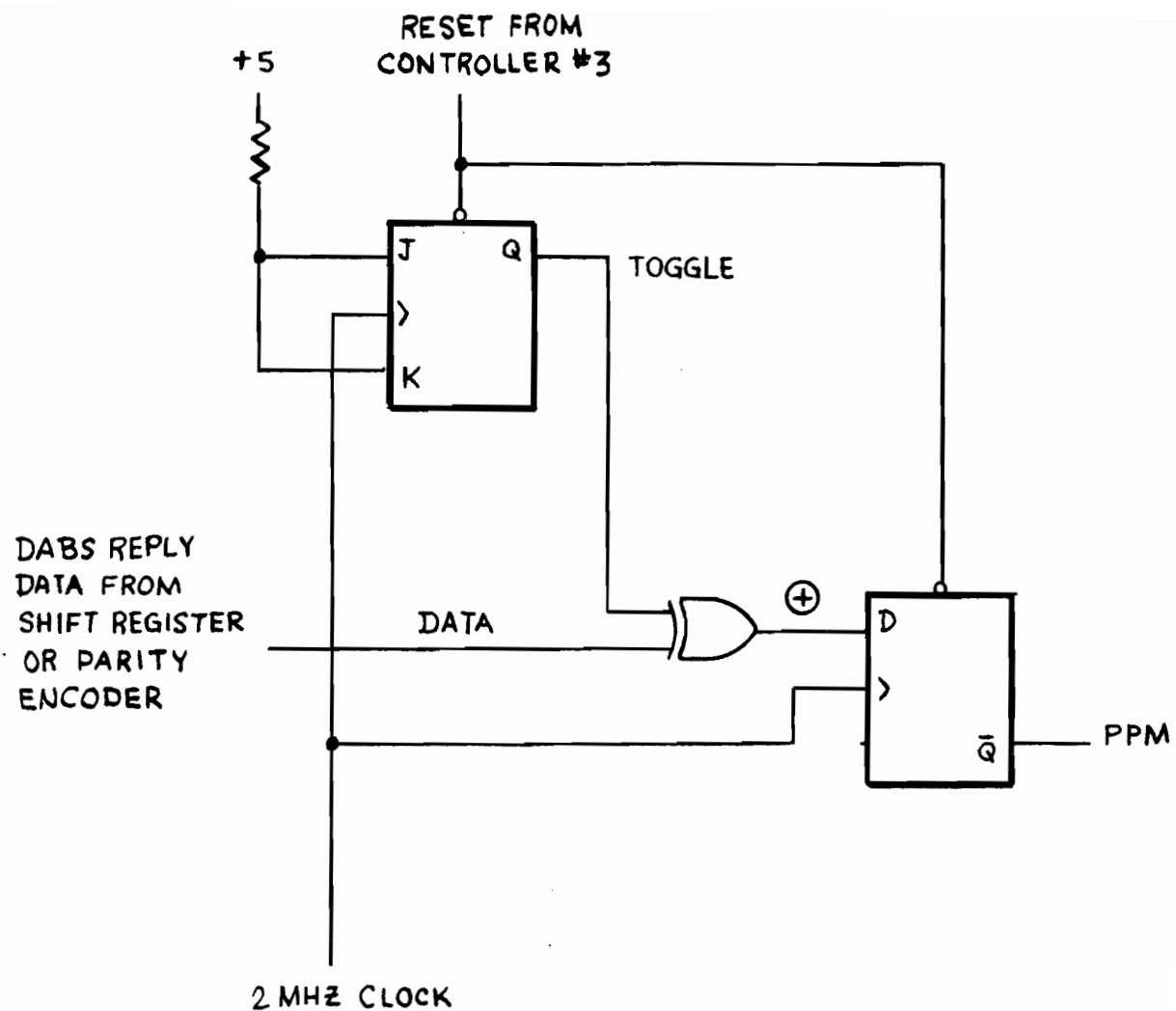


Fig. 2.4-21. PPM modulator.

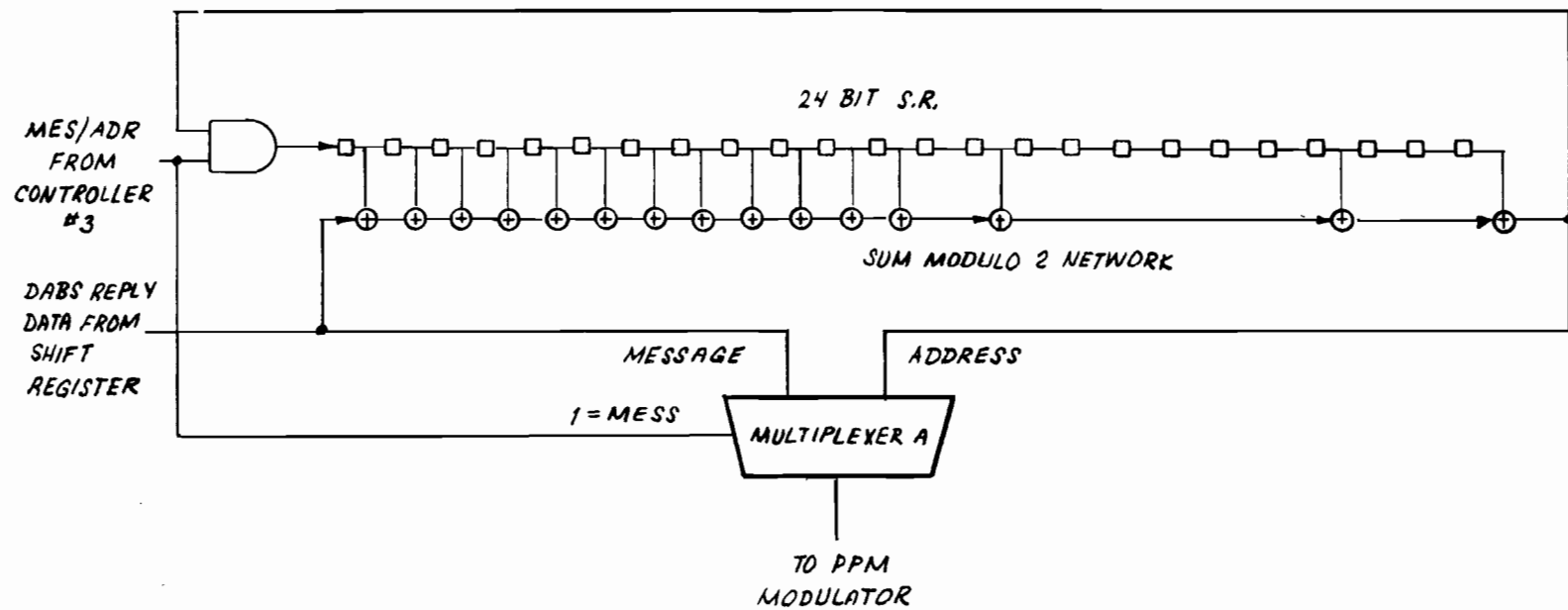



Fig. 2.4-22. Parity encoder.

TABLE 2.4-10  
TIMING FOR AN ATCRBS REPLY

(1)	(2)	(3)	(4)	(5)
ATCRBS PULSE	ACTUAL PULSE TIMING	SHIFT REGISTER OUT	CONTROLLER #3 ROM OUT	PROGRAMMABLE DELAY
F <sub>1</sub>	0	-.25	0	0
C <sub>1</sub>	1.45	1.25	1.25	.200
A <sub>1</sub>	2.90	2.50	2.75.	.150
C <sub>2</sub>	4.35	4.00	4.25	.100
A <sub>2</sub>	5.80	5.50	5.75	.050
C <sub>4</sub>	7.25	7.00	7.25	0
A <sub>4</sub>	8.70	8.50	8.50	.200
X	10.15	9.75	10.00	.150
B <sub>1</sub>	11.60	11.25	11.50	.100
D <sub>1</sub>	13.05	12.75	13.00	.050
B <sub>2</sub>	14.50	14.25	14.50	0
D <sub>2</sub>	15.95	15.75	15.75	.200
B <sub>4</sub>	17.40	17.00	17.25	.150
D <sub>4</sub>	18.85	18.50	18.75	.100
F <sub>2</sub>	20.30	20.00	20.25	.050
All times are in $\mu$ sec			<div style="text-align: center;">  </div> <p>The sum of these two values must be equal to the value in Column (2).</p>	

The PAM modulator circuitry is shown in Fig. 2.4-24. Signals A, B, C, and D are coming from Controller No. 3 ROM: A is a 500 ns pulse. To generate a 450 ns wide ATCRBS pulse, a 50 ns delay element and an AND gate are used. The pulse is then fed into a tapped delay line so that appropriate delays may be selected by the multiplexer. Selection is controlled by the B, C, and D control lines. Output from the multiplexer is then ANDed with the ATCRBS reply data from the shift register. Note that the data from the shift register must also be shifted out at an appropriate time so that the modulating pulse from the multiplexer will occur at the center of the data bit. The timing for the shift register is shown in Column 3.

ATCRBS replies with the SPI pulse appended after  $F_2$  are generated in a similar way.

#### 2.4.1.3 Controlled Reply Generator Interface

The Controlled Reply Generator Interface (1) serves as a link between the CPU and the Controller, and (2) provides buffer space for the CPU to store replies as they are generated. It consists of two 16 bit x 1024 word memories and their read/write address counters, a reply counter for each memory, and associated control logic, multiplexers and driver circuitry. As shown in Fig. 2.4-25 the CRG Interface consists of two sets of components, each labeled with suffix -1 or -2.

At any instant of time, one set of components with the same suffix (-1 or -2) is labeled as primary and the other as secondary (i.e., primary memory, primary reply counter etc.). These assignments are reversible under program control.

To understand how the CRG interface works it is necessary to understand how ARIES handles discrete and ATCRBS interrogations. This is described fully in Vol. 1, Section 3.0 of this document, but for convenience is repeated here.

Discrete Interrogations are handled as they are received. The ARIES receiver interrupts the CPU, which then processes the interrogation data and generates a reply, which is immediately sent to the CRG. This can be done because the specified DABS transponder turn-around time of 128  $\mu$ sec allows sufficient processing time.

In the case of ATCRBS interrogations, however, the specified turn-around time is only 15  $\mu$ sec. This does not allow sufficient time for the CPU to process these interrogations "on the fly". Fortunately, the time and mode of ATCRBS interrogations is predictable, as the sensor follows a fixed interrogation pattern. Therefore, the ARIES system can prepare the complete set of ATCRBS and All-Call replies for a given interrogation in advance of the interrogation actually being sent. When the interrogation is received, these precomputed replies are transmitted and ARIES begins generating the replies for the next ATCRBS interrogation.

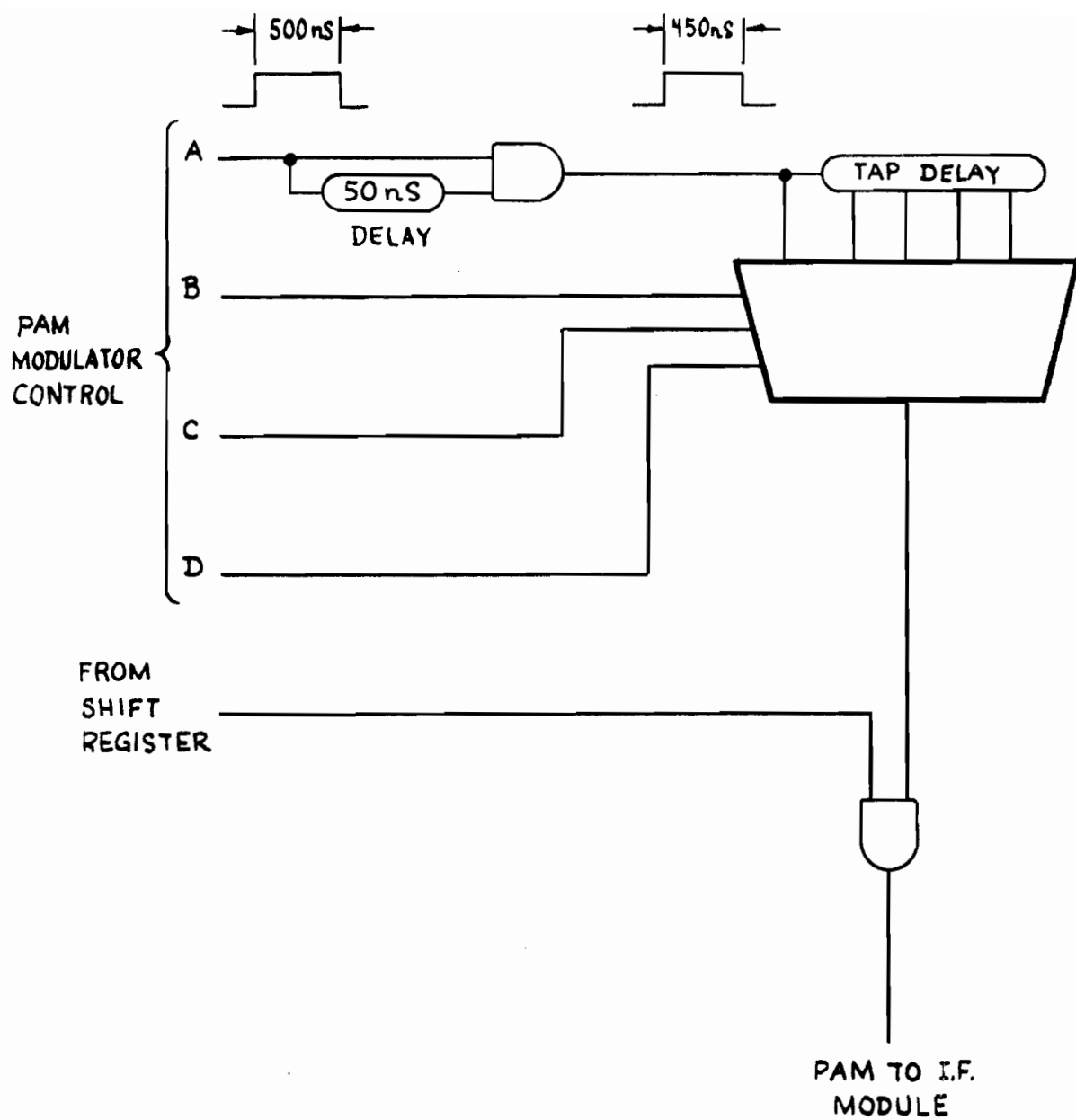


Fig. 2.4-24. PAM modulator.



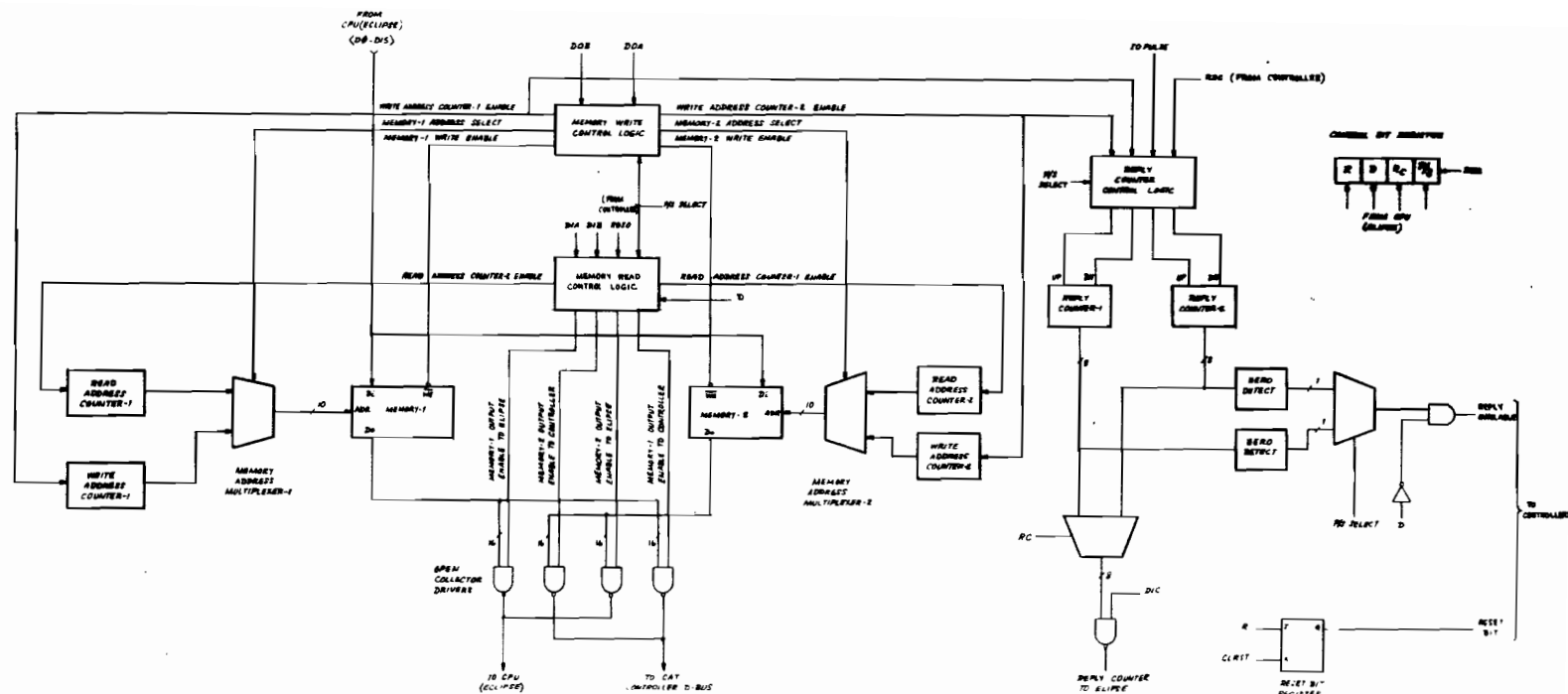


Fig. 2.4-25. CRG controller interface.

The secondary memory is the storage area for these pre-computed ATCRBS and All-Call replies. Since the controller can only read from the primary memory, these will not be transmitted. When the ATCRBS interrogation is received, the software switches the buffer primary/secondary status. Since the ATCRBS replies are then in the primary buffer the controller immediately begins sending them to the CATs.

DABS discrete replies are to be transmitted immediately, and so the CPU must store them directly into the primary memory. Each memory has associated with it a reply counter. This is incremented each time a complete reply is entered into the buffer by the CPU and decremented each time the controller reads a reply. Just as the controller can only look at the primary buffer, it can only access the primary reply counter.

To write a word into the primary memory, the CPU executes a DOA instruction. Words are stored at successive locations in the buffer. Similarly, to store a word in the secondary memory, a DOB instruction is executed. To store the last word of a reply (word 4 for ATCRBS, word 10 for DABS All-Call or discrete replies) either the DOAP or DOBP instruction must be used, as this increments the reply counter for the buffer being 'stored in' and stores the last reply word.

#### 2.4.1.3.1 Memory and Memory Address Counter

Each memory used in the interface is an array of 16 1K x 1 bipolar memory chips. It is capable of storing 102 DABS replies (10 words each) or 256 ATCRBS replies (4 words each). There are separate read and write address counters, so that the memory can be read and written independently. The memory address multiplexer selects whether the read or write address is used. All address counters are 10-bit circular type; and are reset to zero when an IORST is executed by the CPU.

#### 2.4.1.3.2 Memory Write Control Logic

Each DOA and DOB instruction executed by the CPU has an associated DOA or DOB pulse (Readers not familiar with the I/O instructions should refer to the Data General Interface Designer's Reference Manual, Data General Publication No. 015-000031).

These two pulses, as shown in Fig. 2.4-26, drive two two-input multiplexers,  $M_1$  and  $M_2$ . The select inputs of these two multiplexers are of opposite polarity, so that at any one time the output of one multiplexer will be a DOA and the other a DOB. The output with the DOA pulse is always forwarded to the memory designated as primary at the time. Similarly the DOB output

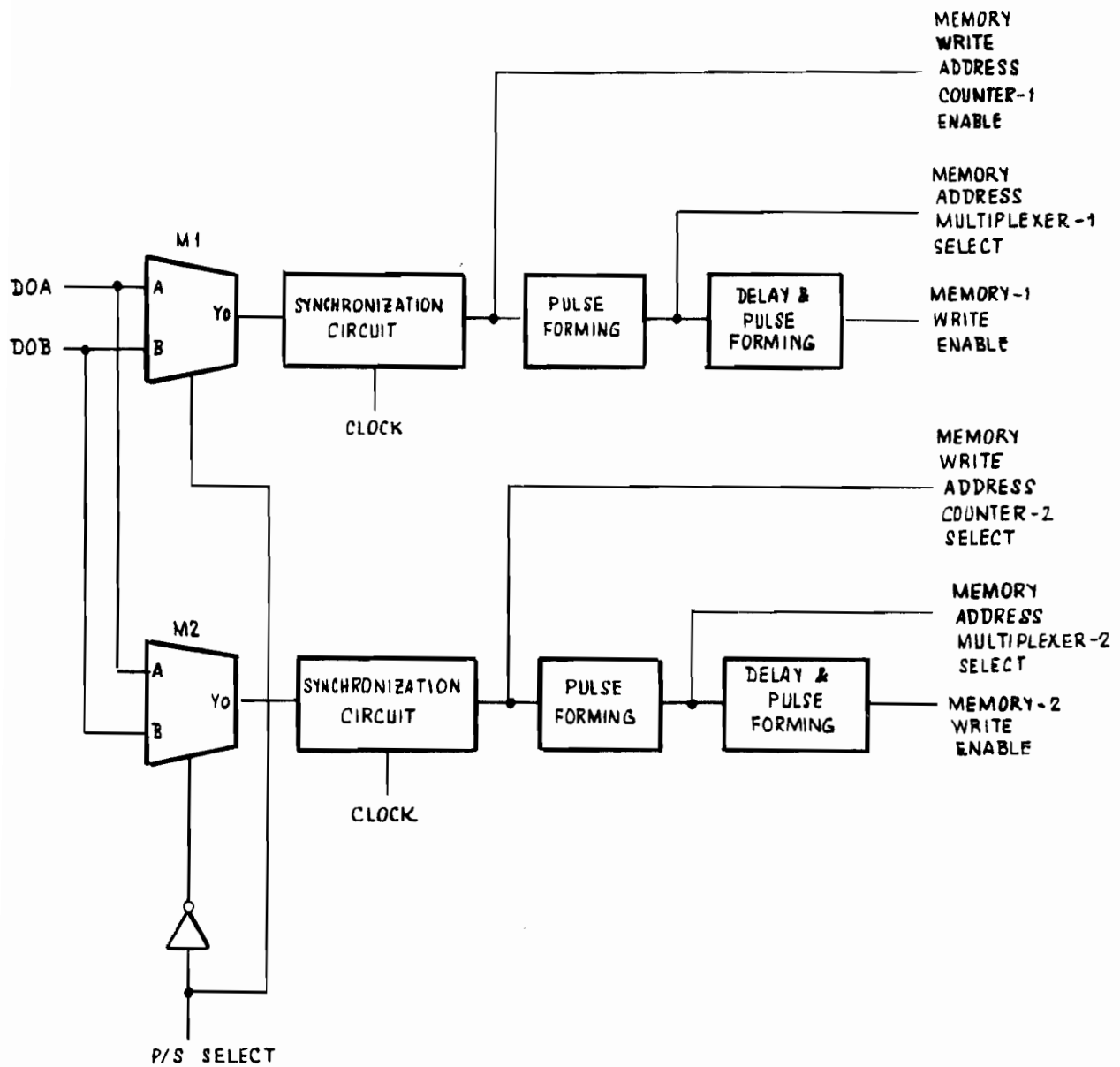


Fig. 2.4-26. Memory write control logic.

is always forwarded to the secondary memory. The primary/secondary (P/S) select line as controlled by the CPU determines the roles of the two memories. When P/S select is set to a logical '1', memory-1 will be chosen as primary. Likewise, Memory-2 will be selected as primary when P/S is set to logical '0'.

To understand how the CPU and the controller can access the primary memory without interference, consider the following. The controller inputs data from the memory during the positive transition of the clock (low to high). Therefore, the period immediately after this transition can be used for writing if the memory access time is small and the read and write address can be set up fast enough (i.e., within one clock cycle). As it turns out, this condition can be met if high speed Schottky memory and logic are used. The 82S11 memory chips used in the design have a worst-case access time of 45 ns providing high enough speed for reading and writing in one 250 ns clock cycle.

A typical writing cycle to the primary memory is as follows. (Refer to Fig. 2.4-27). When the CPU executes a DOA instruction, the data is forwarded to the input of the primary memory, and a DOA pulse is generated. This pulse, as selected by the 2 input multiplexer ( $M_1$  or  $M_2$  depending on P/S) is synchronized to the clock, forming a pulse one clock period wide. This write-counter enable pulse, in addition to causing the memory write-address counter to increment at the end of this period, is also applied to a pulse forming circuit to produce a 50 ns wide memory address select pulse. This pulse enables the memory address multiplexer (see Fig. 2.4-25) to output the write address from the write-address counter to the memory. After a period of 15 ns to allow the multiplexer to settle, a memory write enable pulse, also derived from the 50 ns address select pulse, is applied to the memory enabling it to strobe in the data. At the end of the memory address select pulse, the memory address multiplexer returns to its normal output, allowing the read address from read address counter to be applied to the memory. The whole writing process requires only 50 ns, allowing the remaining 200 ns to set up the read cycle by the controller if needed. Writing into the secondary memory (DOB), involves the same process. The only difference is that reading from the controller is prohibited.

#### 2.4.1.3.3 Memory Read Control Logic

The controller input bus (D-Bus) is tied to two sets of open-collector drivers (refer to Fig. 2.4-25). The input of one set is connected to Memory-1 and the other to Memory-2. At any particular instant only one set of drivers is connected to the primary memory and hence will be allowed to output data to the D-Bus. The driver output enable is generated in the memory read control logic. Similarly, two set of drivers are also used for sending data to the CPU.

Since the controller can only access data from the primary memory, logic is required to multiplex the data from the two memories. The memory read

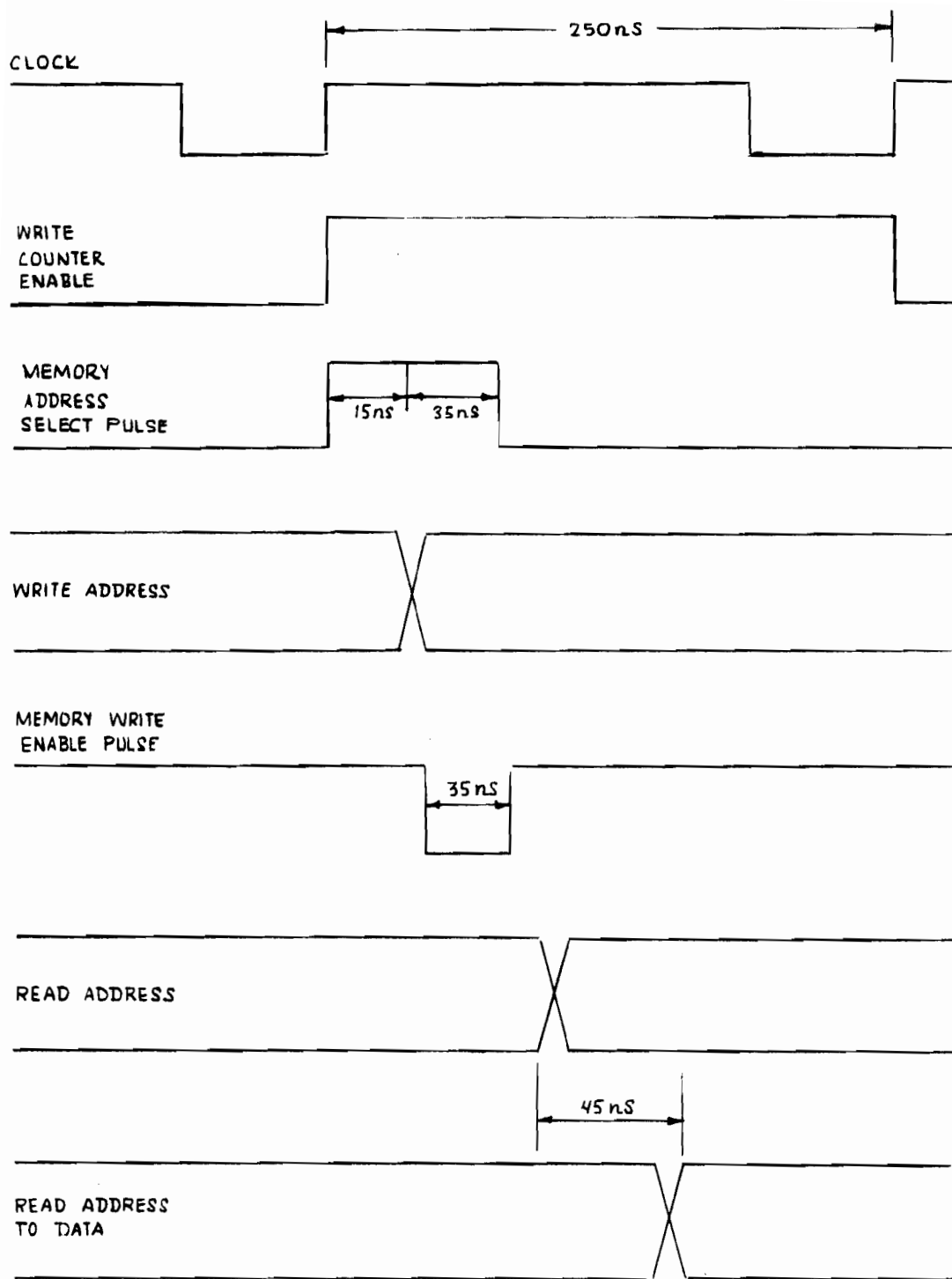


Fig. 2.4-27. Memory writing cycle waveforms.

control logic provides this function. To read a word from the memory, the controller must send a RDIO pulse to the CRG interface. This pulse, as shown in Fig. 2.4-28, is multiplexed by two, two-input multiplexers. Only the multiplexer selected by the P/S bit can forward the RDIO pulse to enable the output driver and increment the read address counter.

During the normal operating mode, the CPU is prohibited from reading the memory. However, it is sometimes highly desirable for diagnostic purposes to check the contents of the memory after known data has been written into it. To enter the diagnostic mode, the CPU sets the D-bit (see Fig. 2.4-25) to a logic '1'. Setting this bit will prevent the controller from reading the memory, and allow the CPU to increment the memory and the read address counter after each operation. DIA and DIB instructions from the CPU are required to read the primary and secondary memory respectively. As in the DOA and DOB instructions, each DIA and DIB instruction is also associated with a DIA and DIB pulse. These two pulses are multiplexed so that a DIA pulse will be forwarded to enable the output driver for the primary memory, likewise, the DIB pulse will enable the output driver from the secondary memory. These two pulses after being synchronized to the system clock are also used to increment their respective read-address counters at the end of each read operation.

#### 2.4.1.3.4 Reply Counters

To keep track of the numbers of replies stored in the memories, two reply counters are provided. These counters are 8 bit up-down type, and at any instant one is associated with the primary memory and the other with the secondary memory, hence they are referred to as primary reply counter, and secondary reply counter. The CPU can increment these two counters at any time. However, the controller can decrement only the reply counter designated as primary at the time. To increment the primary (secondary) reply counter the CPU executes a DOAP (DOBP) instruction. Decrementing the primary reply counter can be accomplished by a RCD pulse from the controller.

Fig. 2.4-29 shows the control logic that performs this function. Assuming Memory-1 is selected as the primary memory (i.e., P/S = 1) the J-K flip-flop will be set by the DOA pulse allowing the IOPULSE to increment Reply Counter-1. Also, the RCD pulse from the controller will be allowed to decrement reply counter -1. If the RCD pulse and I/O pulse occur simultaneously, the 'exclusive or' gate will prevent Reply Counter-1 from counting in either direction.

In a similar fashion, the CPU can increment Reply Counter-2 by a DOBP. Note that the RCD pulse is applied to only Reply Counter-1, so that Reply Counter-2 cannot be decremented by the controller until the CPU changes the status of the P/S bit.

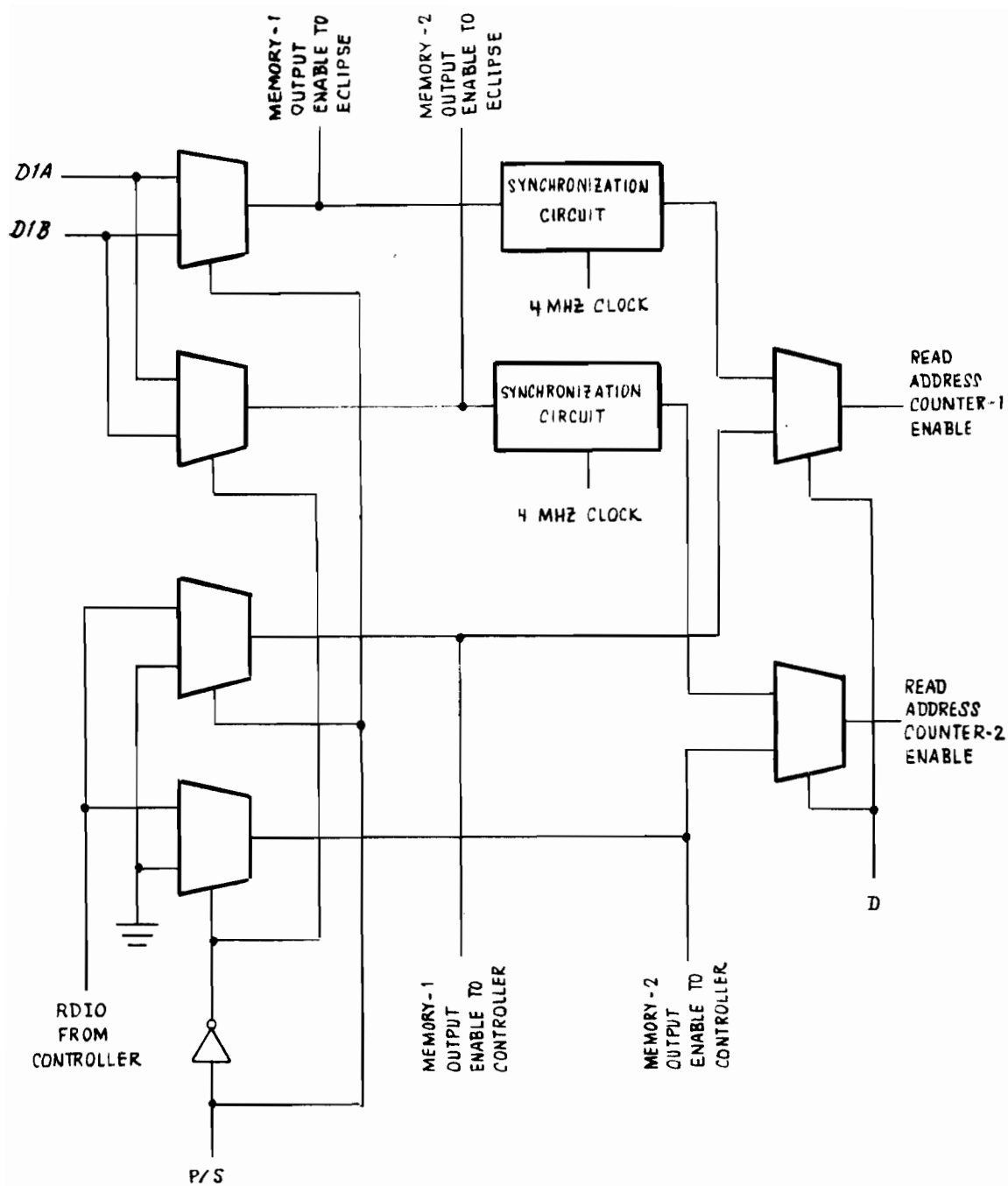


Fig. 2.4-28. Memory read control waveforms.





The output of the reply counters is connected to a zero detection circuit, enabling the controller to check the status of the primary reply counter. During the diagnostic mode, the reply counter will always appear to be zero to the controller. Each reply counter can be reset to zero by an IORST pulse from the CPU. Also, the reply counter can be read by the CPU via the DIC instruction while the RC bit is set appropriately. (The RC bit selects which of the reply counters will read: 0=Primary, 1=secondary).

#### 2.4.1.3.5 Reset Bit Register

The CPU can reset all the CAT's via an IORST instruction, however, this is sometimes not desirable, since IORST will also reset the rest of the system. To circumvent this problem, the Reset (R) control bit is provided. When the controller detects the setting of this bit by the CPU it will initialize all the CAT's. During the resetting process, the controller will reset the R Bit to zero.

#### 2.4.1.3.6 Control Bits

The four interface control bits are software programmable by means of the DOC instruction (see Vol. 3 of this document). Their operation is summarized as follows:

RC: Counter primary/secondary select bit. This bit determines which of the reply counters will be read by a DIC instruction.

D: Diagnostic mode control bit.

0: Normal mode. DIA, DIB do not read buffer data.

1: Diagnostic mode. DIA and DIB can be used to read buffer data. The reply counter is disconnected from the controller.

PS: Buffer primary/secondary select bit. Selects which of the two buffer memories is to be the current primary memory and which is to be the secondary memory.

R: Setting this bit causes the controller to reset the CAT's. This bit is set to 0 by the controller when the reset is complete.

#### 2.4.2 Fruit Generator (FG)

The Fruit Generator consists of the FG controller, three Fruit ARIES Targets (FAT's), the Random Process Generator and the FG interface.

#### 2.4.2.1 Fruit Generator Controller (FGC)

The hardware of the Fruit Generator Controller is identical to that of the Controlled Reply Controller. The firmware resident in the read-only memory does differ, however (refer to Vol. 2, Appendix A of this document for the FG program listing).

#### 2.4.2.2 Fruit ARIES Targets (FAT'S)

The Fruit ARIES Target hardware is also identical to the Controlled ARIES Target hardware so it will not be described again. However, it is important to notice the differences in setting and clearing the Go-2 flip-flop for both cases. In the case of the CAT'S, Go-2 is set by the "TOA" pulse from the Receiver and reset by the "56  $\mu$ s" pulse from the UIT as described previously.

In the case of the FAT'S, the Go-2 flip-flop is not set by the TOA pulse but by the FTOA compare pulse from the adjacent FAT. All FAT'S are connected in daisy-chain fashion as shown in Fig. 2.4-30. The compare pulse from a FAT also clears its own Go-2 flip-flop. This scheme allows the FAT'S to fire in sequence. It is important that they do so as the controller also sends replies to the FAT'S sequentially. Therefore, the Controller can guarantee that the sum of any three successive reply times for the FAT'S will be greater than 20.3  $\mu$ s duration of an ATCRBS reply with no SPI, thus no FAT will request that another reply be started until the current reply is complete.

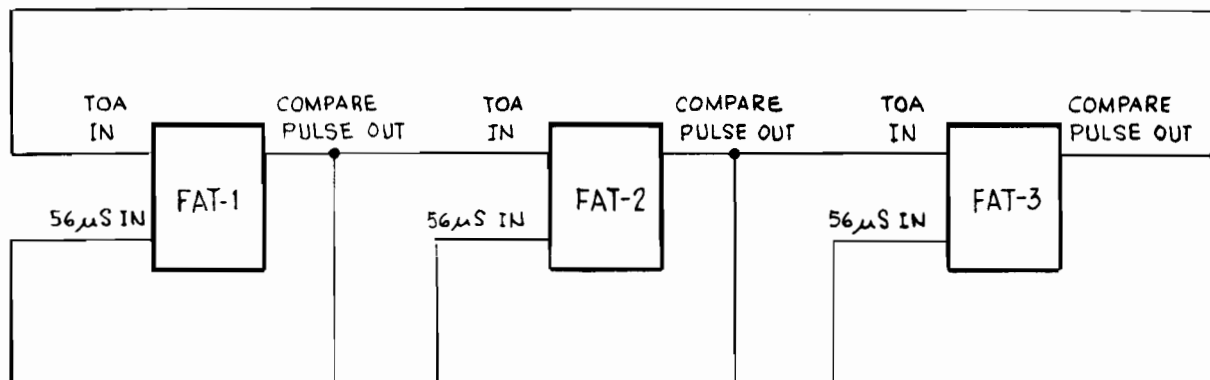
#### 2.4.2.3 Random Process Generator

##### 2.4.2.3.1 General Description

The Random Process Generator (RPG) constructs groups of control words which determine the power, off-boresight angle (including sign), mainbeam/sidelobe status, and ATCRBS code of each fruit reply. The RPG is also responsible for controlling fruit arrival times. To simulate real fruit each of these fruit parameters must vary randomly over its spectrum of possible values.

There are three fruit parameters which are used to control the operation of the RPG: average fruit rate, fraction of fruit in the mainbeam (as opposed to being in the sidelobes), and fraction of fruit having a known, fixed, ATCRBS code (as opposed to having all code bits randomly selected). These three parameters are controlled by the operational software via the FAT Controller.

As shown in Fig. 2.4-31 the RPG consists of six control parameter generators, each responsible for generating one of the above FAT control word parameters. Also shown are the interface circuits in which the control word groups are assembled and through which they are transferred from RPG to the FG Controller.



DAISY-CHAINING OF THREE FAT'S \*

\* THE WIRING IS DONE ON THE BACKPLANE  
OF THE DIGITAL BOX RATHER THAN ON THE  
BOARD ITSELF.

Fig. 2.4-30. Daisy-chaining of three fruit ARIES targets (FAT's).

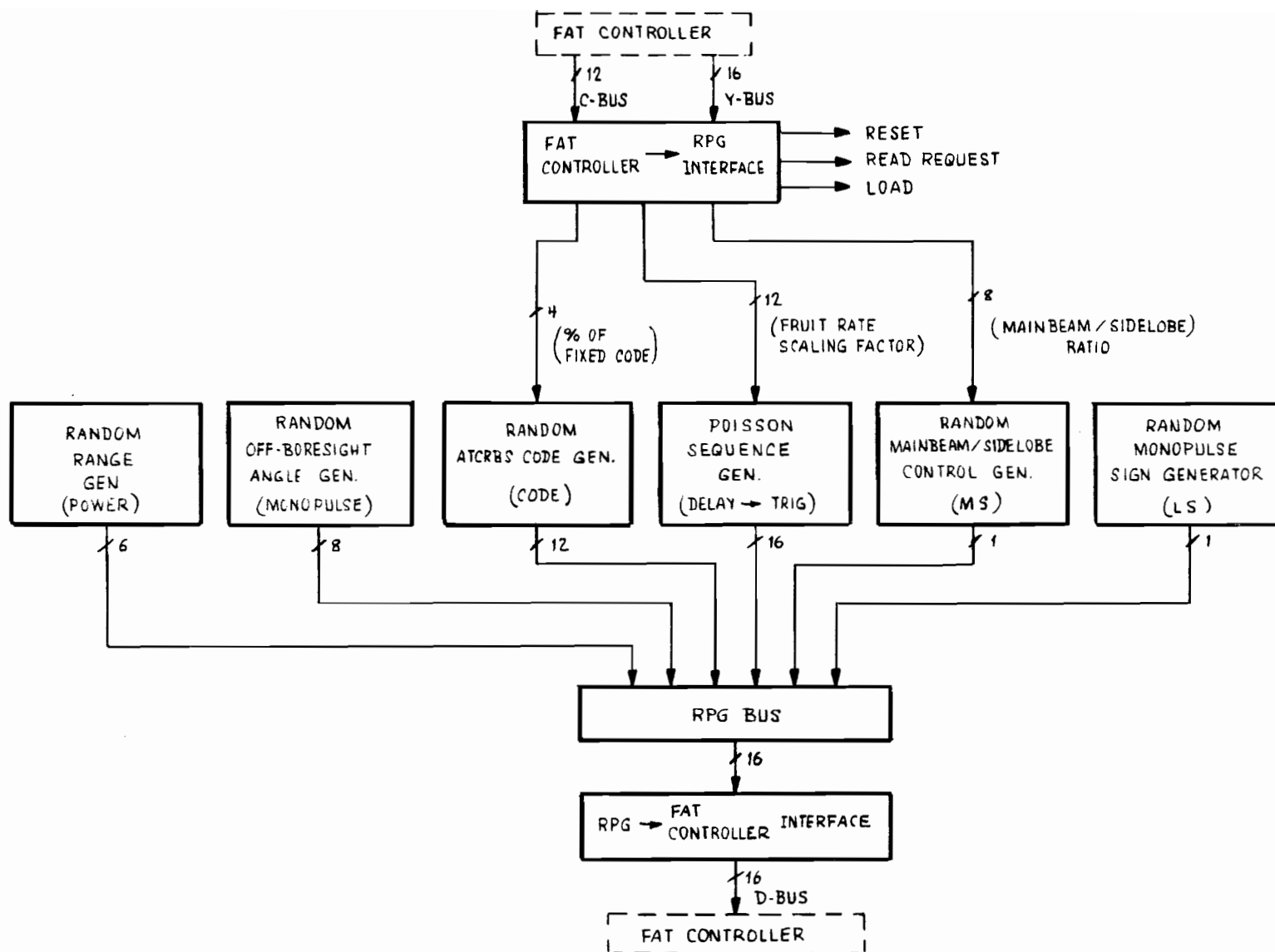


Fig. 2.4-31. RPG block diagram.

The data buses interconnecting the RPG and Controller are the C-Bus, the D-Bus and the Y-Bus. Control commands transmitted via the C-Bus from Controller to RPG are:

- RESET: Resets all random number generators to a predetermined value, and generates 16 new inter-arrival times.
- READ: Causes a fruit control word group to be output to the Controller.
- LOAD: Changes the average fruit rate, mainbeam/sidelobe ratio, and ratio of fixed to random code.

Reading and loading are done over the D-Bus and Y-Bus respectively. Data formats for the fruit control mode (fruit parameters) are given under FRUIT REPLY GENERATOR in Vol. 3 of this document.

#### 2.4.2.3.2 Random Number Generators\*

All random fruit parameters output by the RPG are generated using pseudo-random number generators employing shift registers. Such devices, called "linear maximal sequence code generators", consist of a basic shift register to which Exclusive - Or gates have been added. Outputs from the register stages are input to the Exclusive - Or gates, and the gate outputs fed back to other stages of the register to form single or multiple closed loops. When the register is clocked in a normal manner the output of each stage forms a pseudo-random binary sequence.

The feed-back connections of the Exclusive - Or gates determine the random sequence and whether the sequence is maximal or not. The maximal sequence generator cycles through all its  $2^n - 1$  possible states, except zero, where  $n$  = number of stages. In the RPG, all random number generators are connected to give the maximal sequence; also, only shift register stages numbered 7, 11, 15 and 17 are used. The 7-bit generator used in the RPG is illustrated in Fig. 2.4-32.

Since zero is an illegal state (causing a lock up condition), the generator must not start in this state. This requires a special form of reset circuit. Fig. 2.4-33 shows the circuit. When a reset is decoded by the RPG, counter-1 and counter-2 will be preset to "zero" and "5" respectively. Note that one of

---

\* (See The Design of Digital Systems by John B. Peatman, p. 412).

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	TOGGLE INPUT TO N <sup>th</sup> FLIP-FLOP	RANDOM NUMBER GENERATOR I.D. IN RPG
7											T*	D	D	D	D	D	D	$T_7 = X_1$	1, 2
11							T	T	D	D	D	D	D	D	D	D	D	$T_{11} = X_1$	3, 5
15			T	D	D	D	D	D	D	D	D	D	D	D	D	D	D	$T_{15} = X_1$	4, 6
17	T	T	T	T	T	D	D	D	D	D	D	D	D	D	D	D	D	$T_{17} = X_1$	7

T = TOGGLE FLIP FLOP

D = DELAY FLIP FLOP

\* EXCLUSIVE - OR GATE IS ELIMINATED IN THE FEED - BACK PATH  
BECAUSE TOGGLE FLIP FLOP IMPLEMENTS AN EXCLUSIVE-OR FUNCTION  
BETWEEN ITS INPUT AND ITS OUTPUT.,  $Q_{n+1} = Q_n \oplus T_n$

$T_n$  = TOGGLE INPUT  
 $Q_n$  = PRESENT STATE  
 $Q_{n+1}$  = NEXT STATE

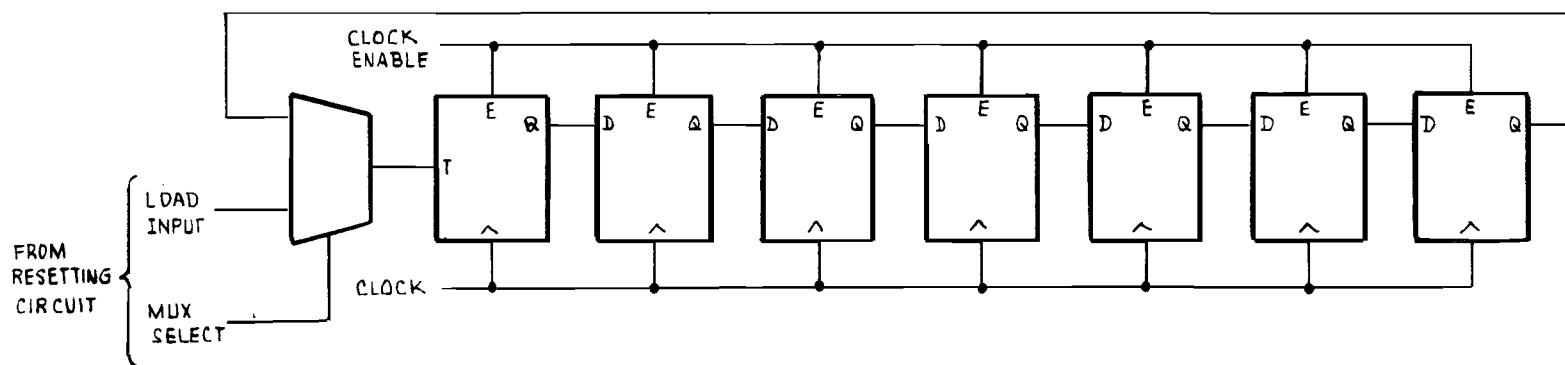


Fig. 2.4-32. 7-bit pseudo random number generator (RNG1, RNG2).

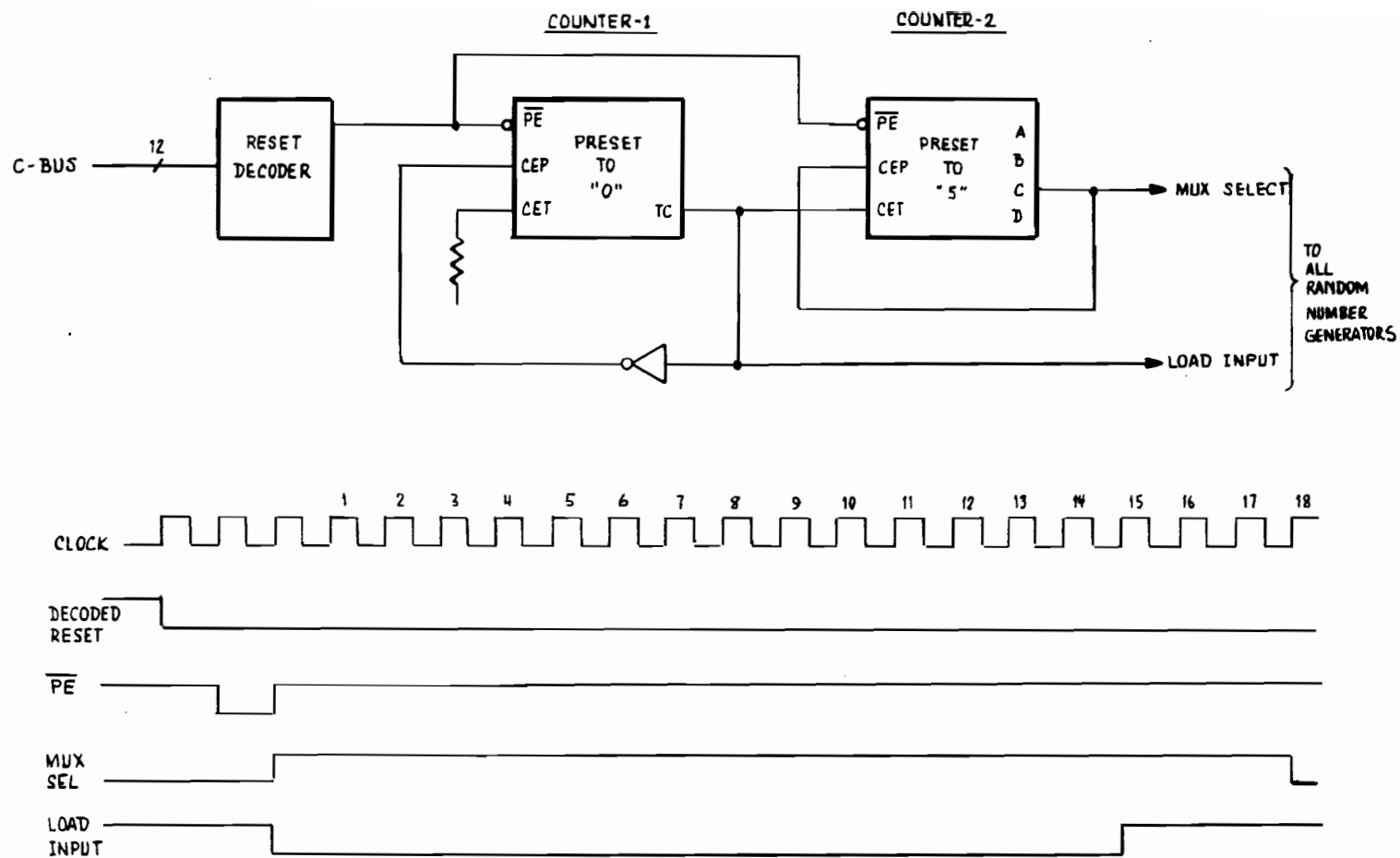


Fig. 2.4-33. Resetting circuit and waveforms.

the enable inputs of counter -2 is tied to the overflow of counter - 1, therefore, counter - 2 is idle until counter -1 overflows. At that time counter -1 is frozen at maximum count (15) and counter -2 starts to increment from 5. When it reaches 8, it also freezes producing the waveform shown in Fig. 2.4-33. Note also that when the mux select from the resetting circuit is high, the feed-back path in the shift register is disconnected (see Fig. 2.4-32), since the multiplexer has selected the other input (load input from resetting circuit). With the feed-back path disconnected, the random number generator operates as an ordinary shift register, allowing a known number to be loaded from the reset circuit. All random number generators are reset to "111000----00"; the number of zeros being equal to n-3, where n = number of stages. Resetting all generators to a known value at the beginning also allows RPG diagnostics to be performed, as a CPU simulation of the RPG can also be started with the same set of conditions. RPG data can be read by the CPU using the RPG diagnostic mode of the FG interface (see Section 2.4.2.4).

#### 2.4.2.3.3 Generation of Random Range (Power) Control Parameter

The specification for the power level of ARIES generated fruit is as follows: "The mainlobe fruit will have a power level referred to the sensor RF port determined as follows:

$$P_{ml} = -20 -20 \log r_{ml} \text{ (dBm)}$$

where  $r_{ml}$  is random and uniformly distributed between 1 and 100". This is equivalent to specifying a random mainbeam power level from -20 dBm to -60 dBm.

Also stated is that the sidelobe fruit shall have a power level referred to the sensor RF port determined as follows:

$$P_{sl} = -55 -20 \log r_{sl} \text{ (dBm)}$$

where  $r_{sl}$  is random and uniformly distributed between 1 and 32. This is equivalent to specifying a random sidelobe power level from -55 dBm to -85 dBm.

Two random number generators are used in the RPG to meet these requirements. The Random Range Generator provides a 6-bit power field distributed according to either the mainbeam or sidelobe power specification. A single-bit from the Random M/S Control Generator determines whether the reply is to be a mainlobe or sidelobe reply. The range (or power) field is used to control the main power attenuator of the analog portion of the fruit generator (FAT), giving power levels from -20 to -60 dBm. If the mainlobe/sidelobe (M/S) bit is set, an additional 35 dB of attenuation is switched in producing a sidelobe output range of -55 to -95 dBm. Since a uniform distribution in range does not give a uniform power distribution, it is necessary



to use a separate 15-bit random number generator, (RNG 4 in Fig. 2.4-34) of which 8-bits are used to address a 512 word "Power ROM". The 9th address bit for the ROM is the mainbeam/sidelobe bit, and this is used to choose between the two distributions. This ROM modifies its uniform input distribution to provide the desired 6-bit power control signal.

The way in which the entries in the 512 word Power ROM are derived is explained in Appendix B.

#### 2.4.2.3.3.1 Generation of Mainbeam/Sidelobe Control Parameter

To generate the mainbeam/sidelobe control bit, an 8-bit uniformly distributed random number is first generated by RNG3. This is then compared with the output from the 9-bit M/S ratio register. The result as determined by the comparator  $A > B$  output is used as the M/S control bit. A and B are the outputs from RNG3 and the M/S ratio register respectively, and if the condition  $A > B$  is true, the reply will be sidelobe. Therefore,  $B = 0$  represents all replies being in the sidelobes, and  $B = 256$  represents all replies in the mainbeam. Numbers greater than 256 will be treated as 256.

The content of the M/S ratio register, as with the other two fruit parameters, is programmable by the CPU. See Vol. 3 of this document for the data format.

#### 2.4.2.3.4 Generation of Monopulse Angle Control Parameter

The simulated fruit is characterized by a uniformly distributed monopulse off-boresight angle. The off-boresight angle control parameter is generated using an 11-bit random number generator (RNG5, see Fig. 2.4-34). Its output drives a monopulse ROM, included in the present implementation in case it should be necessary in the future to modify the distribution of this parameter. It serves no purpose at present.

#### 2.4.2.3.4.1 Generation of Monopulse Angle Sign Control Parameter

The monopulse angle sign parameter, LR, determines whether simulated fruit comes from the left or right of boresight. The parameter is a single bit, generated by random number generator number 4 (RNG4 is also used in the generation of the range (power) control parameter). It provides for half of all replies being to the left of boresight and half to the right of boresight. The LR parameter performs its function by switching a phase shifter in the analog portion of the fruit generator.

#### 2.4.2.3.5 Generation of Random ATCRBS Codes

There are 16 pulses in an ATCRBS reply. In the order of transmission they are:

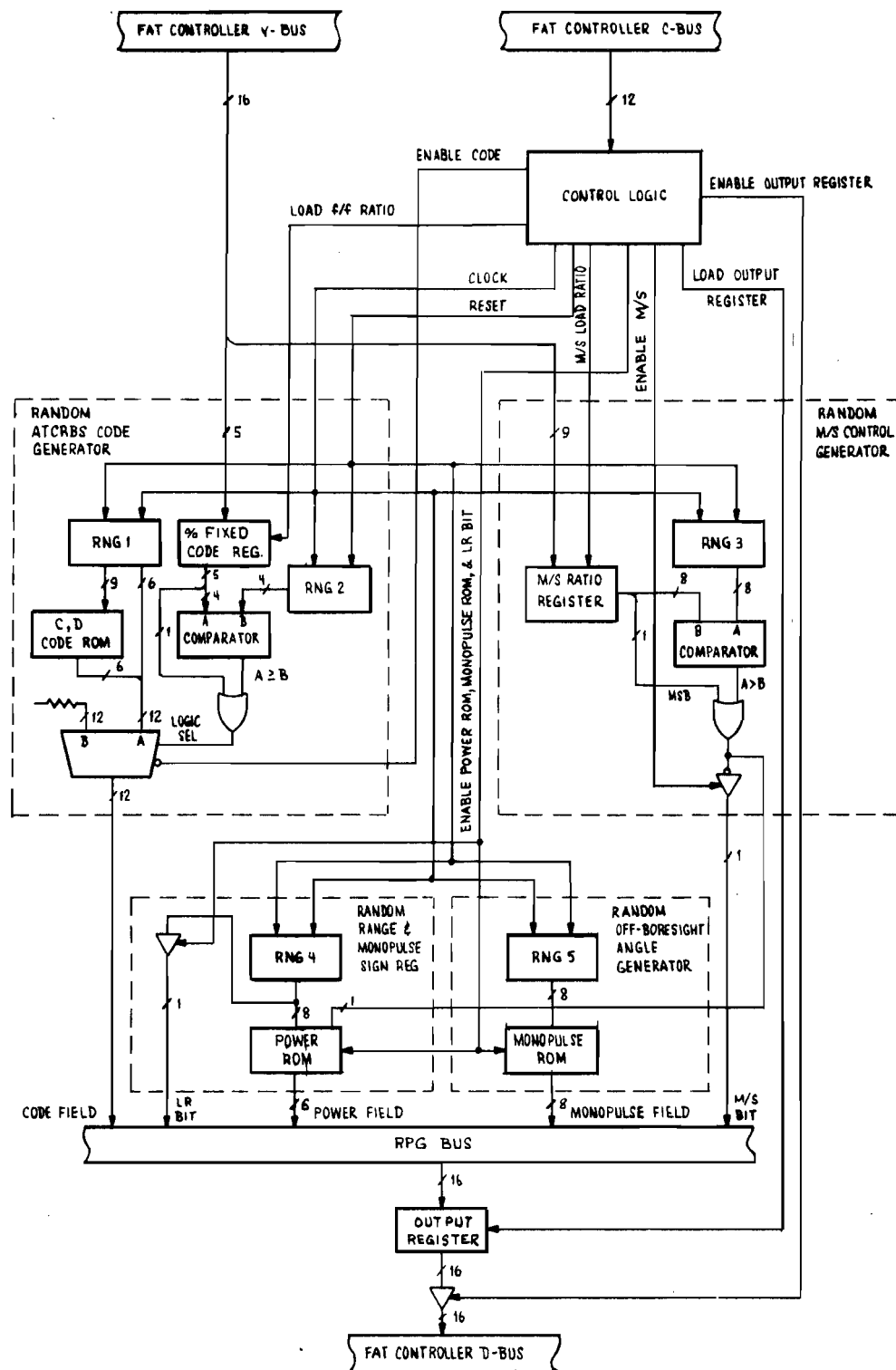


Fig. 2.4-34. ATCRBS code, range, monopulse, LR and M/S Generators.

$$F_1 C_1 A_1 C_2 A_2 C_4 A_4 X B_1 D_1 B_2 D_2 B_4 D_4 F_2 SPI.$$

The  $F_1$  and  $F_2$  pulses are always present; ARIES assumes that the X and SPI pulses are always absent for fruit replies.

The distribution of the remaining bits depends on the type of interrogation causing the reply. For Mode A interrogations, it is assumed that all combinations are equally likely. In particular, this assumes that discrete codes are much more likely than non-discrete. This is not a correct model of any particular environment, due to the fact that the A(=  $A_4 A_2 A_1$ ) and B bits are chosen from a small set of codes assigned to any given control area. However it is undesirable from a system testing point of view to model a particular environment that closely.

The Mode C (altitude) code consists of two Gray codes, one encoding 100 foot increments and one encoding 500 foot increments. The C bits encode the 100 foot increments using the C values, 1, 2, 3, 4, 6. These values are treated as being equally likely.

The A and B bits are used to encode altitude intervals which cover the typical altitude distribution of targets in such a way as to make all combinations almost equally likely. Exceptions are the  $A_1$  and  $A_2$  bits which are on for the intervals from 15000 to 47000 feet ( $A_1$ ) and 17000 to 23000 and 39000 to 55000 feet ( $A_2$ ), where the traffic densities are lower. ARIES ignores this factor and generates uniformly distributed A and B bit values.

The D bits are zero for almost all Mode C replies. The exception is that targets above 32000 feet have the  $D_4$  bit set. The  $D_2$  bit is also a legal altitude bit, but is only set above 63000 feet and so is not of interest. To approximate these conditions, it is assumed that approximately 15% of the replies are from targets above 32000 feet, corresponding to a moderately high level of overflight traffic, and differing from the Los Angeles Basin distribution. Los Angeles is felt to have an anomalously low level of overflights due to its geographic location.

To generate uniformly distributed A and B bits for both altitude and Mode A codes, the 6 lower bits of a 15 bit uniform random number generator are used (RNG1 - see Fig. 2.4-34). The remaining 9 bits are used to generate the C and D bits. Since the C and D bits have a non-uniform distribution, a conversion ROM (the "C, D code ROM"), containing the proper distribution is included. 342 of the 512 ROM entries (about 2/3) contain uniformly distributed values for C and D generated by simply counting through all values a sufficient number of times to fill the space. These provide the simulated data for Mode A replies. The remaining 170 entries have C values uniformly distributed over the set 1, 2, 3, 4, 6. These are again generated by succes-

sively counting through this set until the entries are filled. 25 of these entries have the  $D_4$  bit set, the others have  $D = 0$ . The 6 bits from RNG1 and 6 bits from the C,  $D$  ROM constitute the 12 bit random ATCRBS code.

The fraction of random ATCRBS codes generated (the rest have a fixed, known code) is determined by the CPU. This 5 bit ratio, a fruit control parameter, is loaded into the "% fixed code" register. With the LSB equal to 100/16 percent of fixed replies, the 5 bits can represent from zero ( $\emptyset$  = no fixed replies) to 100% (16 = all fixed replies). Numbers above 16 are treated the same as 16.

To generate the proper percentage, the lower 4 bits of the "% fixed code" register are compared with 4 bits of the output of a 7 bit uniformly distributed random number generator (RNG2). The results as determined by the comparator are used to select which of the 2 inputs to the multiplexer will be used as the ATCRBS code. When the output of RNG2 is smaller, the ATCRBS code generated will be used. Otherwise, all one's will be selected. The "all one's" is not the fixed code, but is merely used to signify to the FAT Controller that it should replace the "all one's" with the fixed code before a reply is sent to the fruit generator. The FAT controller receives the fixed code from the CPU as part of the fruit generation parameters (see the data format in Vol. 3 of this document). The MSB of the % fixed code ratio is used to override the comparator. It is OR-ed with the output of the comparator. When this bit is set, all replies generated are fixed code.

Reading of the random ATCRBS code generator by the FAT-Controller will be discussed in Section 2.4.2.3.7, "Reading the RPG".

#### 2.4.2.3.6 Generation of the Delay-to-Trigger Control Parameter

The delay to trigger control parameter determines when a fruit reply should be triggered by a fruit generator (FAT). To realistically simulate fruit, their arrivals as a function of time should be characterized by a Poisson or exponential distribution. The way in which the RPG generates delay to trigger times with such a distribution by combining the outputs of two random number generators with uniform distribution is explained in Vol. 2, Appendix C of this document.

Fig. 2.4-35 is a block diagram of the overall Poisson sequence generator. The 15 bit random number generator RNG6 provides 10 bit uniformly distributed

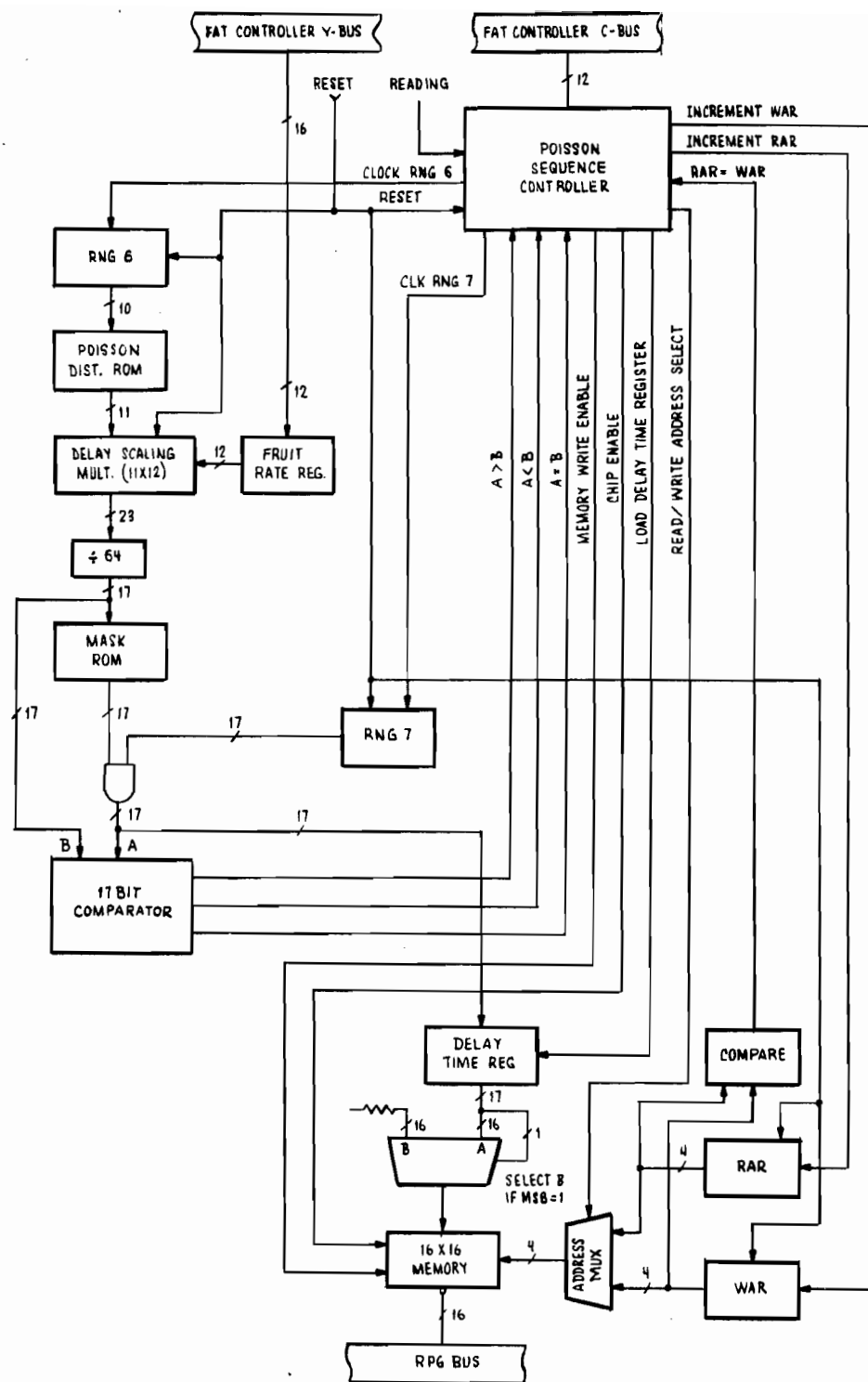


Fig. 2.4-35. Poisson sequence generator.

numbers. This is used to obtain an 11 bit value,  $t_n$ , from the Poisson distribution ROM.  $t_n$  is then multiplied by a 12 bit scaling factor coming from the fruit rate register and controlled by the CPU. It is loaded into the fruit rate register as part of the fruit generation parameters from the FAT Controller. The values of the scaling factor lie between 64 and 4096, corresponding to fruit rates of 64,000 to 1000 respectively. (Scaling factor = 4096/desired fruit rate in thousands).

The output of the multiplier is a 23 bit number. The low order 6 bits are ignored, however, effectively dividing the result by 64. Therefore, the effective scaling factor lies between 1 and 64.

The resulting 17 bit number represents the upper bound,  $t_n$ , of the horizontal strip as discussed in Appendix C. To choose a number from zero to this upper bound, another uniform random generator, RNG7, is used. This 17 bit generator is iterated and compared to  $t_n$  until a number less than equal to  $t_n$  is found. For small  $t_n$ , this may take a significant amount of time, as most of the numbers generated by RNG7 are large. To get around this, the output of RNG7 is masked, eliminating most of the numbers greater than  $t_n$ .

The mask is generated by the mask ROM as follows: if the MSB of  $t_n$  has the value of  $2^m$  then  $(2^{m+1} - 1)$  is output from the mask ROM to the AND gates, allowing all values equal or smaller than  $(2^{m+1} - 1)$  to get through.

Once a number is found it is temporarily stored in the delay-time register. Before it is transferred into the buffer memory, the 17 bit number must be first reduced to a 16 bit quantity as the range-clock in ARIES is only 16 bits long. (The 17 bits are required so that there will be a large enough range to hold the longest delay to trigger values generated at the lowest fruit rate.) To accomplish this, the MSB of the 17 bit number is used to select a multiplexer, so that if the MSB is set, all one's will be output from the multiplexer, otherwise the lower 16 bits of the 17 bit number will be output. This scheme effectively reduces all numbers greater than  $2^{16} - 1$  to  $2^{16} - 1$ , creating a truncated exponential distribution with a spike at the end.

At this point, if the RPG is not currently being read by the FAT Controller, the number will be immediately transferred to the buffer memory. The above process is repeated until the memory is full (contains sixteen delay of trigger times).

This entire sequence of operations is controlled by the Poisson Sequence Controller. The state diagram for the controller is illustrated in Fig. 2.4-36. Not shown in the diagram is the reset input. When reset occurs, the Poisson Sequence Controller will return to state 0 from any state, and remain in this state until the reset is completed. Also, the write address register (WAR) is reset to one count ahead of the read address register (RAR), so that as soon as reset is completed, the Poisson Sequence Controller will start generating sixteen new delay to trigger times. During states 1, 2, 3, and 4 if the read

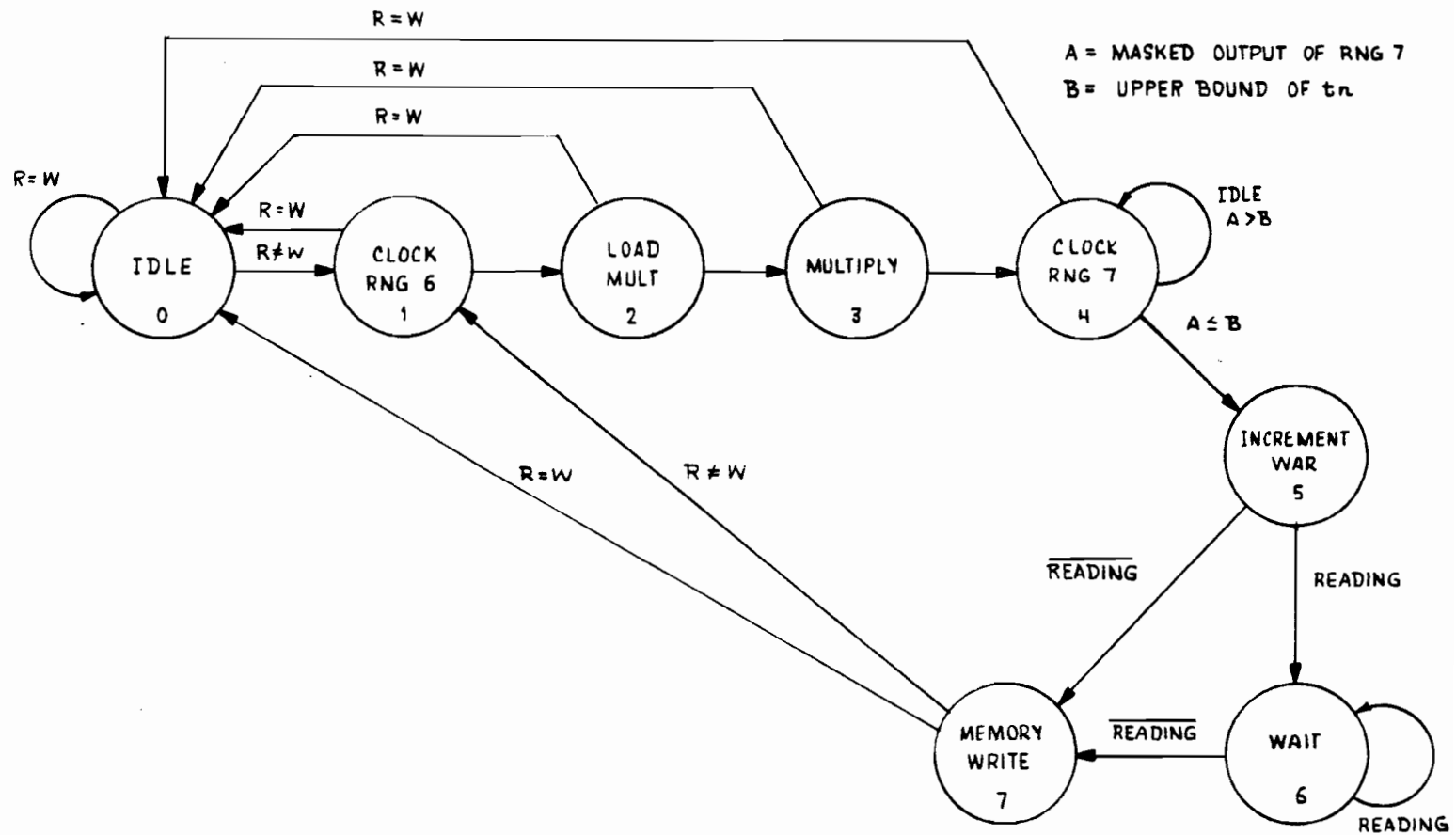


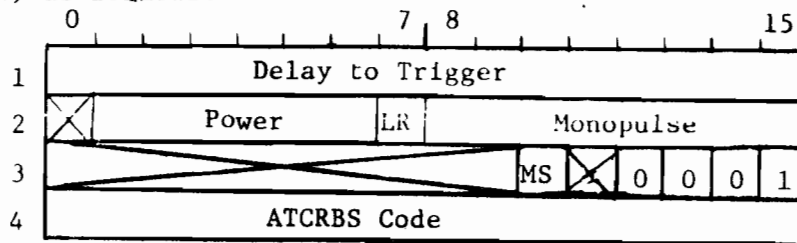
Fig. 2.4-36. Poisson sequence generator state diagram.

address is ever equal to the write address, the Poisson Sequence Controller will return to state 0. State 6 is a waiting state, which is entered when both FAT-Controller and Poisson Sequence Controller try to access the memory. Reading by the FAT-Controller always has higher priority, therefore state 6 will remain until reading is over. At that time, a new delay to trigger time will be written into the memory (State 7).

Reading of the memory by the FAT-Controller is discussed below.

#### 2.4.2.3.7 Reading the RPG

All six fruit control parameter generators discussed in the previous sections are tied to a common bus, called the RPG bus as shown in Fig. 2.4-31. To read the control parameters the FAT-Controller issues a read pulse, enabling the tri-state output of each generator at an appropriate time, and allowing the parameters to be formatted in the output register (Fig. 2.4-34) as follows:



As each group of 16 bit words is output from the register, in four consecutive clock intervals, the FAT Controller receives them via the D-Bus.

After the completion of reading, all random number generators, except RNG7, will be clocked once, allowing the generators to produce the next random numbers. Also, after each reading, the read address register (RAR) in the Poisson Sequence generator will be incremented and pointed to the next memory location.

The read control logic and its timing diagram are illustrated in Fig. 2.4-37.

#### 2.4.2.4 Fruit Generator Interface

The Fruit Generator (FG) Interface:

- (a) Provides fruit parameters to the Random Process Generator (RPG) via the FAT Controller.
- (b) Performs diagnostic tests on the RPG.



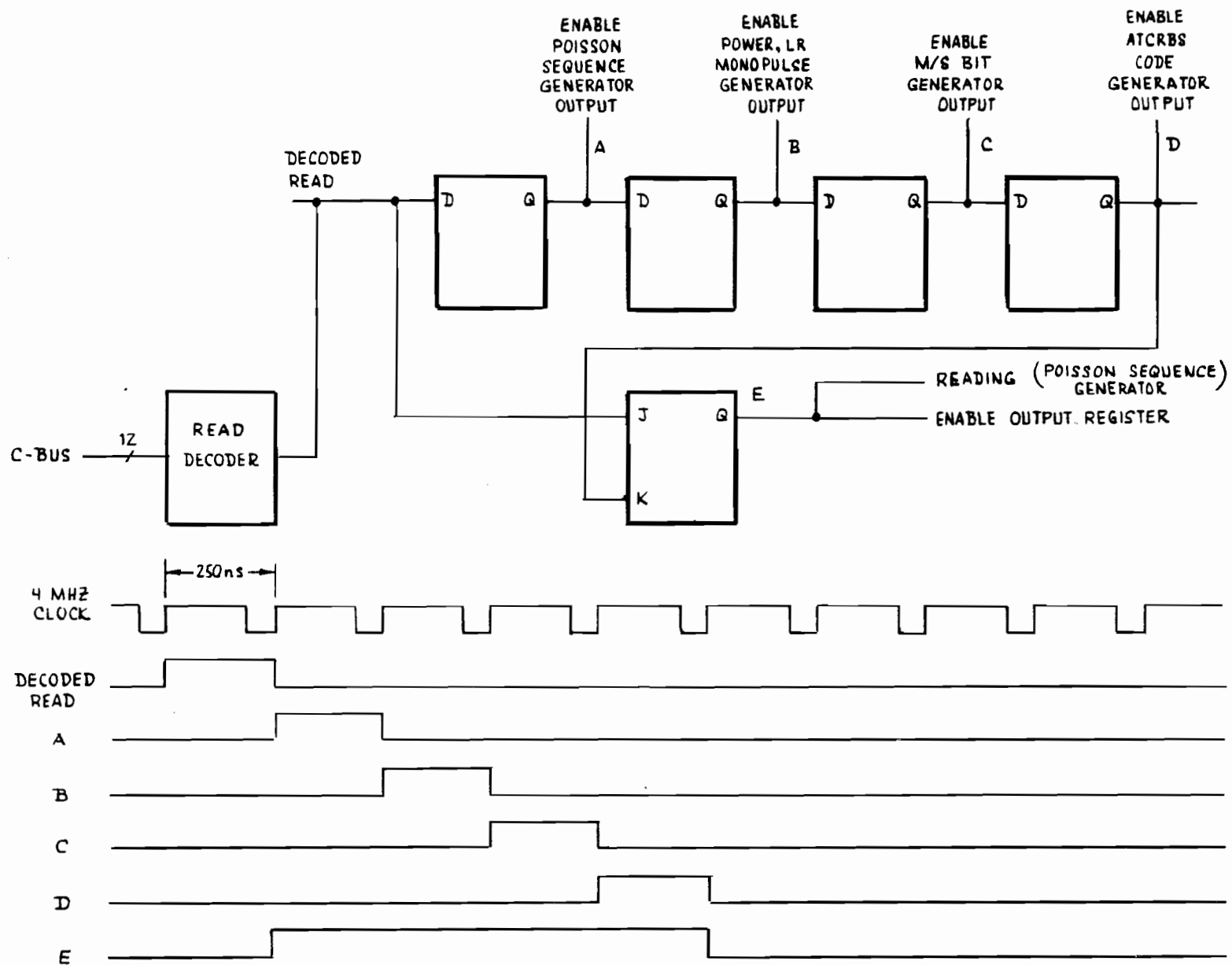


Fig. 2.4-37. Read control logic and timing diagram.

- (c) Performs diagnostic tests on the three FAT'S.
- (d) Provides a loop-test of the Interface itself.

The FG interface, Fig. 2.4-38, contains two nearly independent channels, the output channel for transferring data from the CPU to the controller, and the input channel for transfers in the opposite direction. Each channel consists of a sixteen word memory, a memory address register, and each uses one bit to indicate whether the memory is in a read or write mode. The memories used in the input channel and output channels are labeled as the  $\mu$ -P RAM and the CPU RAM respectively. Fig. 2.4-39 shows the circuit diagram for the output channel. Note that the address counter is wired so that it can only count from 4 to 7, resulting in only four out of sixteen possible locations in the memory being used. This is necessary since all transfers must be in groups of four words. To begin a transfer, the CPU must first check the status bit to be sure that it is allowed to write into the memory. A status bit value of '0' always means that the CPU has access to the memory. After four words have been written into the memory, the address counter wraps around and is left pointing at the first word stored (location 4), the status register is then toggled to the opposite state ('1'). This allows the controller to read four words from the memory. The status bits are connected to the D bus via an open-collector gate which is enabled when the status bits are examined by the controller. The status register will switch back to state '0' when the controller finishes reading. Note that the address counter and the status register are initialized such that the CPU can access the memory whenever an IORST or IOPLS is executed. A similar circuit is used for the input channel, except that the status register is initialized so that the controller has access when the interface is initialized by the IORST or IOPLS pulse.

The  $\mu$ -P RAM used in the input channel can also take data from the CPU RAM, as its input is multiplexed. This provides a means of loop-testing the interface, as the data sent to the output channel by the CPU can be verified by reading the data back from the input channel.

#### 2.4.2.4.1 Mode Decoding Logic

As mentioned previously, the Interface operates in four modes. These modes are programmable by setting the mode register via the DOC instruction as follows:

<u>Mode Register Bit</u>	<u>Mode of Operation</u>
0 0	Interface loop-test
0 1	FAT diagnostic
1 0	RPG diagnostic
1 1	Normal

Fig. 2.4-38. Fruit generator interface.

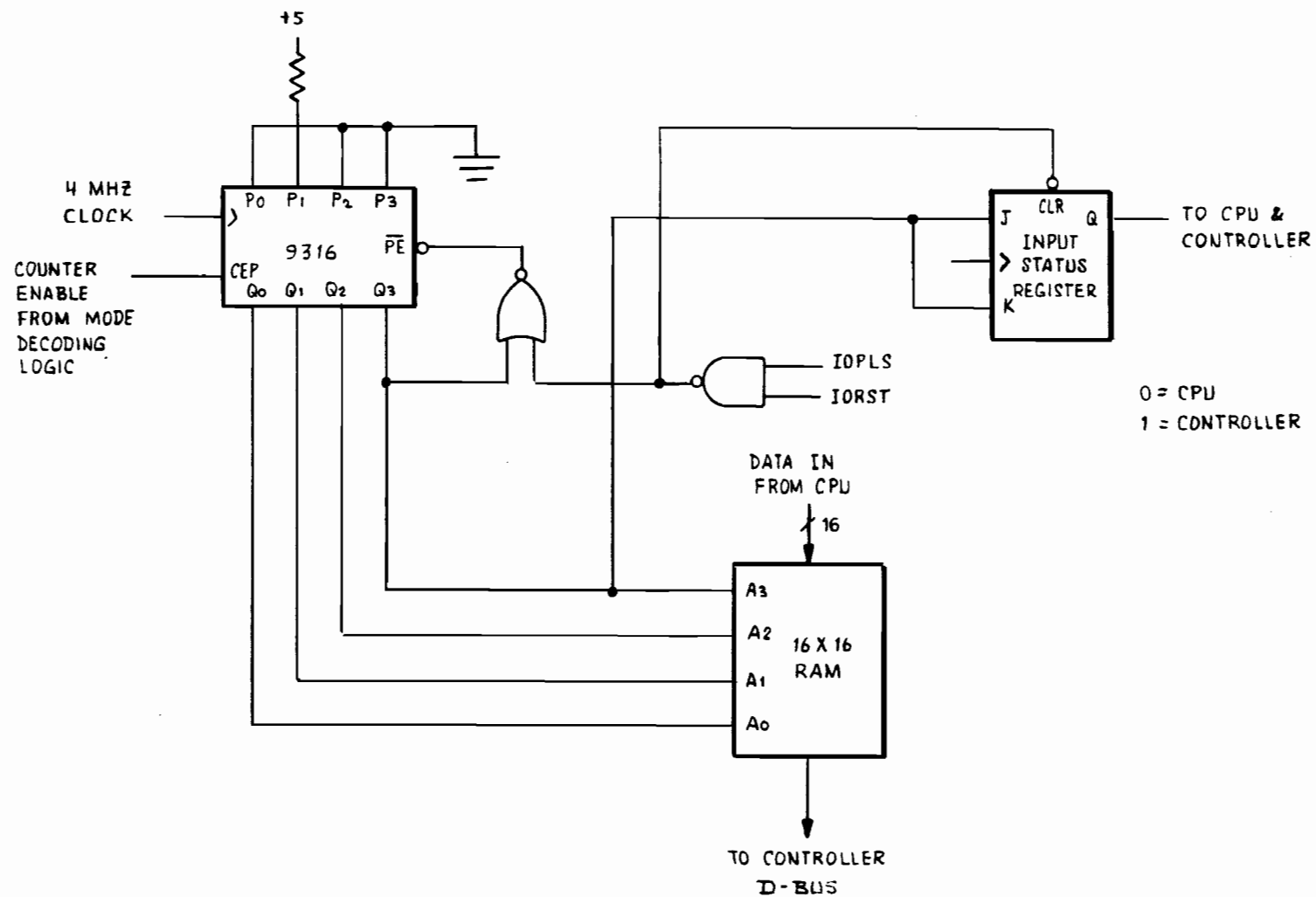


Fig. 2.4-39. Output channel.

To provide the proper address counter enable and memory write pulses for each mode of operation, decoding logic is required. Fig. 2.4-40 shows the mode decoding logic used in the FG Interface. This circuit, depending on the mode bits, combines the I/O pulses from the CPU and the Controller to form the appropriate enable and write pulses. The mode bits are examined by the controller in a similar fashion as the status bit.

#### 2.4.2.4.1.1 Normal Mode

The Normal Mode, the usual operation mode of the Interface, uses only the output channel and allows the CPU to output the following fruit parameters to the RPG via the controller (see Vol. 3 of this document for data format):

- (a) Average fruit rate
- (b) The fraction of replies in the main antenna beam (as opposed to those in the sidelobes).
- (c) The fraction of replies that will have a fixed known value for their data bits (as opposed to having randomly generated data bits).
- (d) Fixed data bits.

The CPU outputs these parameters to the CPU RAM by means of four DOA's. Each DOA is both rising and falling edge detected, creating two pulses DOA $\phi$ R and DOA $\phi$ F respectively. The reason for edge detecting the I/O pulse (DOA, DIA etc.) from the CPU is twofold.

1. It is necessary to synchronize the I/O pulses from the CPU to the system 4 MHz clock.
2. Edge detected pulses are one clock-period wide (250 ns), so that the address counter will increment only once for each DOA (or DIA).

As shown in Fig. 2.4-40, DOA $\phi$ R is used to write the data into the memory and DOA $\phi$ F is used to increment the memory address counter after each write.

To read the data from the CPU RAM, the controller issues a read pulse, causing the output Q<sub>3</sub> of the read counter (Fig. 2.4-41) to stay high for four clock cycles. This in turn causes the address counter to increment after each of the four read operations.

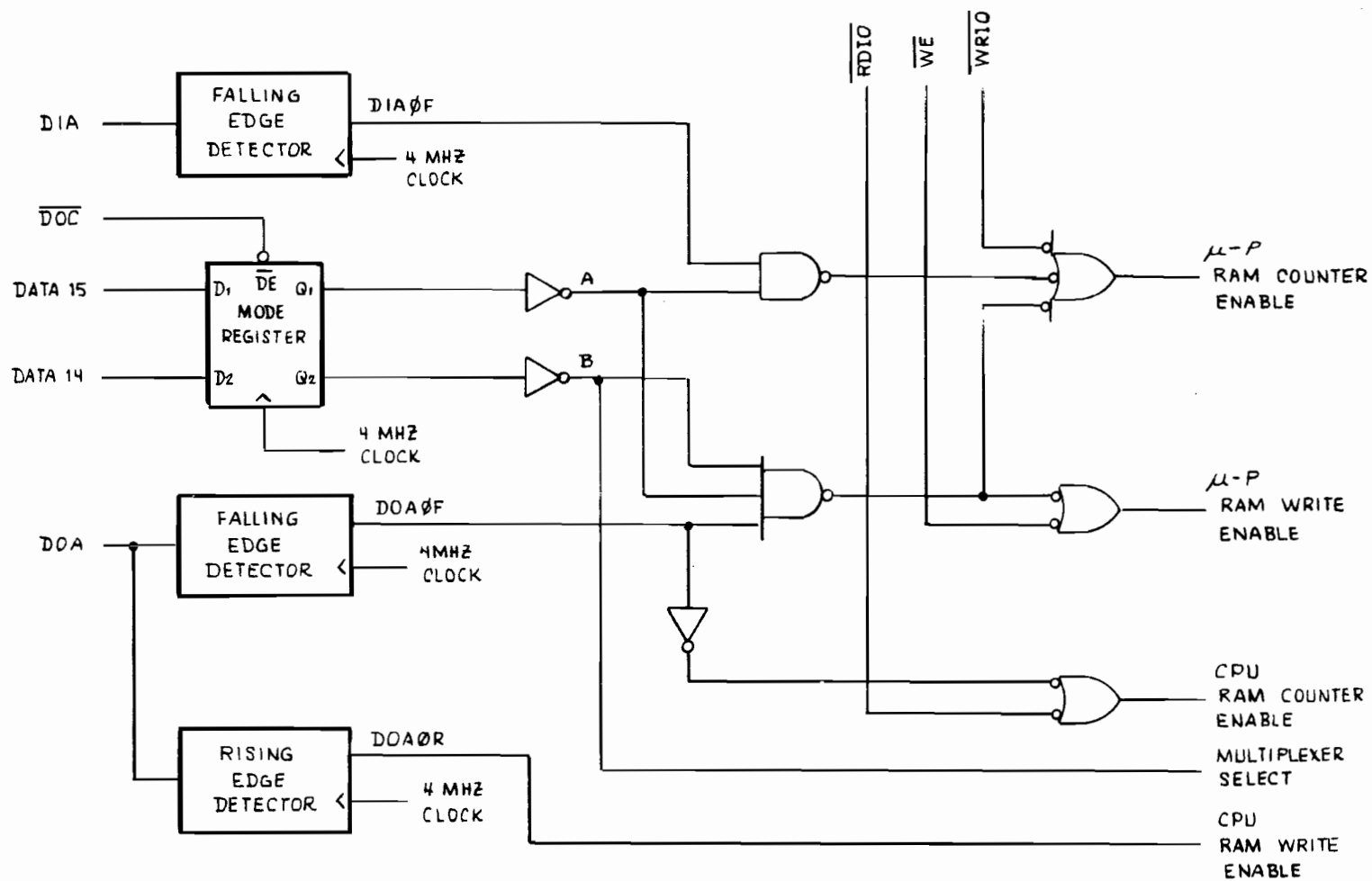


Fig. 2.4-40. Mode decoding logic.

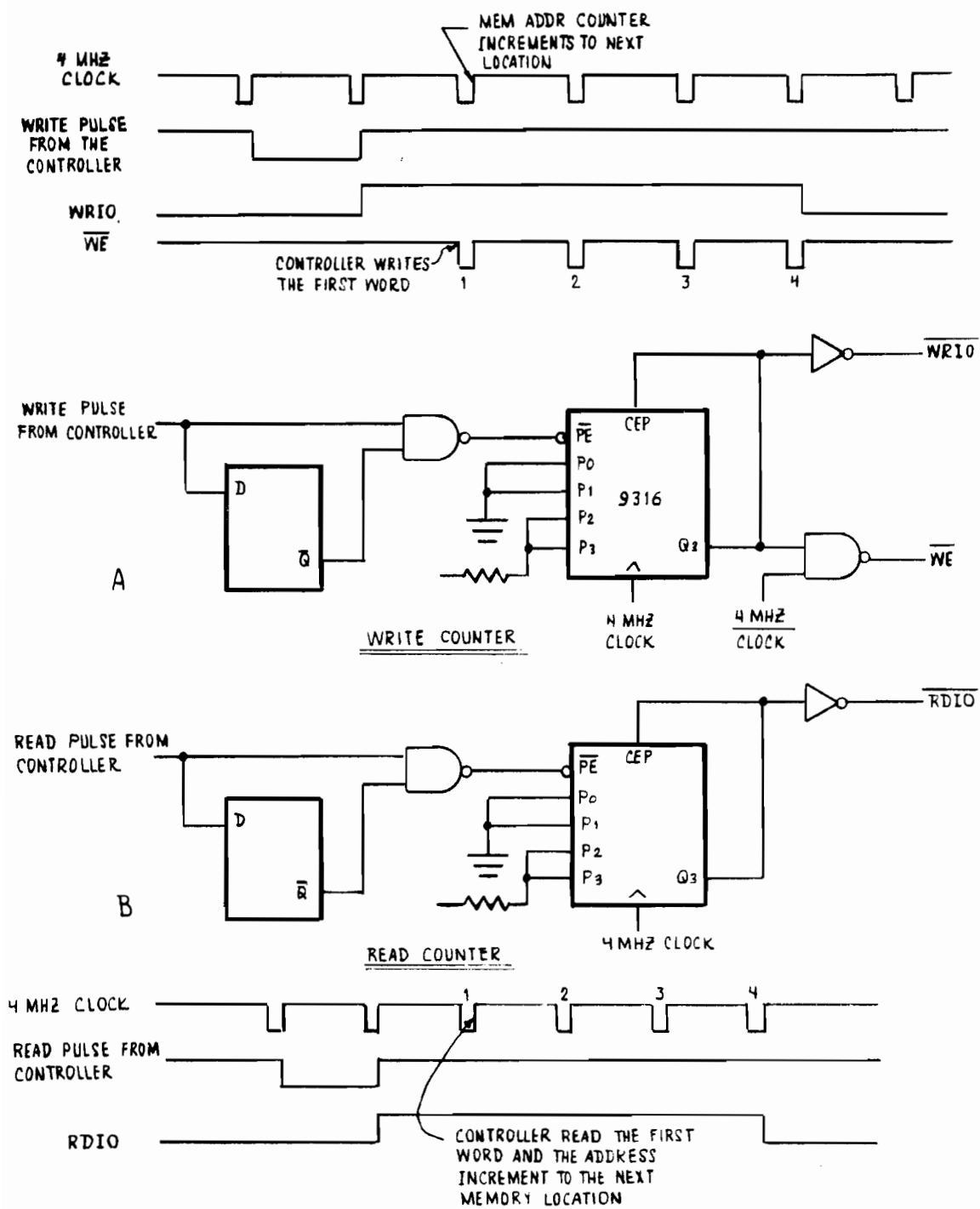


Fig. 2.4-41. Read/write counters.

#### 2.4.2.4.1.2 RPG Diagnostic Mode

The RPG diagnostic mode provides a way of checking the RPG operation from the CPU. Both the output channel and the input channel are used. To begin the operation, the CPU initializes the RPG by sending four fruit parameters to it using the output channel, exactly the same as in the Normal Mode. The controller then requests a fruit reply from the RPG and sends it to the CPU using the input channel. This operation continues until enough data is gathered to check the RPG operation. To write four words to the  $\mu$ -P RAM, the controller issues a write pulse, causing the write counter (Fig. 2.4-41) to generate four memory write pulses  $\overline{WE}$  and one four clock-cycle wide  $\overline{WRIO}$  pulse to the  $\mu$ -P RAM and its address counter respectively.

To read the  $\mu$ -P RAM, the CPU issues four DIA's to the interface. As in the case of DOA's each DIA is edge detected, forming the  $\overline{DIA\phi F}$  pulse. This pulse is forwarded to increment the address counter.

#### 2.4.2.4.1.3 FAT Diagnostic Mode

Normally the FATs receive fruit reply information from the RPG via the controller, and generate the appropriate analog reply signal at 60 MHz. To check the operation of the FAT's one might consider looking at fruit replies with the ARIES Self Test Unit. However, there are problems with this in that (a) the reply parameters are random, making it difficult to determine if the results are correct and (b) even at low fruit rates, replies will occasionally overlap and garble the results of the test. To overcome these problems, the FAT diagnostic mode is provided. This mode allows the CPU to insert known reply data into the FAT's, bypassing the RPG. Only the output channel is used. The method of transferring data is identical to the Normal mode. The replies generated can then be observed using the Self Test Unit.

#### 2.4.2.4.1.4 Loop Test Mode

Lastly, the loop-test modes provide a means of loop-testing the interface since the data sent to the output channel by the CPU can be verified by reading it back from the input channel. The controller does not participate in this mode so that the input and output status bit can be ignored. The 2-input multiplexer is caused to select data as coming from the CPU RAM to the  $\mu$ -P RAM. The writing and reading operations are similar to the other modes, involving four DOA's and four DIA's respectively, except that each DOA also causes data from the CPU RAM to be written into the  $\mu$ -P RAM.



## 2.5 IF Circuits

### 2.5.1 Controlled Reply and Fruit IF Circuits

Each ARIES Controlled Reply Generator and each Fruit Generator contains a subcircuit referred to as an IF unit. Views of the two subchassis making up each subcircuit are shown in Figs. 2.5-1 and 2.5-2; the identical circuit is used in each CAT and in each FAT.

Fig. 2.5-3A is a block diagram of the IF unit. Each such unit contains a crystal-controlled 60 MHz oscillator. Output from the oscillator is modulated by two Watkins Johnson pin diode switches incorporated in a strip line circuit to obtain on and off ratios in excess of 80 dB. The output from the modulator is attenuated in a digitally controlled diode attenuator to control the target power level. The attenuator has a 64 dB range in 1 dB steps. The attenuated signal is divided into the main and omni simulated antenna channels.

The main antenna channel goes to a diode switch that allows the option of simulating mainlobe or sidelobe signal levels at the sensor input. The sidelobe level can be adjusted manually with the sidelobe gain attenuator over a 15 dB range from the preset level of -35 dBm below the mainbeam. The resulting signal is then power divided to form the sum and delta channels going to the combiner. The off-boresight angle simulations are done by changing the gain setting of the delta ( $\Delta$ ) channel with respect to the sum ( $\Sigma$ ) channel. This is accomplished by means of an 8-bit digitally controlled attenuator. Each bit represents 0.125 dB attenuation of the delta channel with respect to the sum channel, and 0 corresponds to  $\Delta = \Sigma$ . The sign of the off-boresight angle is simulated by digitally controlling a  $180^\circ$  phase shifter (LR = 0 and LR = 1 correspond to left and right of boresight, respectively). A fixed attenuator is installed in the sum channel to compensate for losses contributed by the phase shift network in the delta channels, allowing balancing of the sum and delta channels. (See "DABS Monopulse Summary", D. Karp and M.L. Wood, Report No. FAA-RD-76-219, for more detail on DABS Monopulse Operation).

### 2.5.2 IF Combiner

The IF combiner, Figs. 2.5-3-B, 2.5-4, and 2.5-5, combines the  $\Sigma$ ,  $\Delta$ , and  $\Omega$  IF inputs from the three reply generators and the three fruit generators in a set of 6 to 1 power combiners to provide a single set of IF inputs for the DABS sensor. The three combined signals are then attenuated in a set of manually controlled attenuators. These attenuators are used to establish the relative mainbeam to omni levels for the particular DABS antenna which is being used. They may also be used to compensate for cable losses between ARIES and the DABS sensor. The nominal attenuator settings are 8 dB and allow 15 dB variation in 1 dB steps.

-4-18932

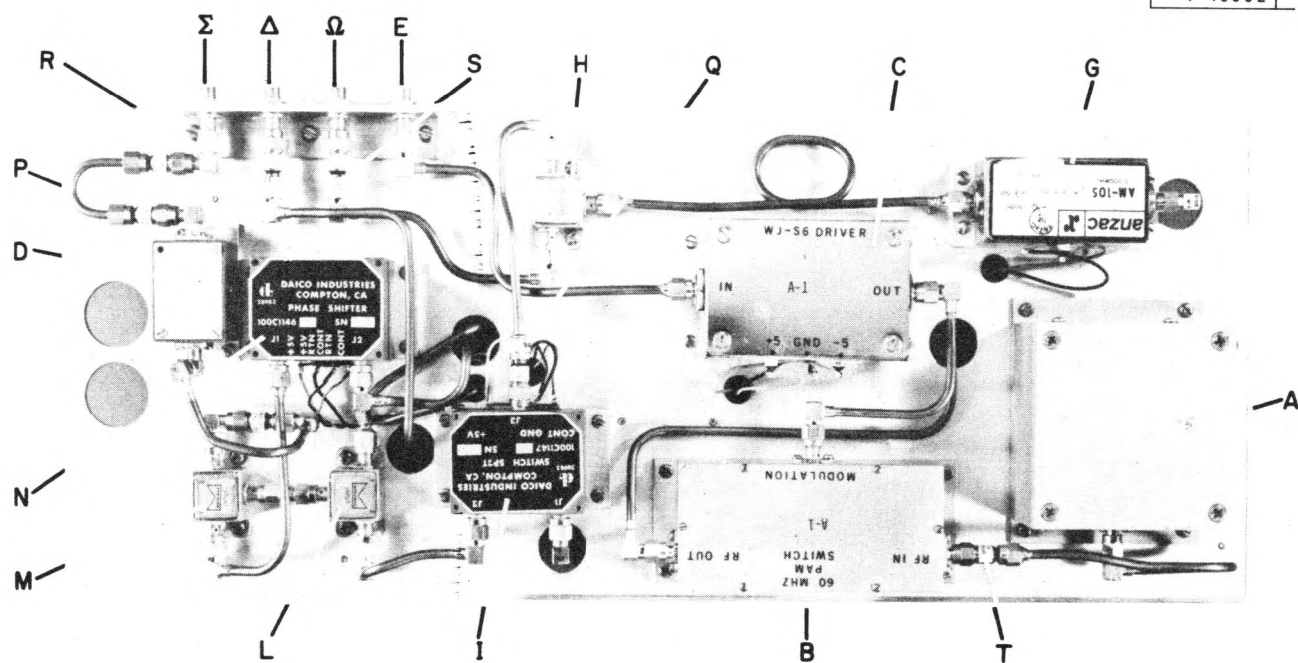


Fig. 2.5-1. Reply generator (side no. 1).

TARGET SIMULATOR (SIDE NO. 1  
(Refer to ARIES Target Simulator Schematic)

A	60 MHz Oscillator
B	PAM Switch
C	Switch Driver
D	Delay Line
E	PAM Data In (From Digital Target Generator)
G	19 dB Amplifier (Anzac)
H	Power Combiner
I	SPDT Switch
L	Power Combiner
M	Power Combiner
N	180 Degree Phase Shifter
P	Attenuators
Q	Fixed Attenuator
R	Fixed Attenuator
S	Fixed Attenuator
T	Fixed Attenuator
$\Sigma$	$\Sigma$ Out to IF Combiner
$\Delta$	$\Delta$ Out to IF Combiner
$\Omega$	$\Omega$ Out to IF Combiner

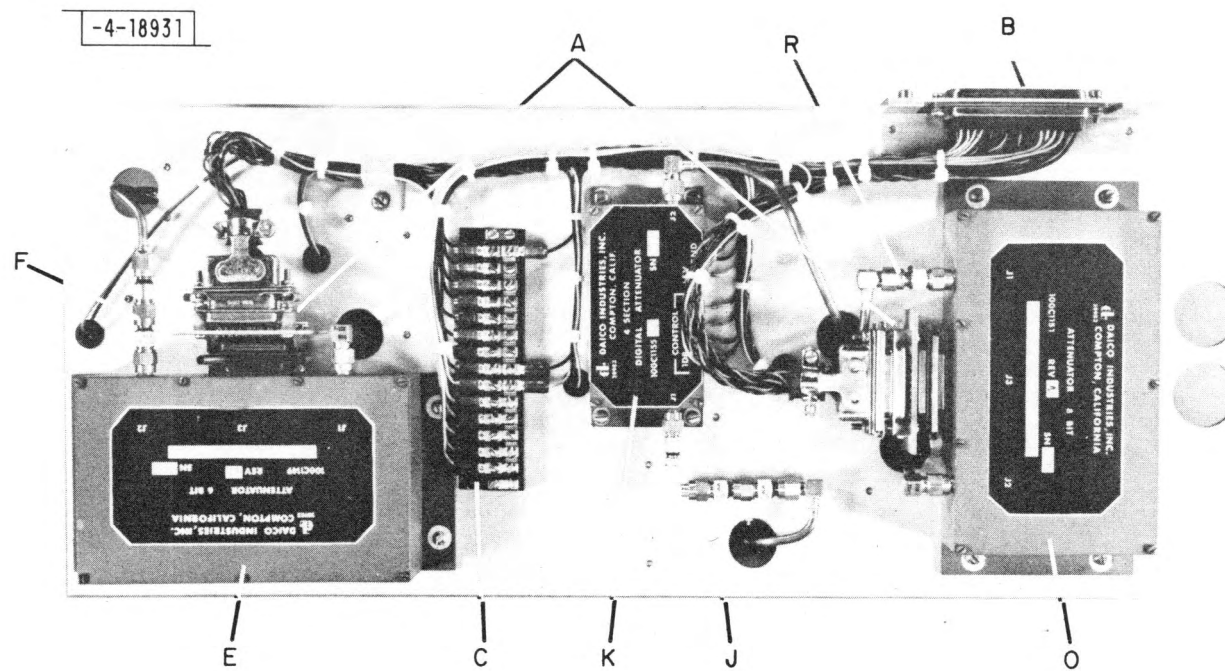


Fig. 2.5-2. Reply generator (side no. 2).

ANALOG REPLY GENERATOR (SIDE NO. 2)

(Refer to Target Simulator Schematic)

- A Termination (Resistor Assembly)
- B Power, Monopulse, Sidelobe, Attenuator Control Bits
- C Terminal Block
- E 6-Bit Digital Attenuator
- F Fixed Attenuator
- J Fixed Attenuator
- K 4-Bit Digital Attenuator
- O 8-Bit Digital Attenuator
- R Fixed Attenuator

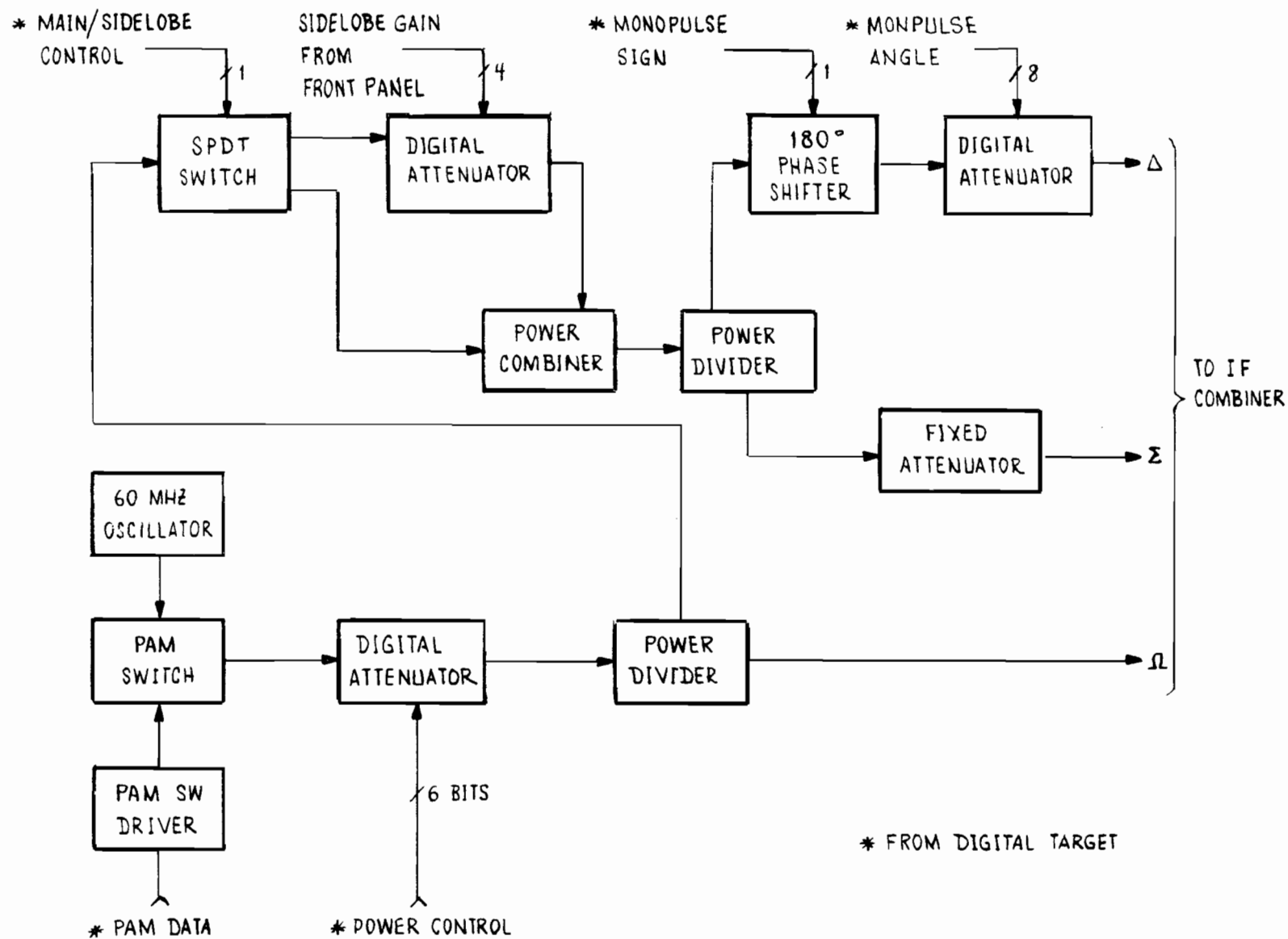


Fig. 2.5-3A. IF unit, block diagram.

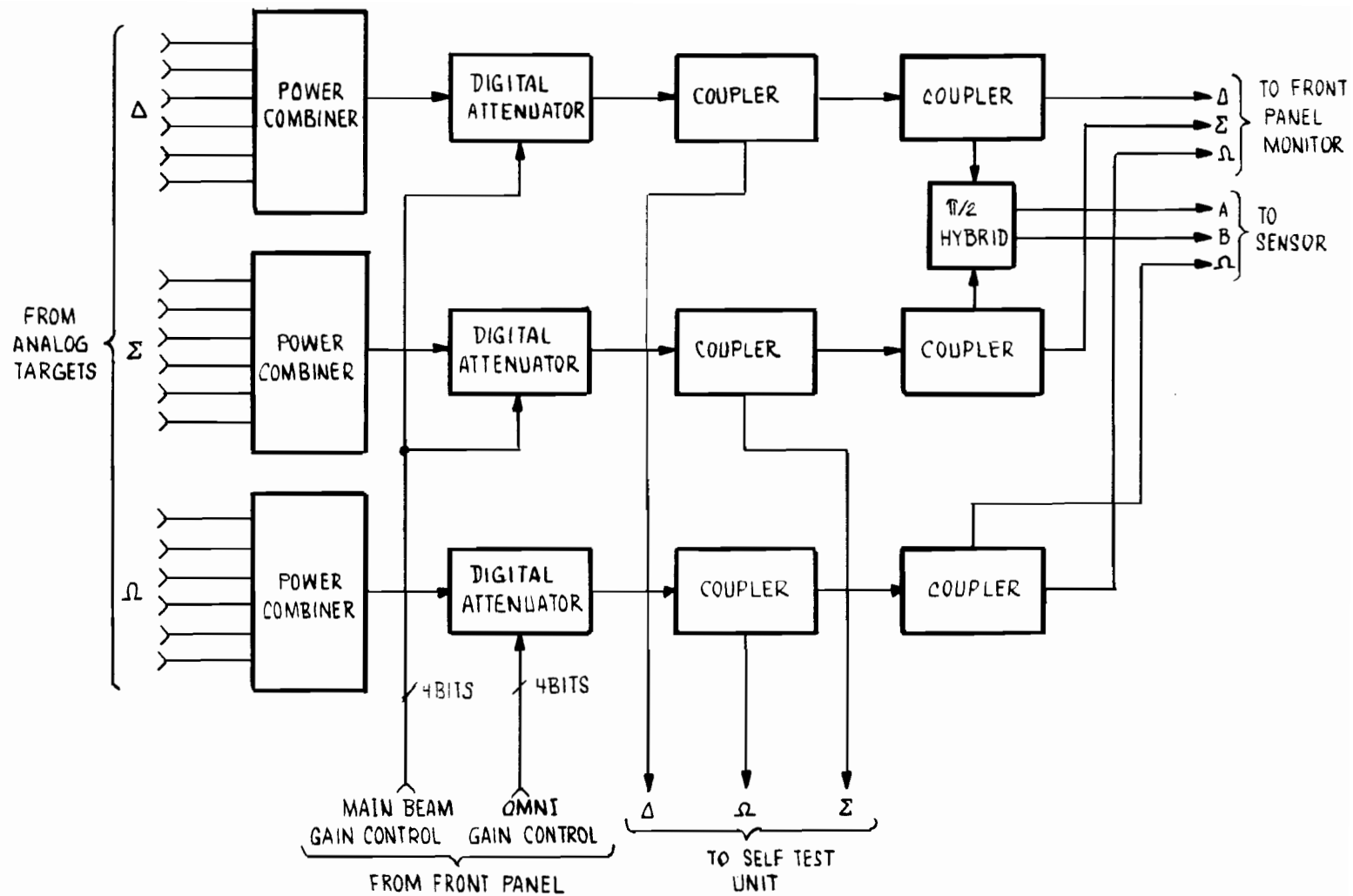


Fig. 2.5-3B. IF unit, block diagram.

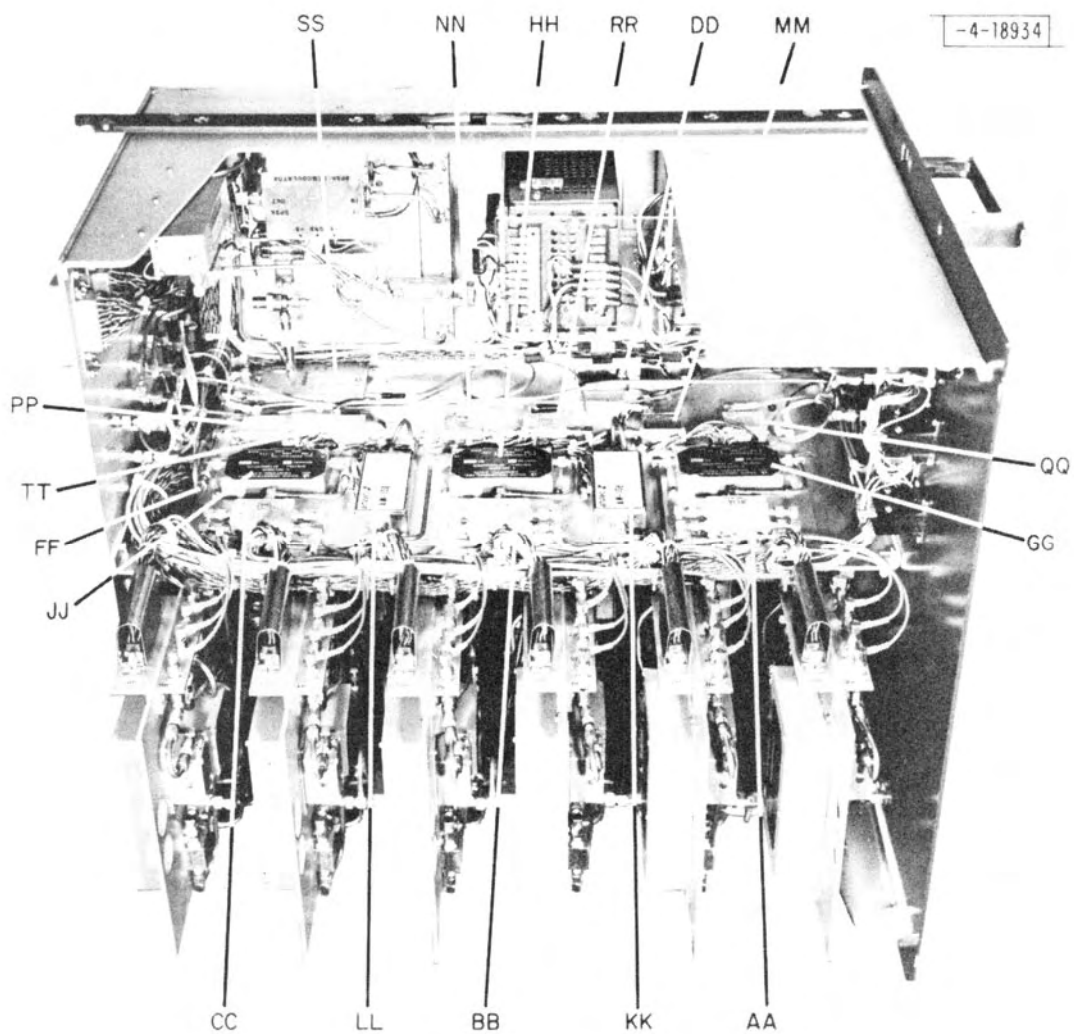


Fig. 2.5-4. IF combiner.



ANALOG DRAWER (TOP VIEW)

(Refer to Target Simulator Schematic)

AA	Power Combiner
BB	Power Combiner
CC	Power Combiner
DD	J103 (See IF Power Combiner Schematic)
FF	Fixed Attenuator
GG	4-Bit Digital Attenuator
HH	4-Bit Digital Attenuator
JJ	4-Bit Digital Attenuator
KK	28.6 dB Attenuator
LL	28.6 dB Amplifier
MM	Coupler
NN	Coupler
PP	Coupler
QQ	Coupler
RR	Coupler
SS	$\pi/2$ Hybrid
TT	Coupler

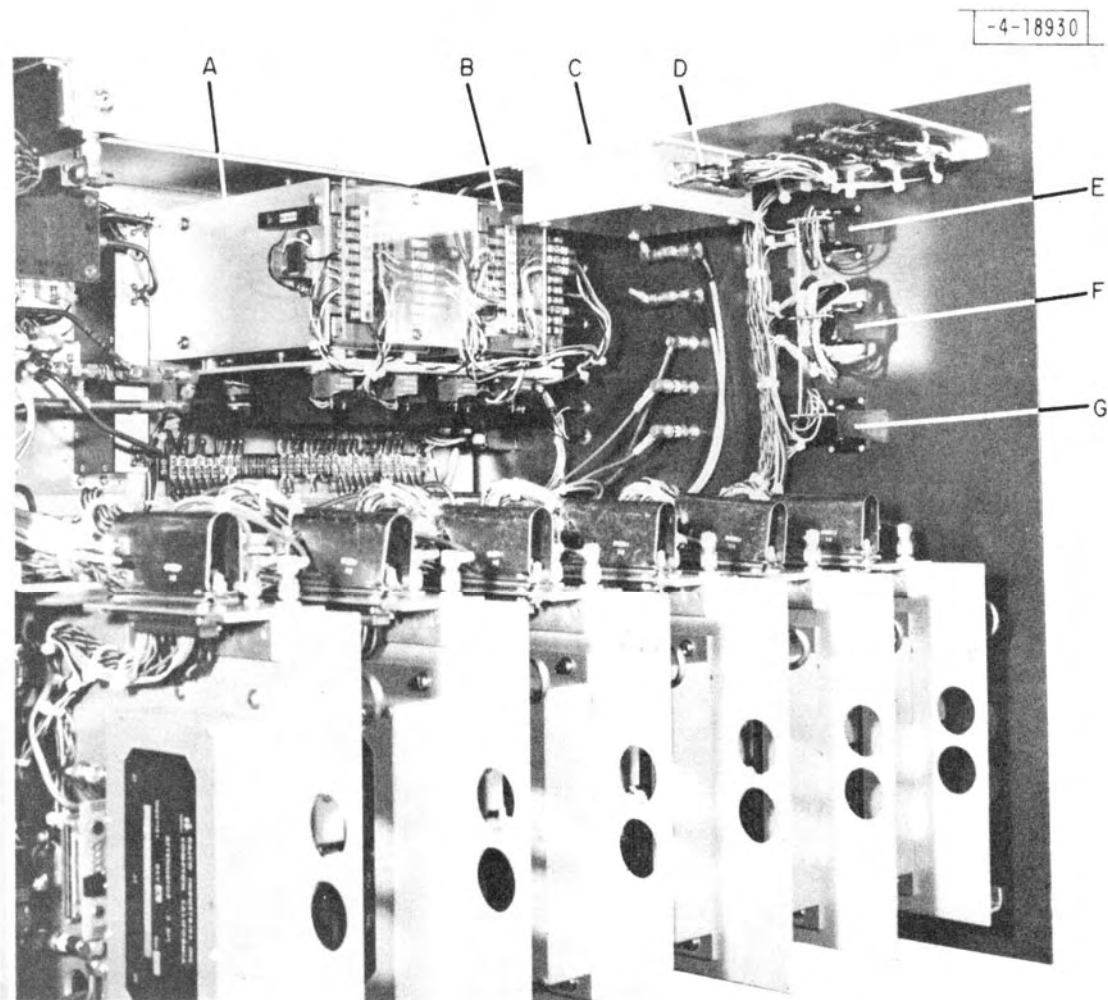


Fig. 2.5-5. IF combiner (left view).

ANALOG DRAWER (LEFT VIEW)

(Refer to Digital Attenuator Control Schematic)

- A 24 VDC Supply (Lambda LCS-A-24)
- B 20 VDC Supply (Lambda LCS-A-20)
- C Control Box
- D J118
- E Omni Gain Control
- F Sidelobe Gain Control
- G Mainbeam Gain Control

After mainbeam attenuation, signals for the ARIES tester and front panel monitors are coupled from each channel by 10 dB couplers. Then the sum and delta signals are combined in a quadrature hybrid to generate the  $A_2$  and  $B_2$  signals proportional to  $\Sigma + j\Delta$  and  $\Sigma - j\Delta$ .

## 2.6 ACP Decoder

Antenna azimuth data is normally transmitted from the antenna to the DABS sensor in the form of Azimuth Change Pulses (ACP's) provided by a shaft encoder. 16384 ACP's occur for each revolution of the antenna and a single Azimuth Reference Pulse (ARP) indicates when the antenna rotates past a reference azimuth such as true north. ARIES is capable of testing a DABS sensor whether it is operating with an antenna or not. When the sensor is operating without an antenna, ARIES generates simulated ACP's and ARP's just as if they were coming from an antenna rotating at one revolution per 4 seconds (this rate can be adjusted manually). When DABS is coupled to an antenna, the antenna generated ACP's and ARP's are brought to the ACP Decoder Unit, so that it too will be aware of the true antenna azimuth and will be able to relate simulated replies and fruit to the actual antenna azimuth.

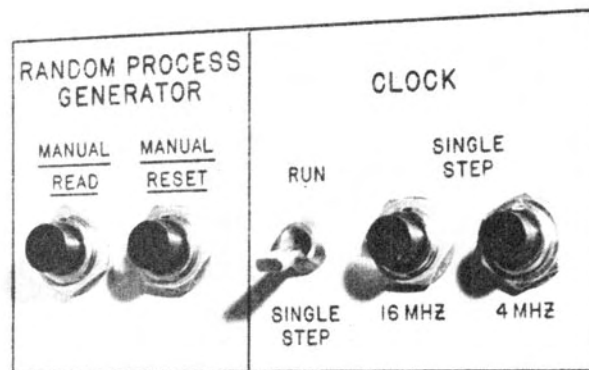
In both azimuth modes the ACP Decoder generates 14-bit parallel azimuth words in synchronism with the DABS ACP's/ARP's so that each interrogation received may be assigned a specific azimuth. These words are sent from the ACP Decoder to the Receiver.

### 2.6.1 Sensor Mode

When the front panel toggle switch marked AZ ARIES and AZ SENSOR (see Fig. 2.6-1 and Fig. 2.6-2) is in the AZ SENSOR position, ARIES azimuth words are slaved to the ACP's received from the DABS antenna. In this case Mux A and Mux B are set to pass the ACP's and ARP's received from the antenna to the DABS sensor. The serial ACP's are also passed to the input of the 14-bit serial-to-parallel converter, whose output is north corrected via addition of a manually programmable constant and transmitted to the receiver via 16-bit (2 bits unused) registers and drivers.

### 2.6.2 ARIES Mode

When the front panel toggle switch is in the AZ ARIES position Mux A and Mux B are in the opposite state, and ARIES originates serial ACP's and ARP's, sending them directly to the sensor, and again forwarding 16-bit, parallel, north corrected ACP's to the receiver.



CAT CONTROLLER  
MANUAL RESET



FAT CONTROLLER  
MANUAL RESET



ARIES



Fig. 2.6-1. Digital subsystem sub panel (front view).

Fig. 2.6-2. ACP decoder, block diagram.

### 2.6.2.1 Programmable ACP Counter and Correction Counter

These two counters operate in unison to count down the 4 MHz clock to an average ACP rate matching that which would be received from the DABS antenna were it connected and turning at one revolution per exactly 4.0 seconds, i.e., 16384 pulses in 4.0 seconds. Each such pulse would be separated by 244.1406250  $\mu$ sec. However, since the ARIES clock pulses are 0.250  $\mu$ sec apart, it is clear that by counting them down, ACP's cannot be generated which are precisely 244.1406250  $\mu$ sec apart. To generate 16384 ACP's per antenna revolution whose total period is 4.0 sec and whose average spacing is 244.140625  $\mu$ sec, the following artifice is employed: 9216 pulses are generated 244.2500  $\mu$ sec apart, the following 7168 pulses 244.000  $\mu$ sec apart, and the cycle repeated continuously. Thus the reference pulse spacing is:

$$\begin{aligned} &9216 (244.250 \times 10^{-6}) + 7168 (244.0 \times 10^{-6}) \\ &= 3.995392 \text{ seconds} \end{aligned}$$

To generate the pulses 244.0000  $\mu$ sec apart, preset counter PC1 is "programmed", (i.e., connected by the switches shown above it on Fig. 2.6-3) to count by

$$\frac{244.00}{.25} = 974 \text{ counts.}$$

To generate the pulses 244.2500  $\mu$ sec apart, preset counter PC-2 is connected to the COUNTER ENABLE input of PC-1 via Gate 1 and a D-Flop in such a way as to cause PC-1 to cycle-skip or miss one clock pulse every ACP. The action is as follows: when Gate 1 is enabled, the D-Flop remains reset (Q output low) for one clock period following each ACP, thus disabling PC1. The correction counter, PC-2, is switch programmed so as to count by 9216. Hence for 9216 ACP's the NAND gate (Gate 1) is enabled so that every ACP pulse during this portion of the cycle resets the D-Flop for one clock cycle, effectively inhibiting PC-1 for one clock interval of 250 nsec.

### 2.6.3 Output Circuits

ACP's from PC-1 are stretched from .25  $\mu$ sec to 1.5  $\mu$ sec in order to simulate the true ACP's, converted to a parallel 14-bit format, passed through a north correction adder capable of adding from 0 to  $2^{14} - 1$  ACP's ( $0^0$  to  $360^0$ ) and forwarded to the receiver by appropriate registers and drivers.

Full count pulses from the 14 bit serial-to-parallel converter are also stretched to 1.5  $\mu$ sec to form the ARP pulses forwarded to the sensor in the ARIES mode.

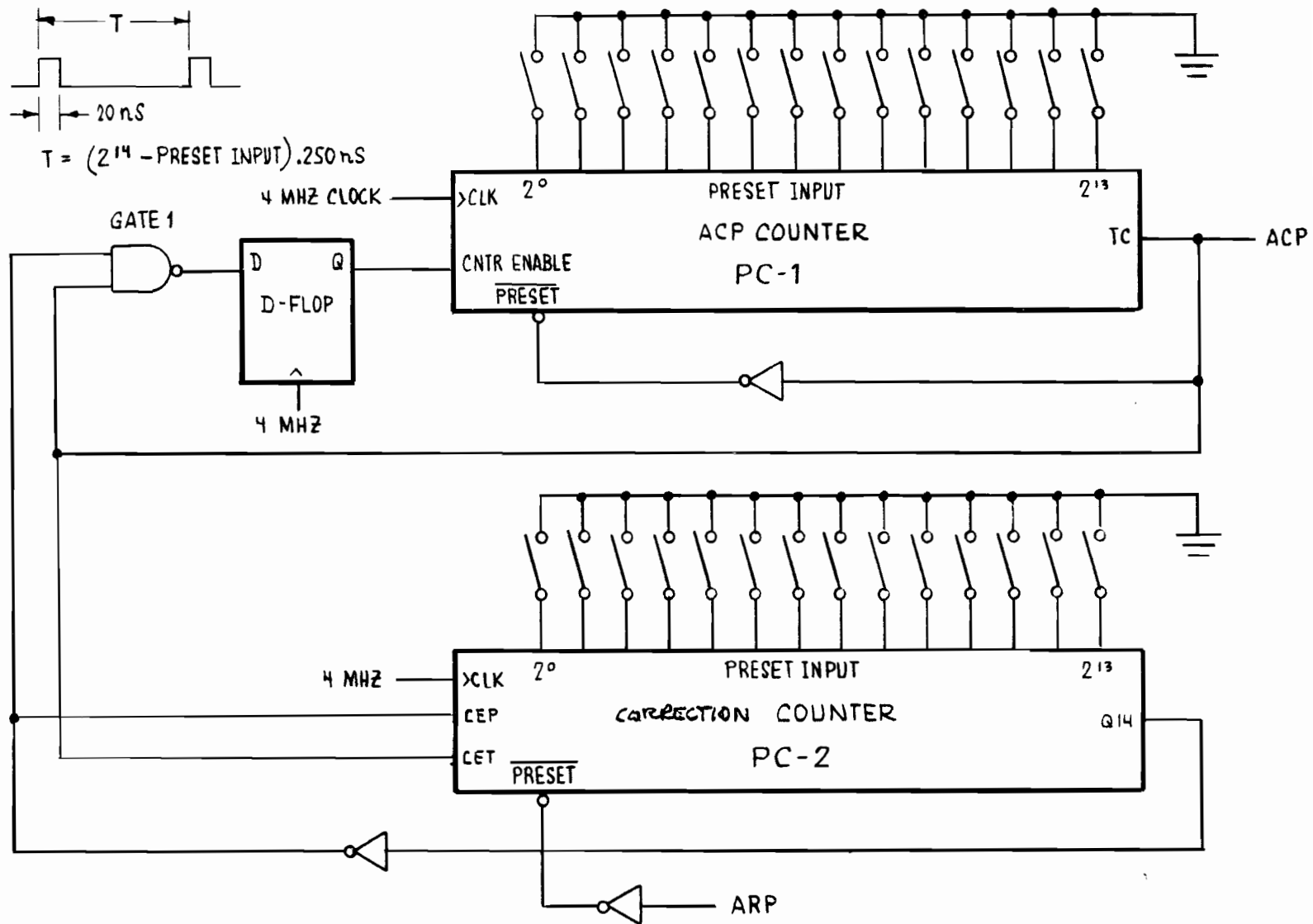


Fig. 2.6-3. ACP and correction counters, block diagram.



## 2.7 Status Formatter

The Status Formatter polls status and error condition lines from the UIT, Receiver, ACP Decoder, CAT's, FAT's, RPG, IF Combiner, front panel switches, and itself. When a status change or error condition occurs the status formatter interrupts the CPU. The software can then read the device status, if desired. Three polling modes are available as commanded from the ARIES CPU:

- (a) continuous polling until the status of any unit changes,
- (b) the CPU can request a status update at the end of the current polling cycle,
- (c) a self-diagnostic mode in which polling is inhibited and known status words are entered.

Under (a) and (b) polling must proceed to the end of the current polling cycle prior to any further action.

### 2.7.1 Operation of the Formatter

The Formatter, as shown in Fig. 2.7-1, consists of a 16-bit status word input register, a 16 word x 16-bit memory and associated memory address counter, a 16-bit comparator, an error register, a 3-bit roll-call counter and 3:8 decoder, and a controller which provides signals necessary to properly sequence the operations of these circuit elements for the mode of operation selected.

In the continuous polling mode the roll-call counter cycles continuously to cause the 3:8 decoder to generate 1-bit commutating signals fed to eight status report selection circuits, one located in each unit polled. These status formatter select signals (SF-SELX) enable the status/data read circuits of pairs of units (CAT1 with UIT, CAT2 with Receiver, CAT3 with ACP Decoder etc.) in sequence. See Fig. 2.7-2. Pairing is permitted since each 16-bit status word consists of two bytes, with bits 0-7 assigned to one unit of the pair and bits 8-15 assigned to its companion unit. (See Status Formatter word format definitions in Vol. 3 of this document.)

Referring to Fig. 2.7-2, it may be seen that a positive transition of the select signal (SF-SELX) for any pair of units will cause the D flip-flop to enable all of the data NAND gates, thereby transferring status/data from both units to the 16-bit input register of Fig. 2.7-1.

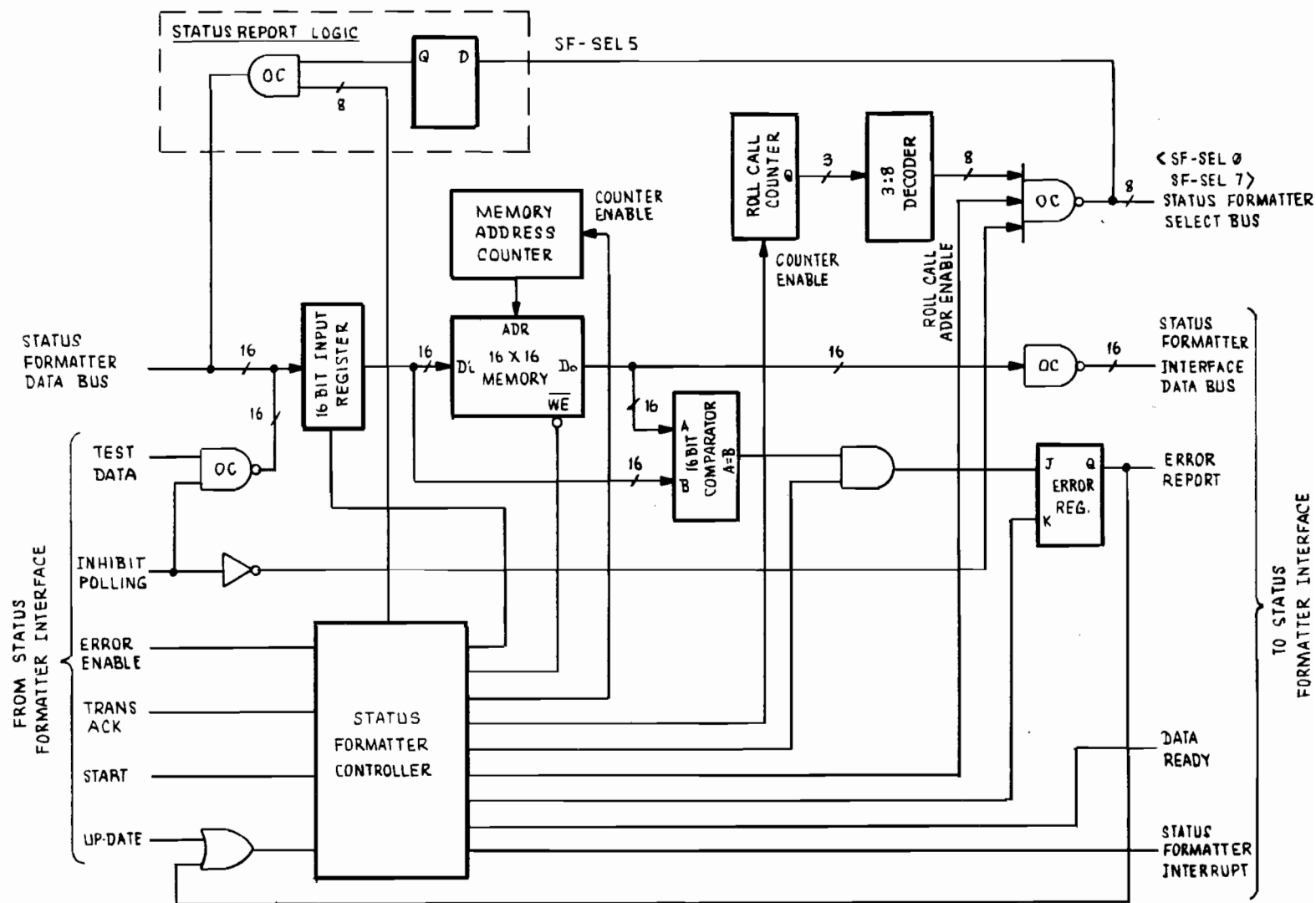


Fig. 2.7-1. Status formatter, block diagram.

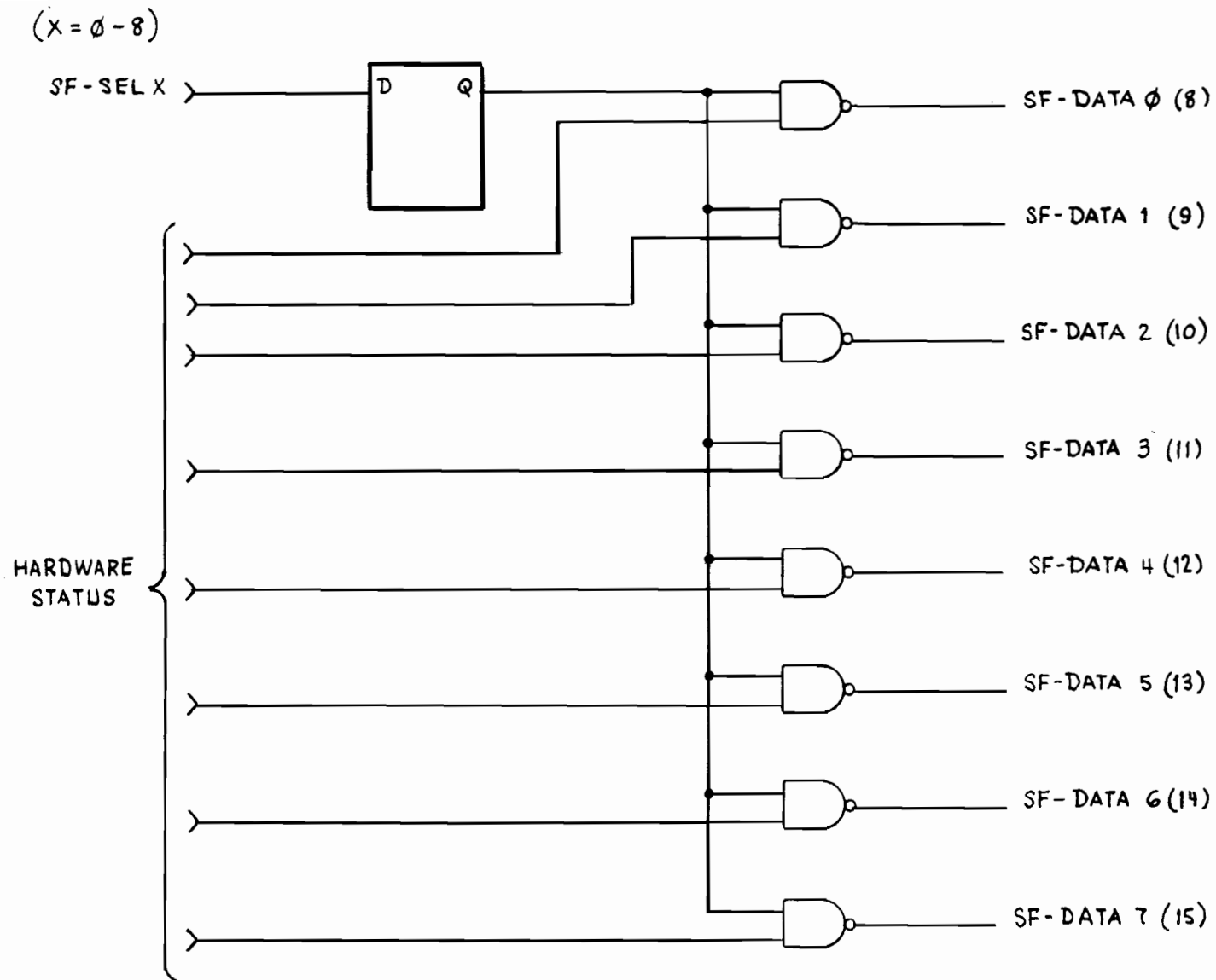


Fig. 2.7-2. Status report logic.

The essential action of the status-change checking circuitry is the bit-by-bit comparison (in the 16-bit comparator) of the newly-read data (resident in the 16-bit input register) with the corresponding data read during the last polling cycle and stored since that time in the memory at its assigned address. When no discrepancies are found, the error register is not set, when discrepancies are found the error register is set and the CPU informed of the status change by means of an interrupt.

#### 2.7.2 Status Formatter Polling Cycle

The status formatter controller is responsible for assuring that the states or operations shown in the diagram of Fig. 2.7-3 occur in proper sequence. The particular sequence or operational mode is determined by the the start, stop (inhibit), self-diagnose, and read commands received from the CPU. The steps of the diagram are described in Table 2.7-1.

#### 2.7.3 Formatter Self-Diagnostic Mode

Two control bits originating in the status formatter interface under operator command, INHIBIT POLLING and TEST DATA, invoke the formatter self-diagnostic mode and determine the test value (0's or 1's) to be input to the status formatter respectively. These control lines may be seen entering the left margin of Fig. 2.7-1.

If INHIBIT POLLING is set to 1, polling is prevented from taking place and the value of the TEST DATA bit is placed on all the input lines to the status formatter. This value will appear in all 128 bit positions of the status words if the formatter registers, memory and control logic are operating correctly.

#### 2.7.4 Status Formatter Interface

The Status Formatter (SF) Interface provides a means of controlling the SF modes just described, and serves to transfer hardware status from the SF to the CPU.

The interface consists of a C-Register, a Transfer Acknowledge Register, and drivers. The SF mode control bits (Update, Error Enable, and Inhibit Polling) and the Test Data bit are stored in the four-bit C-Register and can be set by means of a DOC instruction. For diagnostic purposes, the C-Register can be read by the CPU via a DIC instruction. Also read by the DIC instruction are two bits from the SF: Data-Ready (DR) and Error Report (ER) (see Table 2.7-2).

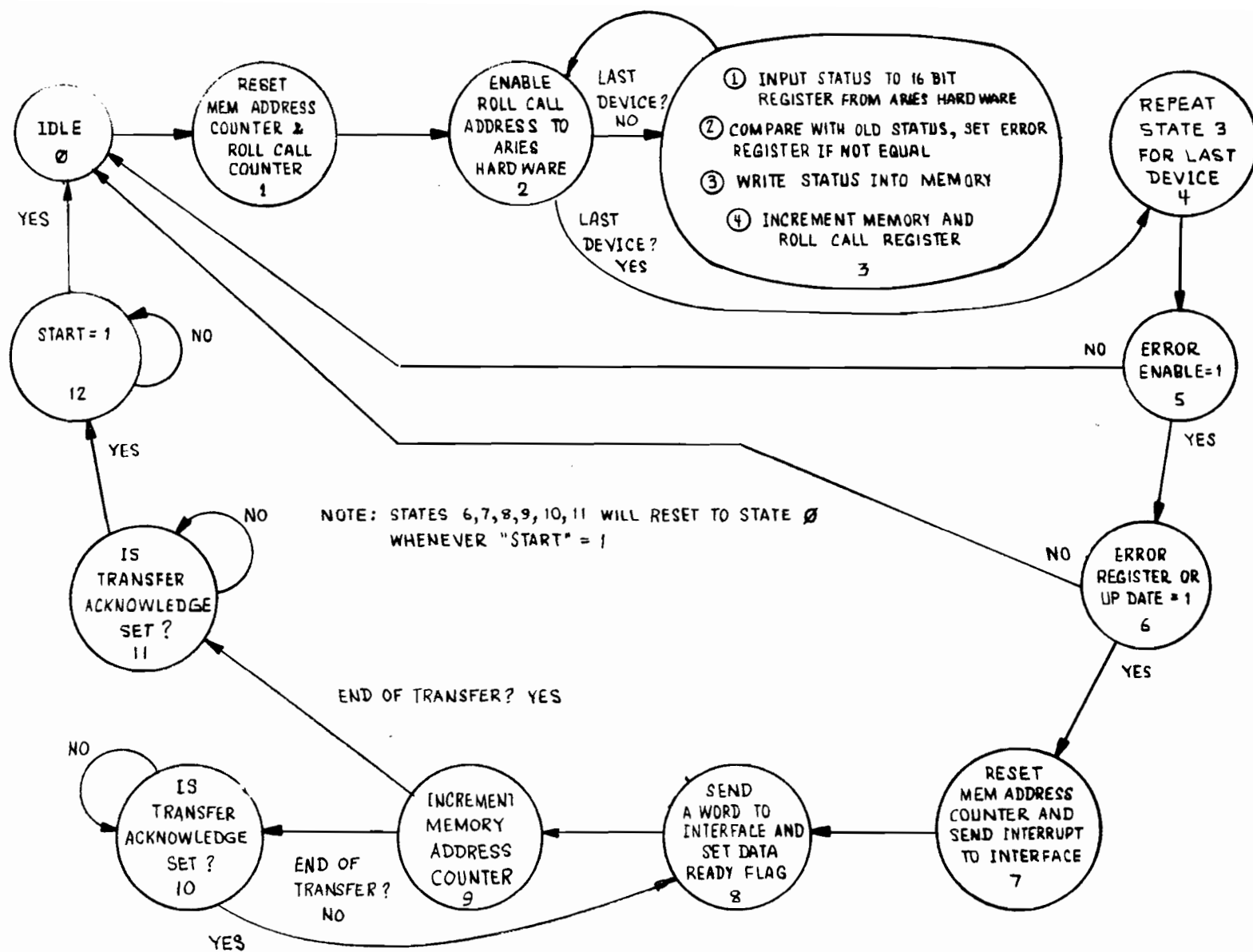


Fig. 2.7-3. Status formatter control sequence.

TABLE 2.7-1  
STATUS FORMATTER OPERATIONAL STEPS

- Step 0: An idle condition in which roll-call address generation circuits are enabled but polling is not taking place.
- Step 1: Memory address and roll-call counters are reset.
- Step 2: Roll-call address gates are enabled. Polling state checked- if it is not the last device, go to step 3; otherwise, go to step 4.
- Step 3: Status bits from the pair of units to be polled transferred to the 16-bit input register; status bits compared with old status bits - error register set if not equal, new status written into memory, and memory and roll-call counters advanced. Repeat step 2.
- Step 4: Repeat step 3 for the last device. When finished go to step 5.
- Step 5: Examine ERROR ENABLE bit (set by software) if not set, repeat entire polling cycle (go to step 1); if set, go to step 6.
- Step 6: Examine error register and UPDATE bit (set by software); if neither the register nor the UPDATE bit are set, go to step 1 and repeat cycle, if either is set proceed to step 7.
- Step 7: Reset memory address counter and send interrupt to the status formatter interface.
- Step 8: A data word is sent from the status formatter to the SF interface and the DATA READY flag set.
- Step 9: Increment the memory address counter, and if this is the last transfer, go to step 11; otherwise go to step 10.
- Step 10: Informed of the data (on the status formatter data bus) by the DATA READY flag, the interface, when ready, accepts it and indicates that the transfer has been completed by setting the TRANSFER ACKNOWLEDGEMENT line. Go to step 9.
- Step 11: Similar to State 10, but the TRANSFER ACKNOWLEDGE causes a transition to state 12.
- Step 12: The START line (S:) is checked. If the S-line is set, the memory address and roll-call counters are reset and the entire cycle is reactivated from step 0 on. If S is not set, action ceases until it is set by the programmer.

TABLE 2.7-2

BIT DEFINITIONS

- DR: Data Ready. This bit indicates that the status formatter has a word to transfer to the CPU. It will normally be 0 during polling, and 1 if polling stops due to a status change or an UPDATE request and all 8 words have not yet been read. It is intended to be used only for hardware diagnostic purposes. After each DIA instruction, DR = 0 until the interface obtains the next word from the status formatter. Normally this word will arrive too quickly for the CPU to notice that DR = 0, but in the event of hardware failure this may not be the case.
- ER: Error Report. This bit reads the state of the flip-flop which indicates if a status change has been detected during the current polling cycle (ER = 1). Again, this is useful primarily for hardware diagnostic purposes. However, in the case where polling has stopped due to an UPDATE request, this can be used to determine if a status change was also detected on the last polling cycle.

Upon receipt of an SF interrupt, indicating eight words of hardware status are available, the CPU can read the words by executing successive DIA's.

Since the CPU and the SF operate asynchronously, a means of hand-shaking must be provided to insure proper data transfer. This is the purpose of the Transfer Acknowledge Register. The register is normally in a reset state (Q = 0) as the reset input of the register is held low by the DATA READY bit. When there is a word available in the SF Interface data bus, the DATA READY bit will be set by the SF (Logical 0) causing the D and reset inputs of the register to go high. When the CPU has read the word, the DIA pulse will set the Trans-Ack register, which in turn causes the SF to output the next word.

It is important to realize that there is no requirement for the CPU to read all eight words or to read any. If none, or only the first few of the words are read by the CPU, then it can restart the Status Formatter polling sequence by executing either a START or IORST instruction. Fig. 2.7-4 shows a simplified diagram for the SF Interface.

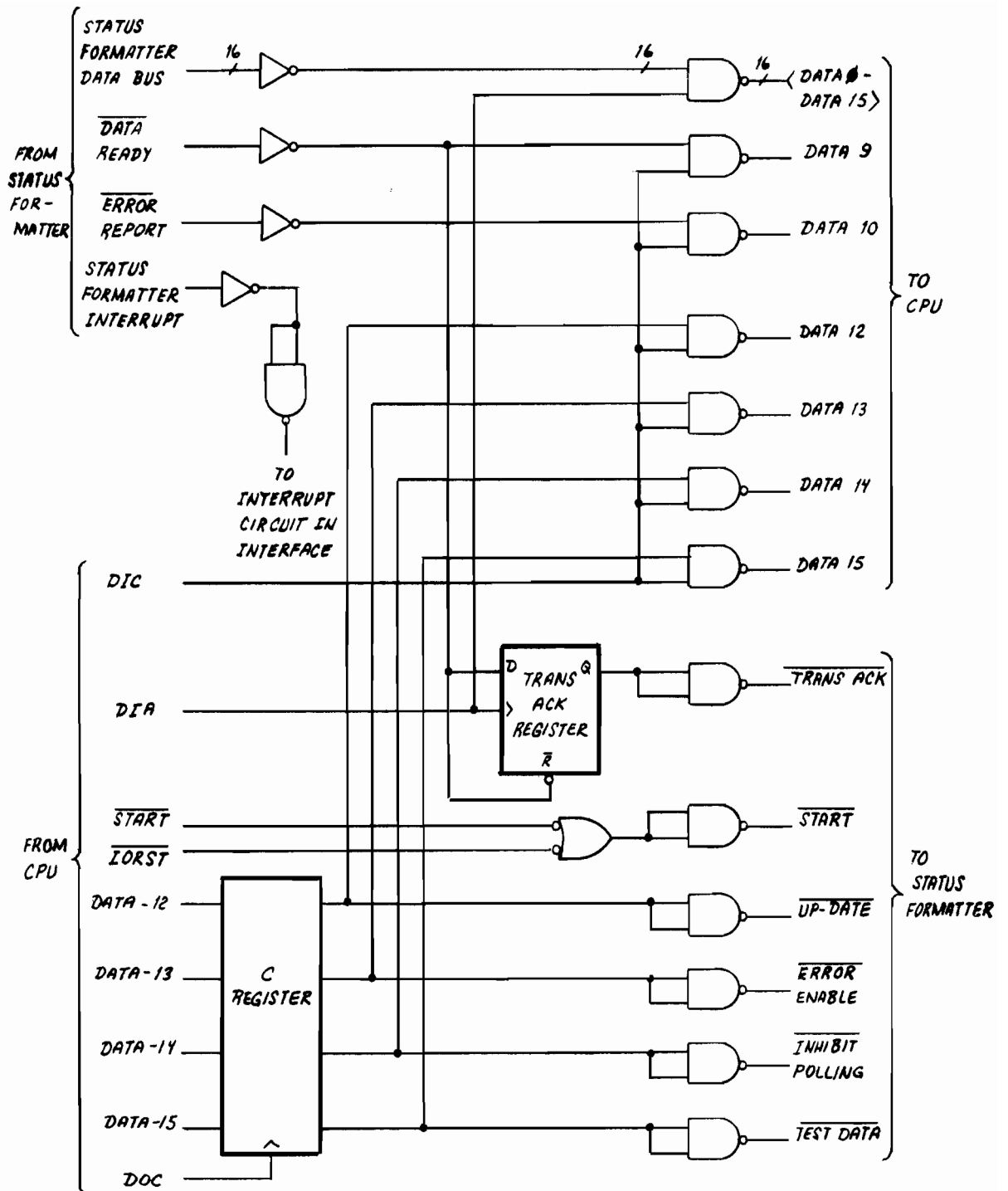


Fig. 2.7-4. Status formatter interface.



## 2.8 Self Test Unit

### 2.8.1 Modes of Operation

The purpose of the Self Test Unit (STU) is to provide for loop tests of the Receiver, Target Generators and the IF Combiner. During these loop tests the STU operates under CPU control, in one of two modes: the Interrogation Generation Mode or the Reply Sampling Mode.

In the Interrogation Mode the STU generates and forwards any one of ten different pre-programmed interrogations, or a DABS interrogation programmed from the front panel, to the receiver. The expected interrogation type and data (DABS only) are then compared to the data block from the receiver. A loop test of the receiver (both analog and digital portions) is thus provided.

In the Reply Sampling Mode, a known reply is generated by the Controlled Reply Generator (CRG) or by the Fruit Generator (FG). The corresponding sum ( $\Sigma$ ), difference ( $\Delta$ ), and omni ( $\Omega$ ) outputs from the IF Combiner are sampled by the STU for reply time accuracy, amplitude and phase correctness, and reply data bits. This provides a test of the reply controller, the reply controller interface, the digital target generator, the analog target generator, and the IF Combiner.

Figure 2.8-1 is a block diagram of the STU. The STU mode of operation is selected by the CPU via two registers: the control word register and the trigger time register. The control word register receives control words from the CPU which contain the following subfields: uplink select (UPSEL) - selects the type of pre-programmed interrogation to be sent; sample select (SSEL) - selects which IF signal ( $\Delta$ ,  $\Sigma$ ,  $\Omega$ ) the reply sampling circuit of the STU will sample; and reply mode select (RD) - selects which reply mode (DABS or ATCRBS) is to be expected by the STU.

The trigger time register contains a 16-bit interrogation time from the CPU. Its content is continuously compared with the contents of the 16-bit range counter. Each time the range counter value reaches this value, the selected interrogation is sent. The range counter is reset to zero when a TOA is received from the receiver, synchronizing the STU range counter with the range counters of the reply generators and the receiver.

### 2.8.2 Interrogation Generator

The Interrogation Generator consists of a digital and an RF section. Interrogation pulses generated in the digital section are modulated at 1030 MHz by the analog section and coupled into the receiver input.

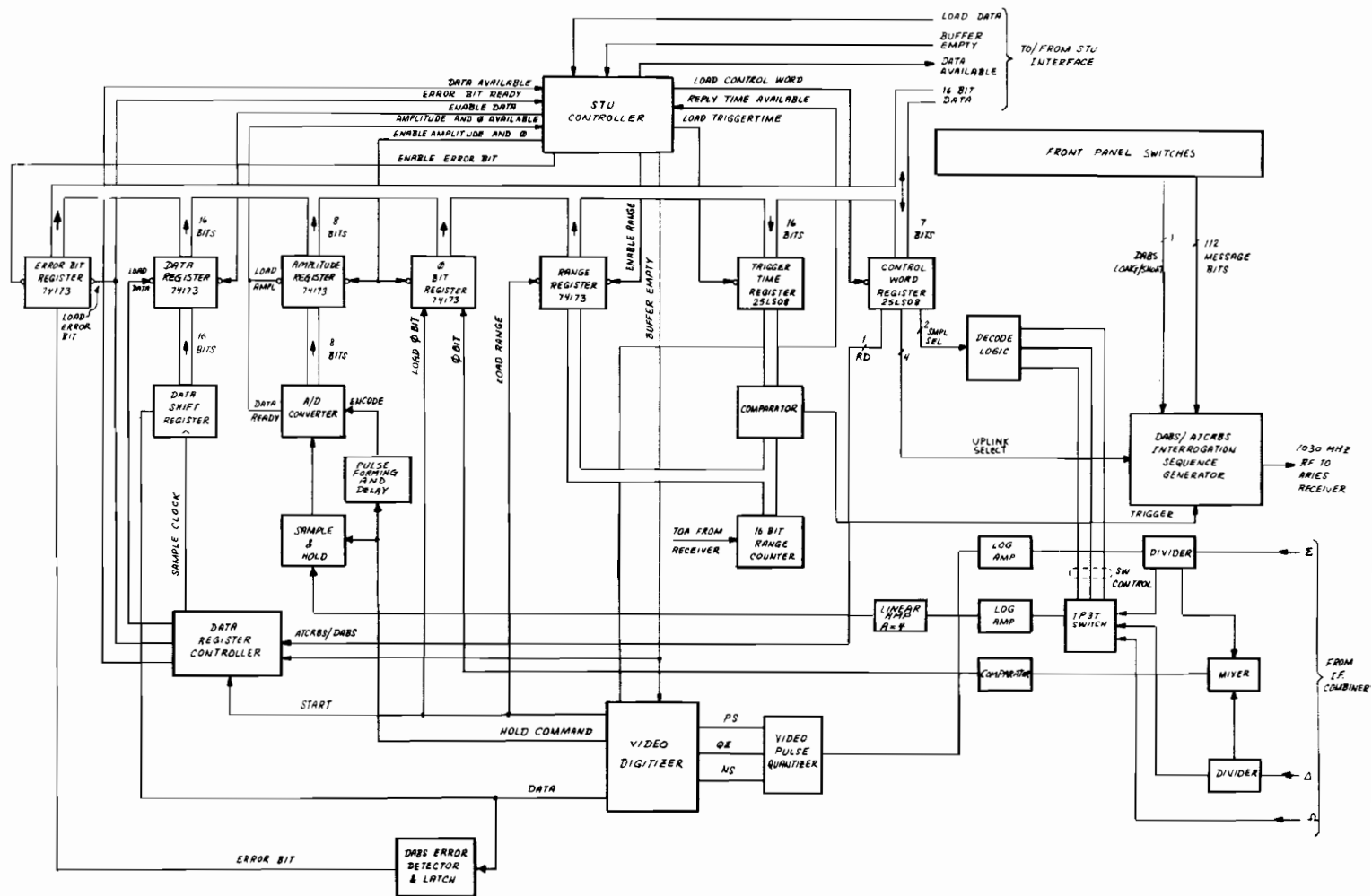


Fig. 2.8-1. STU block diagram.

As shown in Fig. 2.8-2, central to the digital section are two pre-programmed ROMS. One contains the interrogation preambles and the other contains the messages and addresses for DABS interrogations. Front panel programmed interrogations or any one of the ten pre-programmed interrogations in the ROM may be selected by specifying the proper value of the uplink select field in the control word (see Volume 3 of this document). The uplink select word is decoded by the decode ROM which enables the 8:1 multiplexers to output the appropriate interrogation. The ten pre-programmed interrogations and their messages are shown in Table 2.8-1. Note that these are the messages as sent to the receiver, i.e., after uplink encoding has been performed. The receiver will perform the corresponding decoding operation, and so the DABS address fields in the CPU memory after processing by the receiver will be different from those shown.

An interrogation is initiated each time the 9-bit address counter receives an uplink trigger pulse from the comparator. This pulse resets the counter to zero. As the ROM's are sequentially addressed, the interrogation preambles followed by the message (DABS only) will be forwarded to the RF section.

As may be seen in Fig. 2.8-3, the preamble and message gate input from the ROM is used to key the 1030 MHz oscillator to form the preamble pulses, and if the interrogation is a DABS type, it also allows the 1030 MHz CW to be modulated by the DPSK input from the ROM.

The DPSK modulator used in the RF section is a balanced mixer type. The phase reversal of the RF signal is accomplished by reversing the polarity of the DPSK input to the mixer. The phase reversal happens under control of the toggle flip-flop (Fig. 2.8-2) as follows. When a "1" is to be transmitted, a "1" from the ROM will enable the toggle flip-flop to change state, causing a phase reversal of the RF signal at the beginning of a bit interval representing a binary one. Alternatively the toggle flip-flop will not change state if a "0" is received so no phase reversal will occur and a binary zero will be represented. A waveform for a DABS interrogation is illustrated in Fig. 2.8-4.

Note that in Fig. 2.8-2 the counter will automatically disable itself when a full count is reached. It remains in this state until another uplink trigger is received.

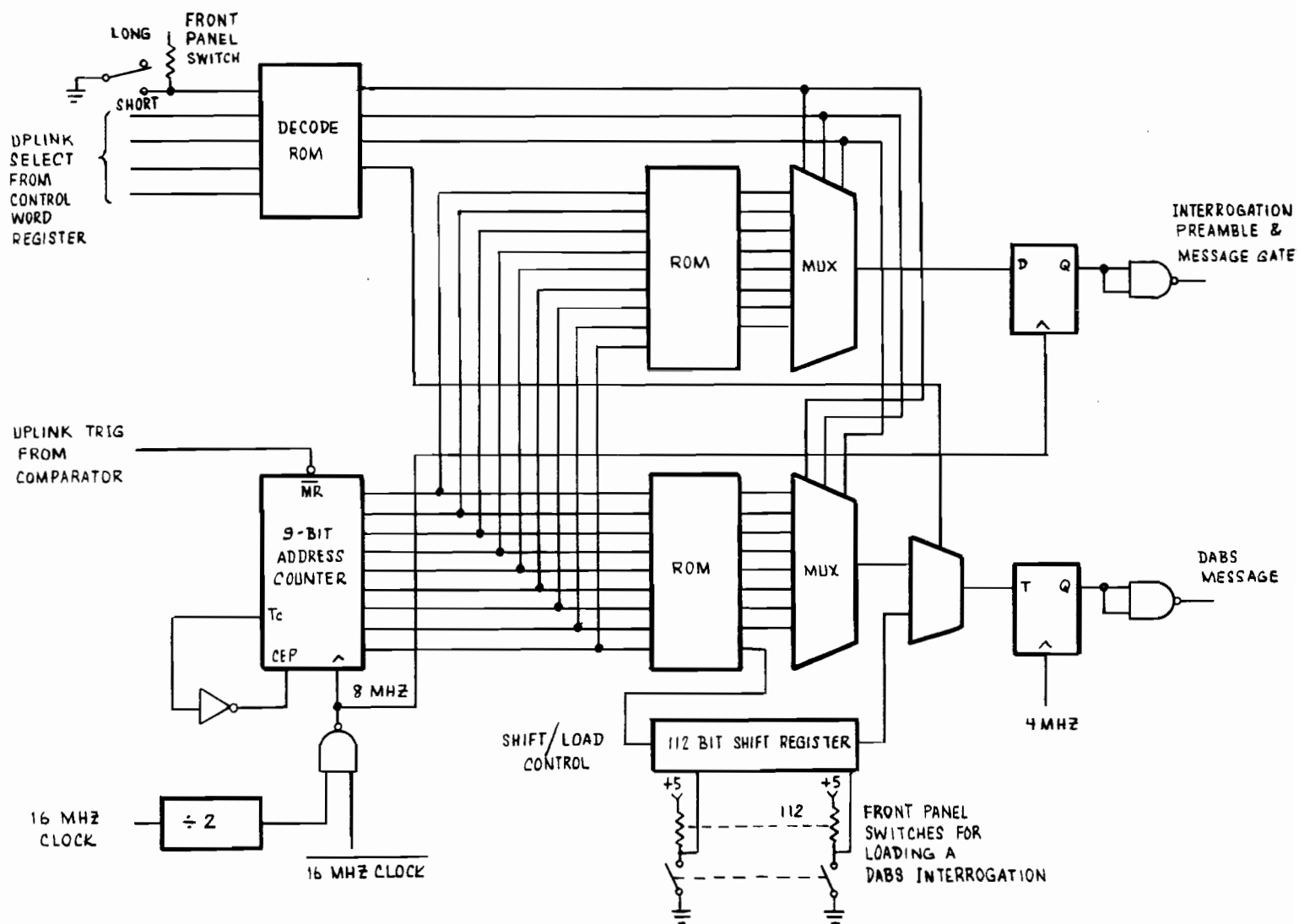


Fig. 2.8-2. Interrogation generator (digital section).

TABLE 2.8-1

<u>Interrogation Type</u>	<u>Data*</u>	
(1) ATCRBS/DABS All-Call Mode A	-	
(2) ATCRBS/DABS All-Call Mode C	-	
(3) DABS-only All-Call	127777	
	177777	
	137010	
	023000	
(4/10) DABS Surveillance	(4) 020000	(10) 026377
	000000	125252
	100146	054606
	057400	122000
(5) ATCRBS Mode A (No P4 Pulse)	-	
(6) ATCRBS Mode C (No P4 Pulse)	-	
(7) ATCRBS Mode D (No P4 Pulse)	-	
(8/9) DABS Comm-A	(8) 077777	(9) 070604
	177777	000000
	177777	000000
	177777	000000
	177777	000000
	177502	000071
(11-14) Not used	040241	041660
(15) Front Panel Programmed DABS Interrogation	Programmable	

\* Data is given in octal, and each row represents a 16 bit word.

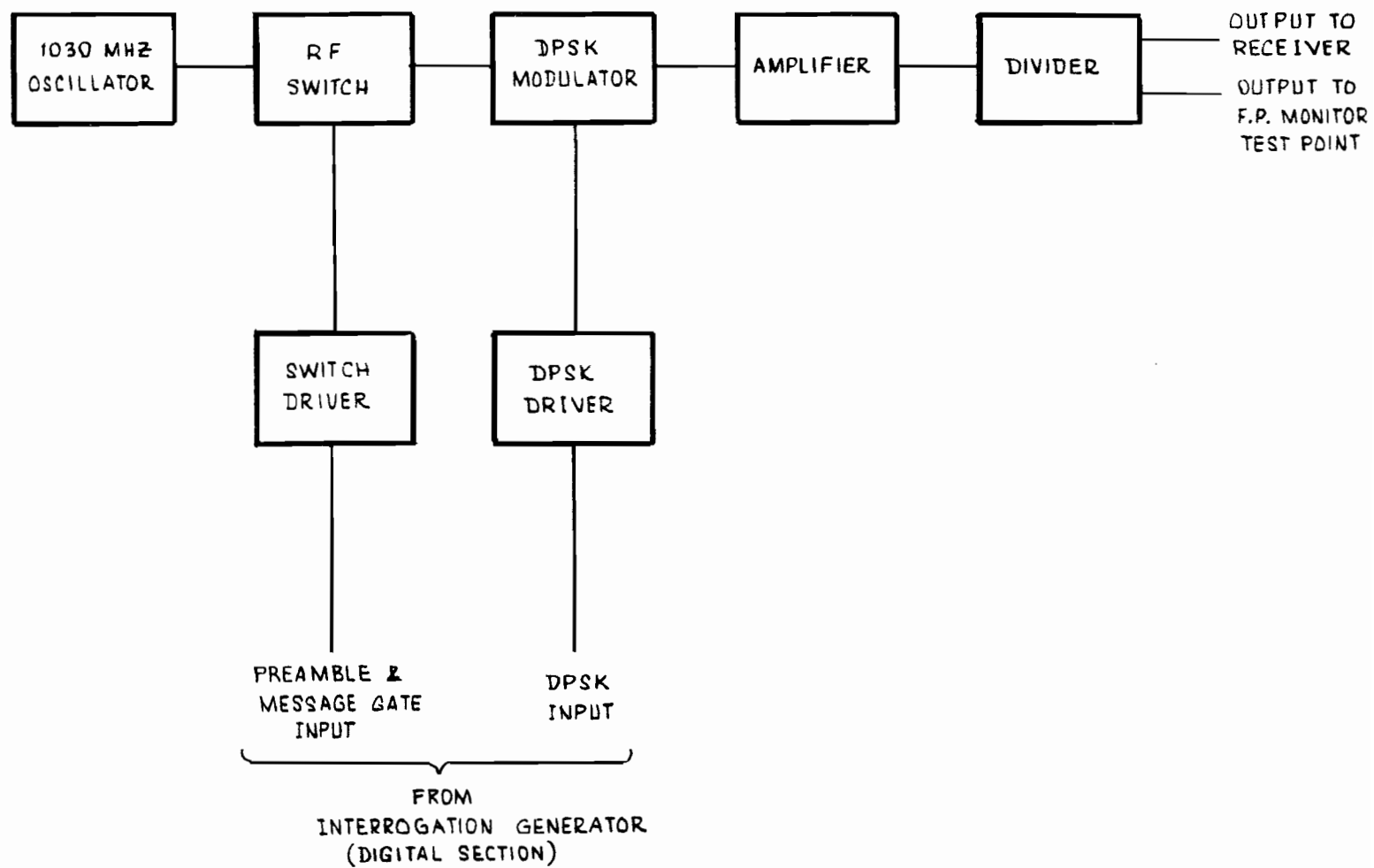
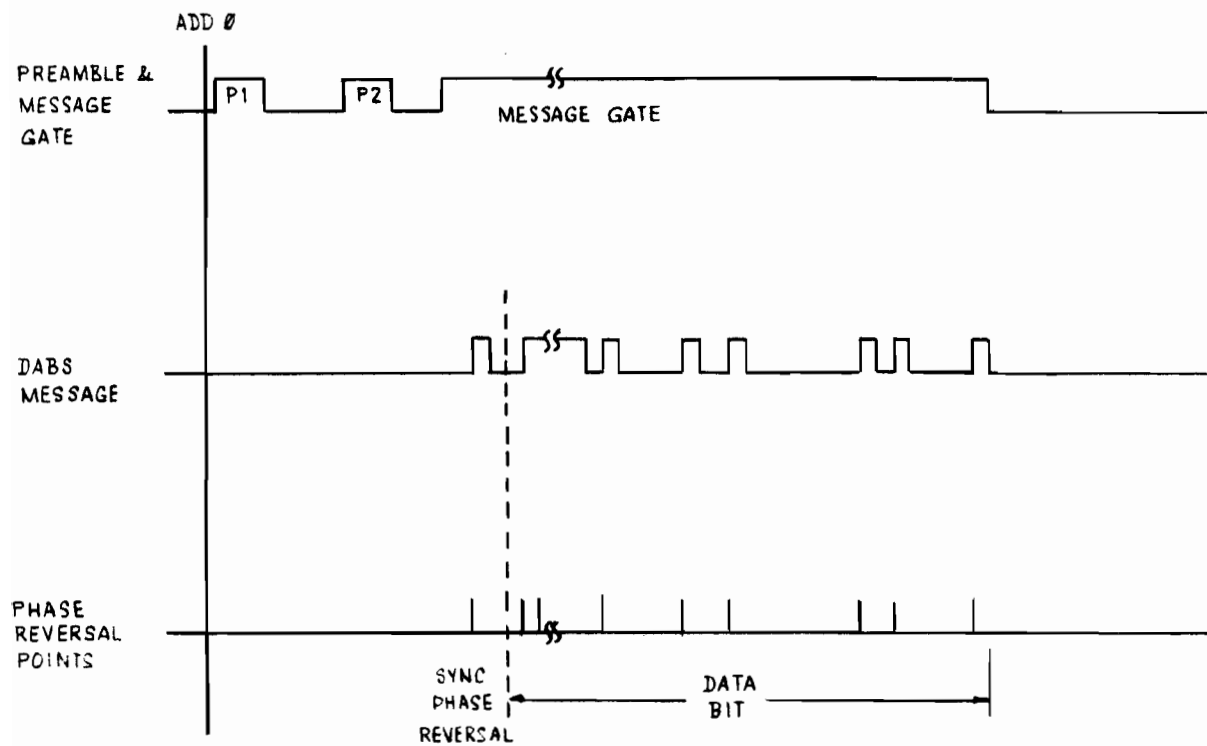


Fig. 2.8-3. Interrogation generator (RF section).



DATA BIT REPRESENTING 0111...1010000100100000010100001

Fig. 2.8-4. Comm-A interrogation waveform.

### 2.8.3 Reply Sampling

Any of the three outputs ( $\Sigma$ ,  $\Delta$ ,  $\Omega$ ) from the IF Combiner may be selected for amplitude sampling. This is done by controlling a 1P3T diode switch via the sample select (SSEL) field in the control word. The selected IF signal is amplified and detected by a log amplifier. The resultant video pulse is then sampled and converted to a 10-bit binary word by the sample and hold (S/H) amplifier and analog/digital (A/D) converter respectively. (S/H and A/D are controlled by the video digitizing circuit to be explained shortly).

Before the output from the log amp is applied to the S/H, it is further amplified by a linear amplifier providing an input range of 0 to 10 volts. The hold command for the S/H is delayed so that it falls in the middle of the input pulse, see Fig. 2.8-5.

In addition to the amplitude measurement, the phase difference (assumed to be  $0^\circ$  or  $180^\circ$ ) between the  $\Sigma$  and  $\Delta$  channels may also be measured. This is accomplished by multiplying the  $\Sigma$  and  $\Delta$  channel signals together. This technique assumes that the  $\Sigma$  and  $\Delta$  amplitude levels are identical. The result from the mixer is used to set the  $\phi$  bit. ("0" for  $0^\circ$  and "1" for  $180^\circ$ ). This setup provides a means of checking the phase shifter in the analog target generator. (See Section 2.5.1).

Data extraction and reply detection are done by a second log amplifier connected to the  $\Sigma$  channel. Video output from the amplifier is quantized by the VPQ circuits, providing the positive slope (PS), negative slope (NS) and quantized sum ( $Q\Sigma$ ) outputs to the video digitizer.

A simplified diagram of the VPQ circuit is shown in Fig. 2.8-6. Input from the  $\Sigma$  channel, after amplification, follows two paths, one delayed by 125 nsec with respect to the other. The two signals are then subtracted from each other, producing a positive and a negative pulse corresponding to the positive and negative slope of the input signal from the  $\Sigma$  channel (see Fig. 2.8-7).

The amplified input from the  $\Sigma$  channel is also applied to a comparator, where it is constantly compared to the MTL (set at -1.25 V for -79 dB).  $Q\Sigma$  is "true" only if the  $\Sigma$  input is above the MTL, therefore it is used for gating the PS pulse in the video digitizer, see Fig. 2.8-8. If the input pulse level is above MTL, the resulting PS pulse will be synchronized and the J-K flip-flop ( $IC_1$ ) will be set. It remains set until the synchronized NS pulse arrives, thus a TTL pulse with the same pulse width as the input is created. If this is the first pulse of the reply, the following events will occur:



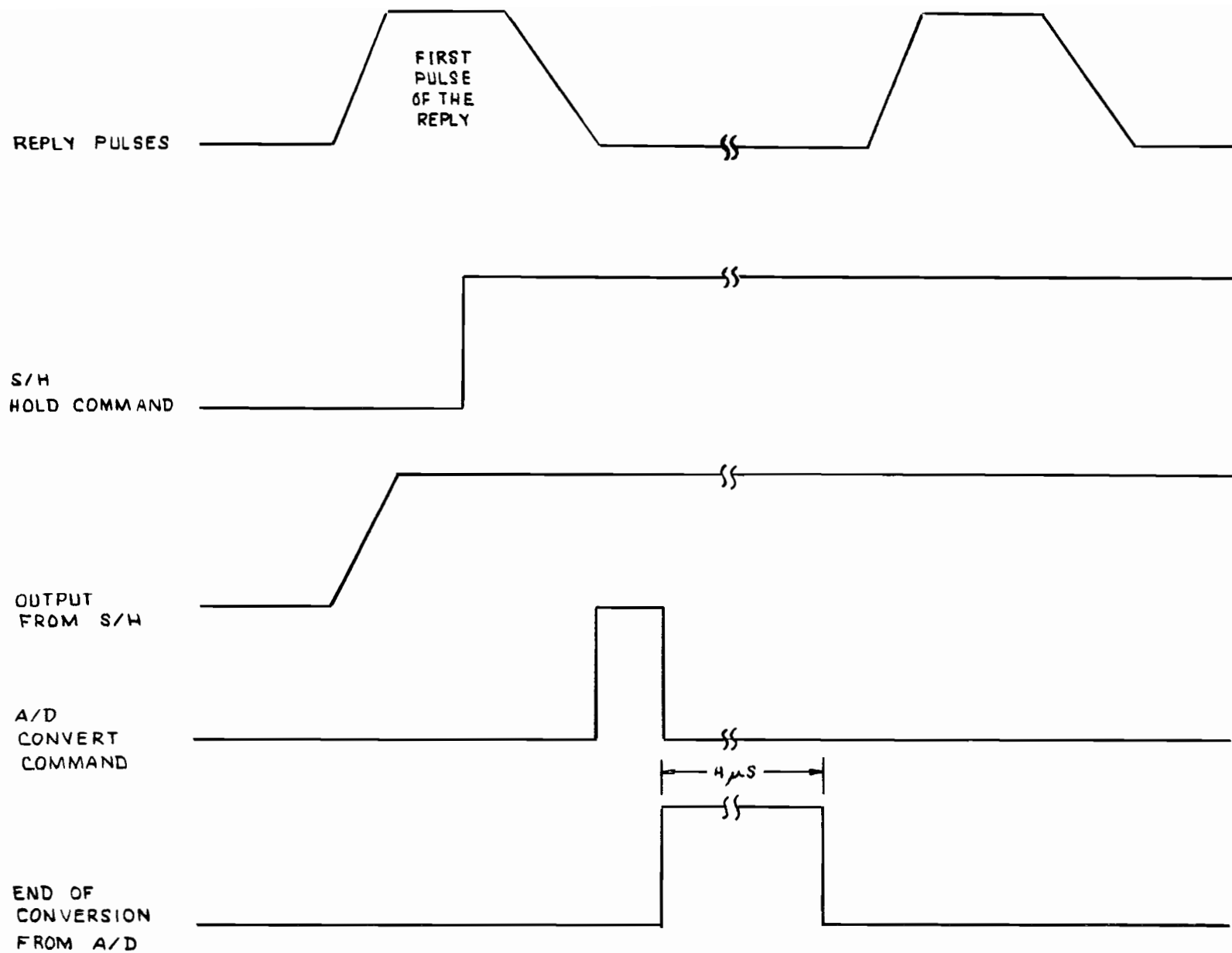


Fig. 2.8-5. Reply sampling waveforms.

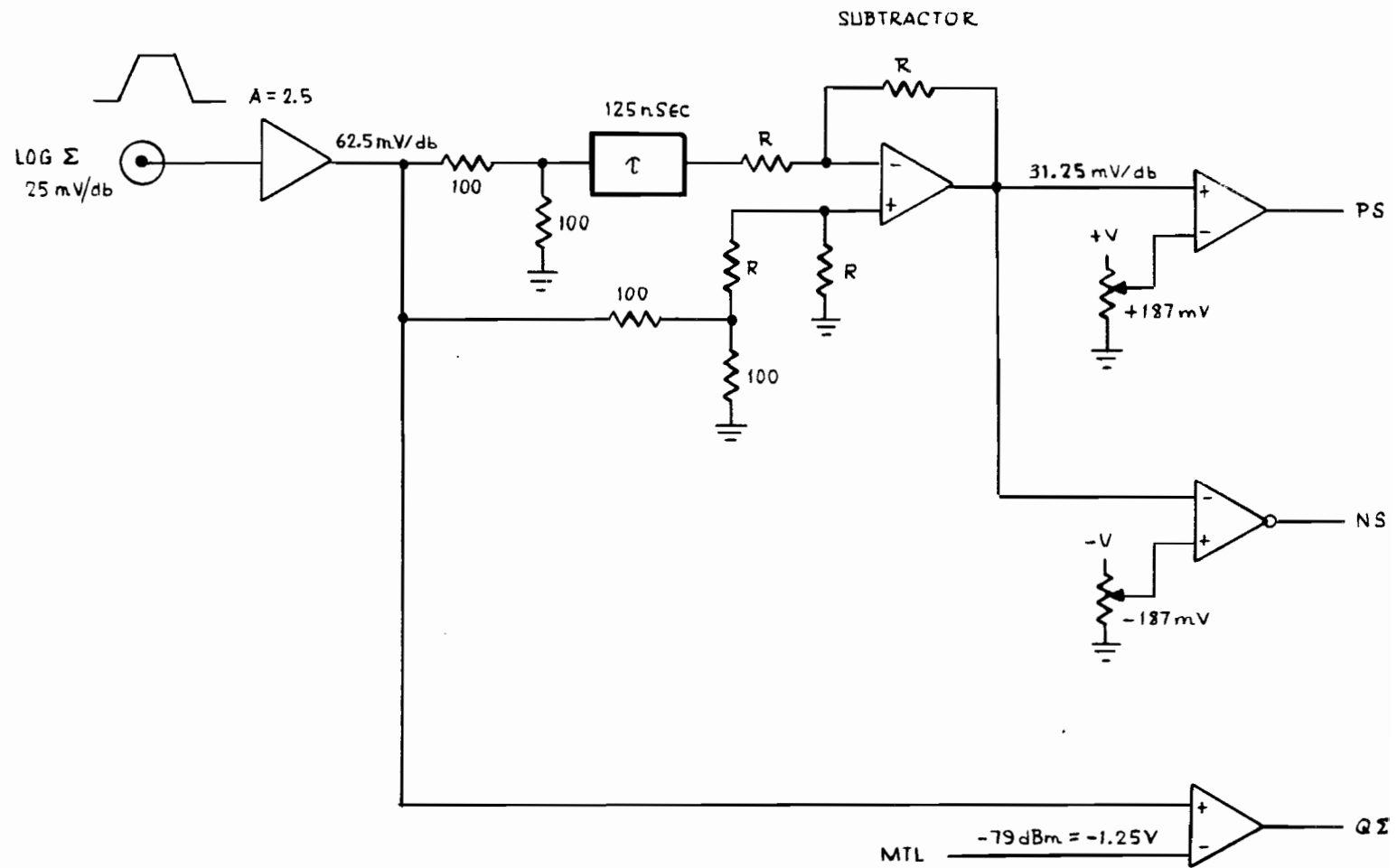


Fig. 2.8-6. Simplified diagram of VPQ.

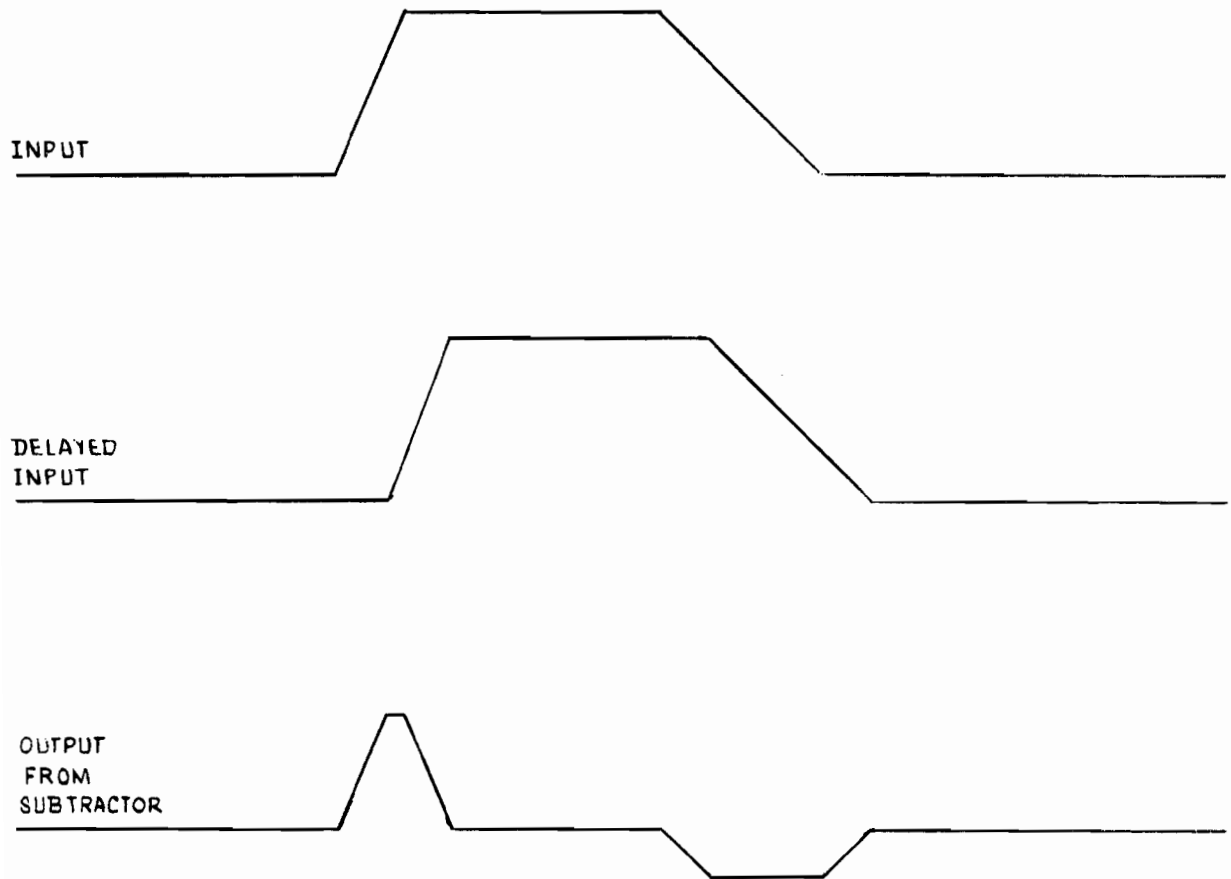


Fig. 2.8-7. VPQ waveforms.

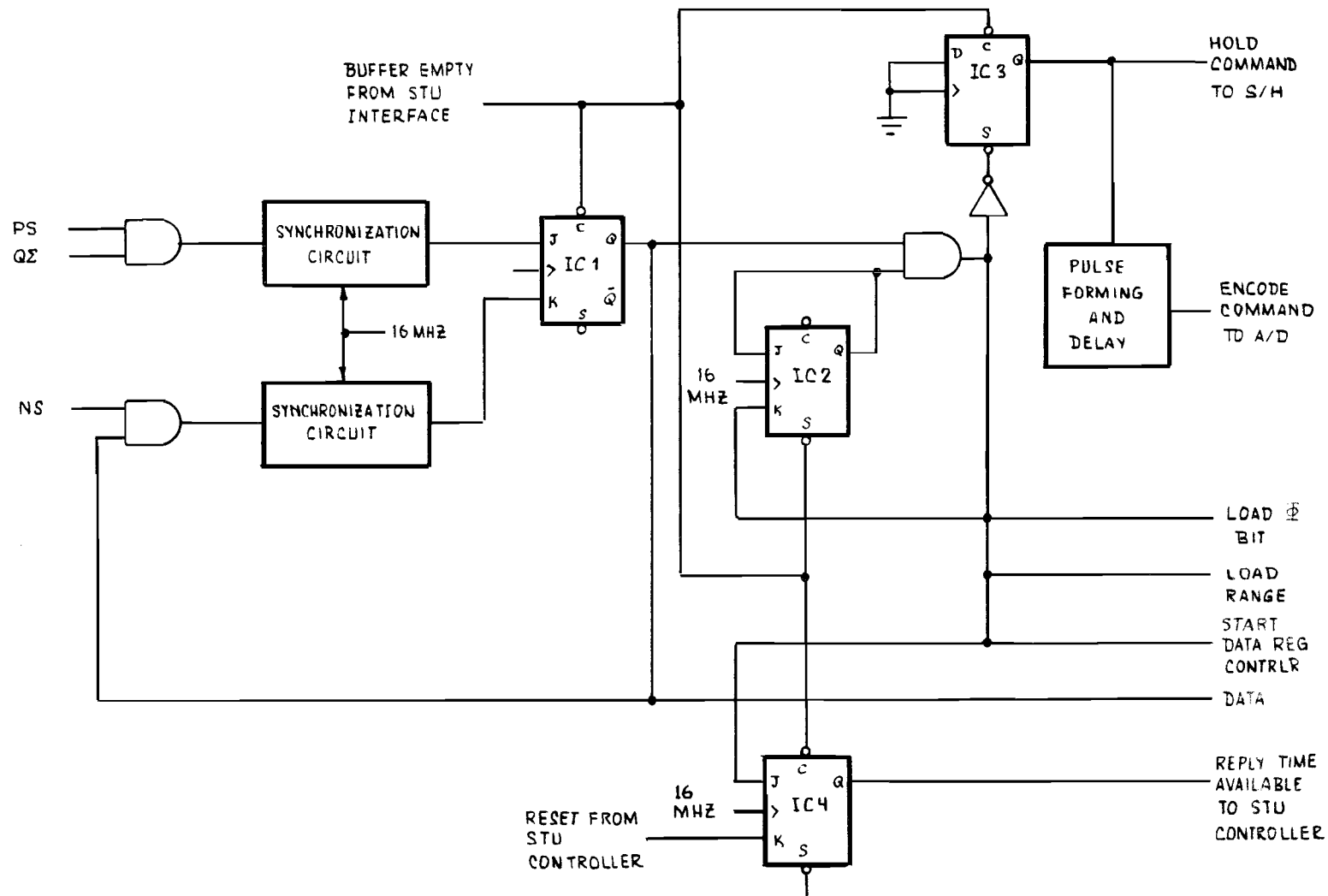


Fig. 2.8-8. Video digitizer.

1. The phase bit ( $\Phi$ ) and range will be loaded into their appropriate registers.
2. The range available bit ( $IC_4$ ) will be set.
3. The S/H amplifier will be set into a hold mode (set  $IC_3$ ), and 250 ns later a convert command will be sent to the A/D.
4. The data register controller will be started.

$IC_2$  will latch up after the first pulse, so that the above operations will occur only once, on the first pulse of each reply.  $IC_1$ ,  $IC_2$ ,  $IC_3$ , and  $IC_4$  are initialized by the CPU (buffer empty) before a reply is sent to the STU.

The A/D converter used in the STU is a successive approximation type with a conversion time of 4  $\mu$ sec. When a conversion is finished, a data ready pulse is generated, causing the result of the conversion to be loaded into the amplitude register, and a bit indicating this to the STU controller.

#### 2.8.4 Data Register Controller

Reply data from the Video Digitizer are "sampled" into a shift register. After 16 shifts the results are then parallel loaded into the "data register" and subsequently transferred to the CPU, see Fig. 2.8-1. The sampling and loading operations are controlled by the Data Register Controller.

A simplified diagram of the Controller is presented in Fig. 2.8-9. It consists of a 9-bit address counter, a programmed ROM, and a multiplexing circuit that selects different outputs from the ROM.

The address counter is zeroed when a start command is received from the Video Digitizer. As the Counter counts up at the 4 MHz rate, the sampling and loading commands will be output from the ROM. Since the ROM provides both DABS and ATRBS commands, it is necessary to multiplex the outputs. Multiplexer M1 and M2 provide this function.

The data rate for DABS replies is 1 MBPS. It is easy to program the ROM to sample the data every 1  $\mu$ sec, as the system clock is running at 4 MHz (see Fig. 2.8-10). For ATRBS replies, the timing differs from that of the system clock and its multiples; therefore it is not possible to program the ROM to sample the ATRBS reply data directly, and a delay scheme must therefore be used. Four ROM outputs (A, B, C, and D) together with multiplexer

Fig. 2.8-8. Video digitizer.

1. The phase bit ( $\Phi$ ) and range will be loaded into their appropriate registers.
2. The range available bit ( $IC_4$ ) will be set.
3. The S/H amplifier will be set into a hold mode (set  $IC_3$ ), and 250 ns later a convert command will be sent to the A/D.
4. The data register controller will be started.

$IC_2$  will latch up after the first pulse, so that the above operations will occur only once, on the first pulse of each reply.  $IC_1$ ,  $IC_2$ ,  $IC_3$ , and  $IC_4$  are initialized by the CPU (buffer empty) before a reply is sent to the STU.

The A/D converter used in the STU is a successive approximation type with a conversion time of 4  $\mu$ sec. When a conversion is finished, a data ready pulse is generated, causing the result of the conversion to be loaded into the amplitude register, and a bit indicating this to the STU controller.

#### 2.8.4 Data Register Controller

Reply data from the Video Digitizer are "sampled" into a shift register. After 16 shifts the results are then parallel loaded into the "data register" and subsequently transferred to the CPU, see Fig. 2.8-1. The sampling and loading operations are controlled by the Data Register Controller.

A simplified diagram of the Controller is presented in Fig. 2.8-9. It consists of a 9-bit address counter, a programmed ROM, and a multiplexing circuit that selects different outputs from the ROM.

The address counter is zeroed when a start command is received from the Video Digitizer. As the Counter counts up at the 4 MHz rate, the sampling and loading commands will be output from the ROM. Since the ROM provides both DABS and ATRBS commands, it is necessary to multiplex the outputs. Multiplexer M1 and M2 provide this function.

The data rate for DABS replies is 1 MBPS. It is easy to program the ROM to sample the data every 1  $\mu$ sec, as the system clock is running at 4 MHz (see Fig. 2.8-10). For ATRBS replies, the timing differs from that of the system clock and its multiples; therefore it is not possible to program the ROM to sample the ATRBS reply data directly, and a delay scheme must therefore be used. Four ROM outputs (A, B, C, and D) together with multiplexer

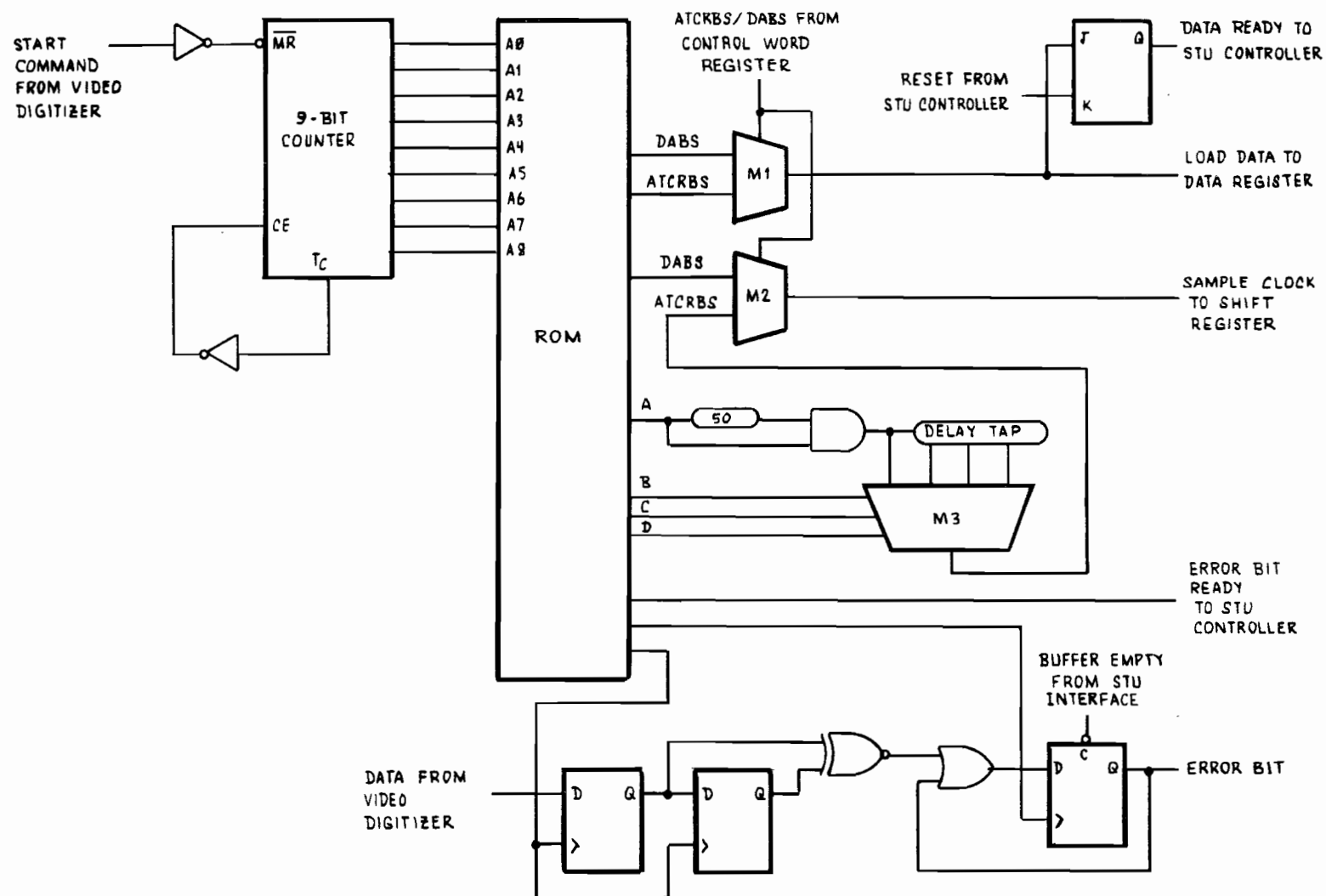


Fig. 2.8-9. Data Register Controller.



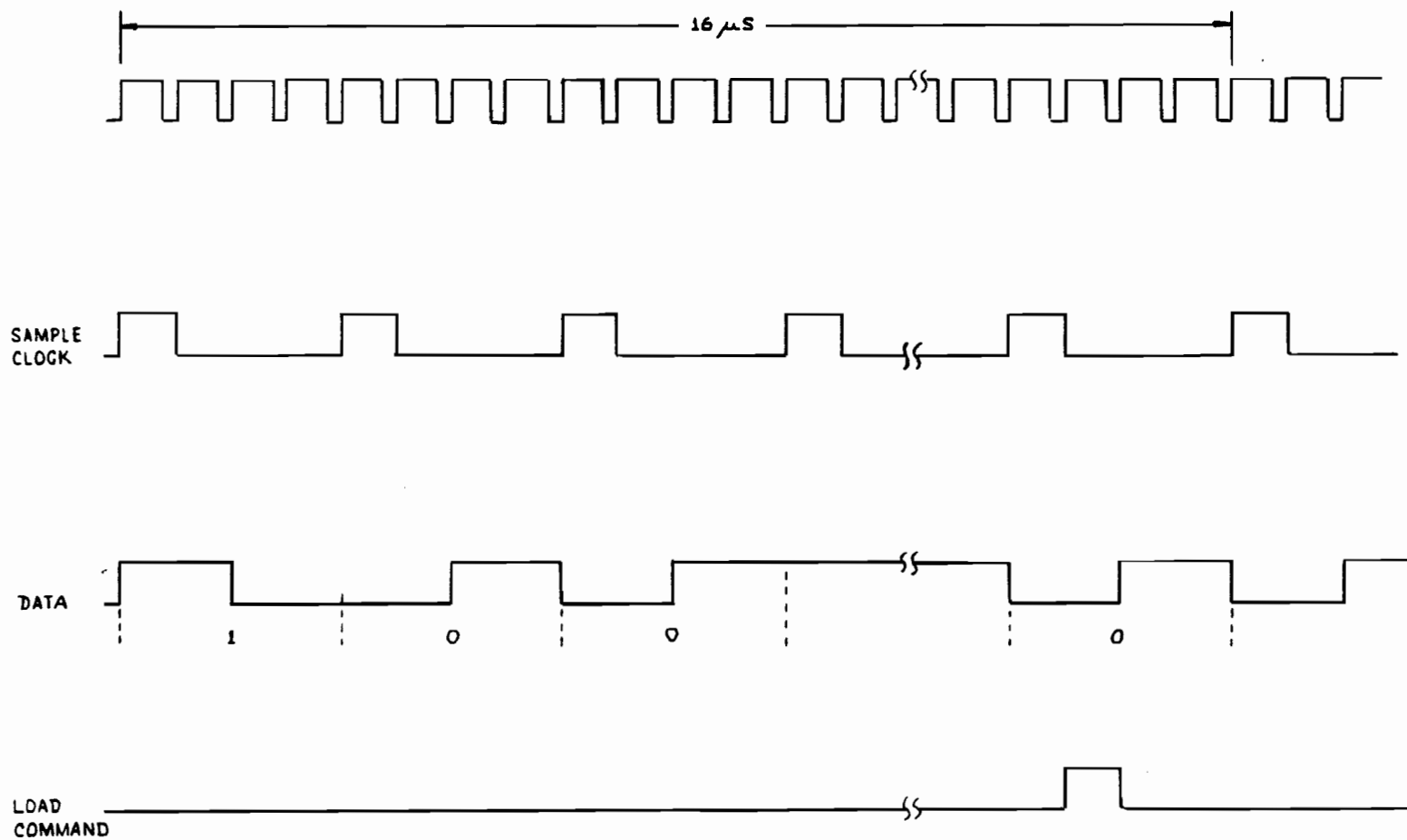


Fig. 2.8-10. Sample and load command for DABS replies.

M3 and a multiple tap delay element provide the sample clock for the ATCRBS reply. This scheme is identical to one used in the digital target generator (see Section 2.4.1.2.4.2). When the 9-bit counter reaches the maximum count it automatically disables itself.

DABS reply data are modulated using a pulse position modulation (PPM) technique. A logical "1" is represented by a 0.5  $\mu$ s pulse in the first half of a 1  $\mu$ sec data bit interval, and a logical "0" by a 0.5  $\mu$ s pulse in the second half. To provide a check on the PPM format, format error detection is designed into the STU. This is shown in Fig. 2.8-9. The same data that goes from the Video Digitizer to the shift register is also applied to the error detector. It is basically a two-bit shift register, with the output summed by an exclusive nor gate. Whenever an illegal "0,0", or "1,1" occurs from the shift register, the result will be latched in the D-flip-flop and subsequently reported to the CPU via the error bit register. Note that the clock for the two-bit shift register and the D-flip-flop are coming from the ROM, so that the D-flip-flop will be clocked only after two halves of a data bit have been shifted into the register. The D-flip-flop is reset by the CPU before a reply is sent to the STU.

#### 2.8.5 STU Controller

For each reply received, the STU will transfer a block of eleven words to the CPU. These consist of reply time, reply amplitude, phase bit (LR), uplink select bits (UPSEL), sample select bits (SSEL), mode select bit (RD), error bit (EB), and reply data bits. These data are held by different registers as shown in Fig. 2.8-1.

Output of these registers are tied together forming a common open collector bus. Each register is then enabled by the STU Controller at the appropriate time so that the data are transferred to the CPU in the format shown in Volume 3 of this document.

Operation of the STU Controller may be understood by examining the state diagram of Fig. 2.8-11. State  $\emptyset$  is the initial state of the Controller, entered whenever the STU is reset, either by buffer empty or IORST/CLR from the STU interface. It remains in this state until the first pulse of the reply is detected by the Video Digitizer, causing the reply time available bit to be set ( $IC_4$  in the Video Digitizer). This bit allows the Controller to advance to State 1, thereby enabling the range register to output the data to the bus. At this time the A/D converter is converting the reply pulse amplitude. States 2 and 3 wait for the end of the conversion signal from the A/D and enable the phase bit and amplitude register respectively.

For diagnostic purposes, the control word sent by the CPU is echoed back by State 4. Since it takes 16  $\mu$ sec to assemble 16 reply data bits, an idle state is needed to wait for the completion. This is the purpose of State 5. As soon as a 16-bit data word is ready, State 6 will enable the "data register" on the bus. States 5 and 6 will repeat until all 7 data words are transferred. Since ATRBS replies have only one 16-bit reply word, only the first word will be valid. The other six are transferred as "don't care" words. Finally, State 7 enables the error bit register when it is valid, as indicated by the error bit "ready" from the Data Register Controller.

When State 7 is terminated, State  $\emptyset$  will be re-entered, and the whole process will be repeated for the next reply. Although it is not shown in the state diagram, each 16-bits of data transferred to the interface is accompanied by a data available pulse. The STU interface uses this pulse to latch in the data words (see section on the STU interface). Since there is sufficient time between each transfer, no handshake from the interface is required.

#### 2.8.6 STU Interface

The STU Interface controls the operating mode of the Self Test Unit and, in the Reply Sampling mode, provides buffer space for storing the 11 data words from the STU.

To place the STU Interface in operation, (i.e., after power up), either an IORST or CLR command must be issued by the CPU. These two instructions reset the D flip-flops  $IC_1$  and  $IC_2$  to zero, (see Fig. 2.8-12) and clear the control register in the Self Test Unit.

Since two control words are required by the STU, they must be stored in two separate registers, A and B as shown in Fig. 2.8-12. The outputs Q and  $\bar{Q}$  of  $IC_1$  are used to gate the clock input of these registers, so two suc-

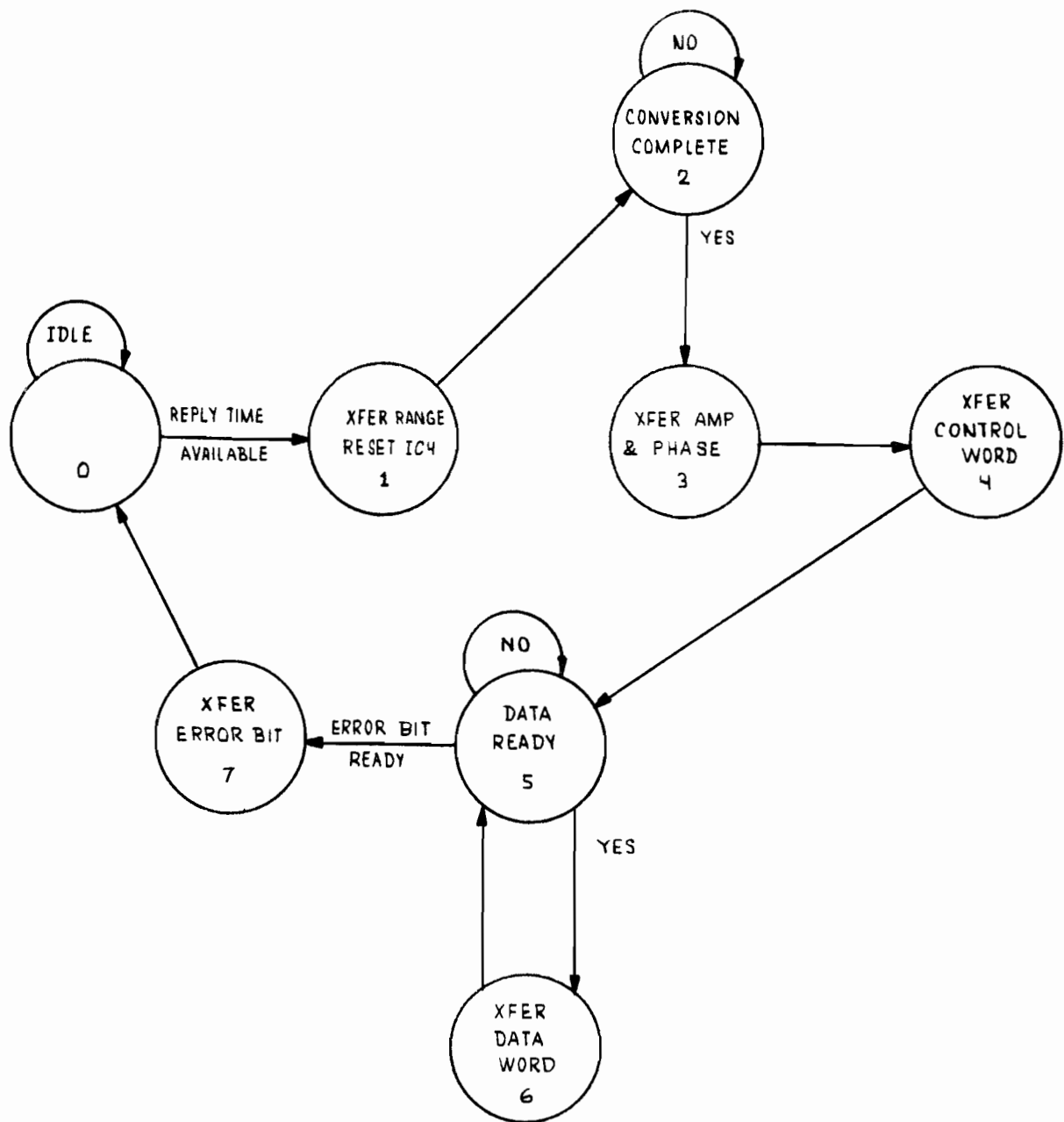


Fig. 2.8-11. State diagram of STU controller.

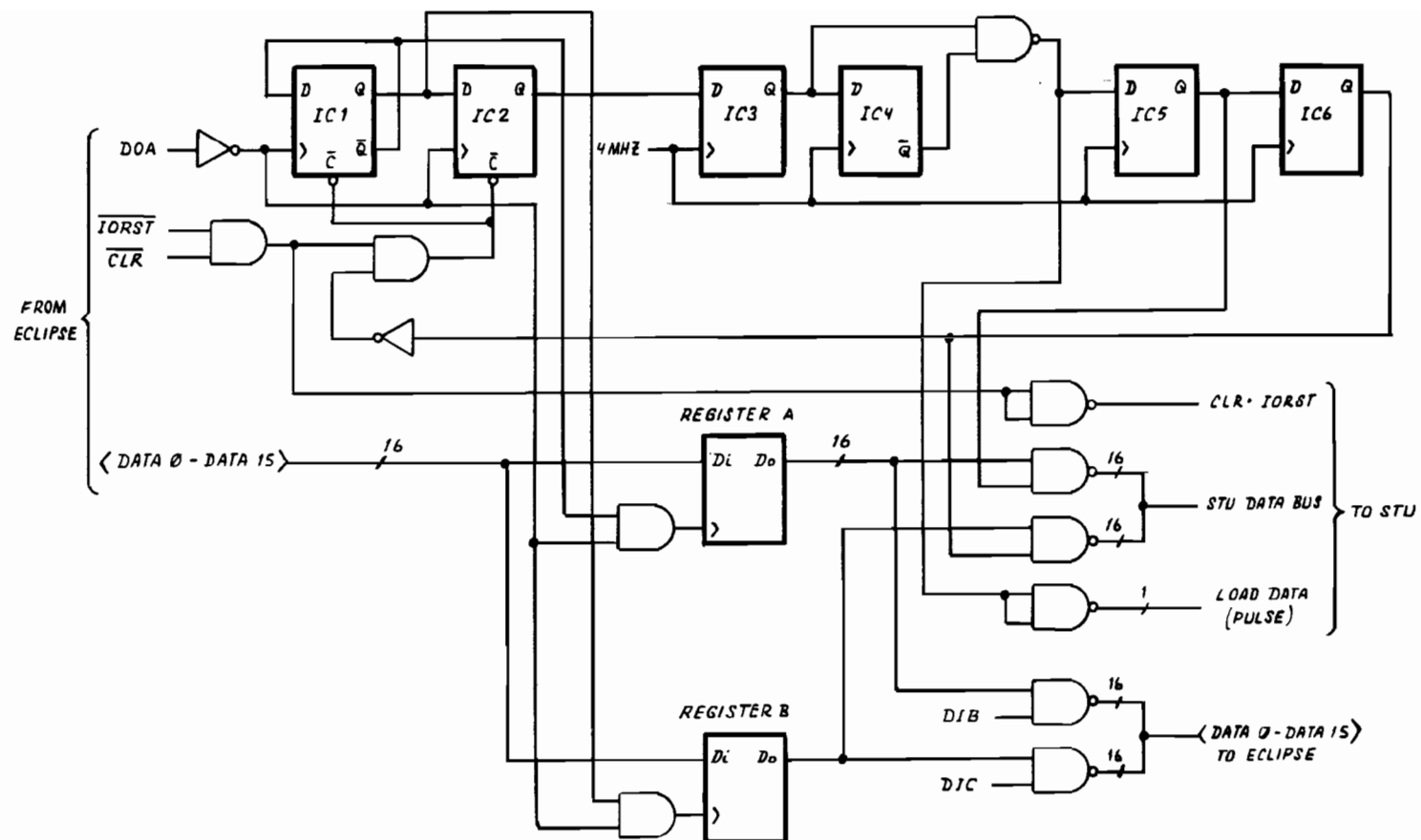


Fig. 2.8-12. Control registers.

cessive DOA's from the CPU will place two control words in the appropriate registers. At the end of the second DOA, IC<sub>2</sub> will be set causing a LOAD DATA pulse to be generated by the edge detecting circuit (IC<sub>3</sub> and IC<sub>4</sub>). Besides causing output to the STU, the LOAD DATA pulse, after being delayed by IC<sub>5</sub> and IC<sub>6</sub>, places the contents of registers A and B on the STU data bus. The STU uses the LOAD DATA pulse to latch-in the two control words.

For diagnostic purposes, the contents of registers A or B may be examined by the CPU if the DIB or DIC instruction is executed.

In the interrogation generation mode only the two control words must be specified. However, for the reply sampling mode a START instruction must also be issued by the CPU. The purpose of this instruction is to (1) reset the STU interface and (2) prepare the STU to look for a reply.

When the STU Interface receives a START pulse, BUSY will be set, in turn causing a Buffer Empty pulse to be sent to the STU and the address counter for the memory be reset (see Fig. 2.8-13). The Buffer Empty pulse causes the STU to look for a new reply. At this time, if a reply is generated by either a CAT or FAT, it will be detected by the STU. The 11 words resulting will be transferred to the STU Interface. Each word transferred is accompanied by a Data Available pulse. This pulse gates the data into the memory and increments the memory address counter. At the end of the eleventh word, a CPU interrupt is generated and the memory address is reset to the first location. Note that only locations 5 through 15 of the memory are used, since there are only 11 words to be transferred.

When the CPU processes the interrupt, it can obtain the data by executing 11 successive DIA's. After each DIA, the memory counter will be incremented. It is not necessary for the CPU to read all 11 words or any of them. Issuing a START pulse will initiate a new reply listening cycle for the STU.

## 2.9 Radar Report Interface

The Radar Report Interface (RRI) reformats radar reports resident in the CPU's memory into Production Common Digitizer (PCD) formatted reports for serial transmission to the radar input port of the DABS sensor.

This device uses the data channel method of transferring data, and can therefore output a complete block of data from memory once the starting address and word count are specified.

Operation of the RRI may be understood by considering the following data transfer cycle. Before a data channel transfer is initiated, the CPU must specify the starting address of the memory and the word count. The

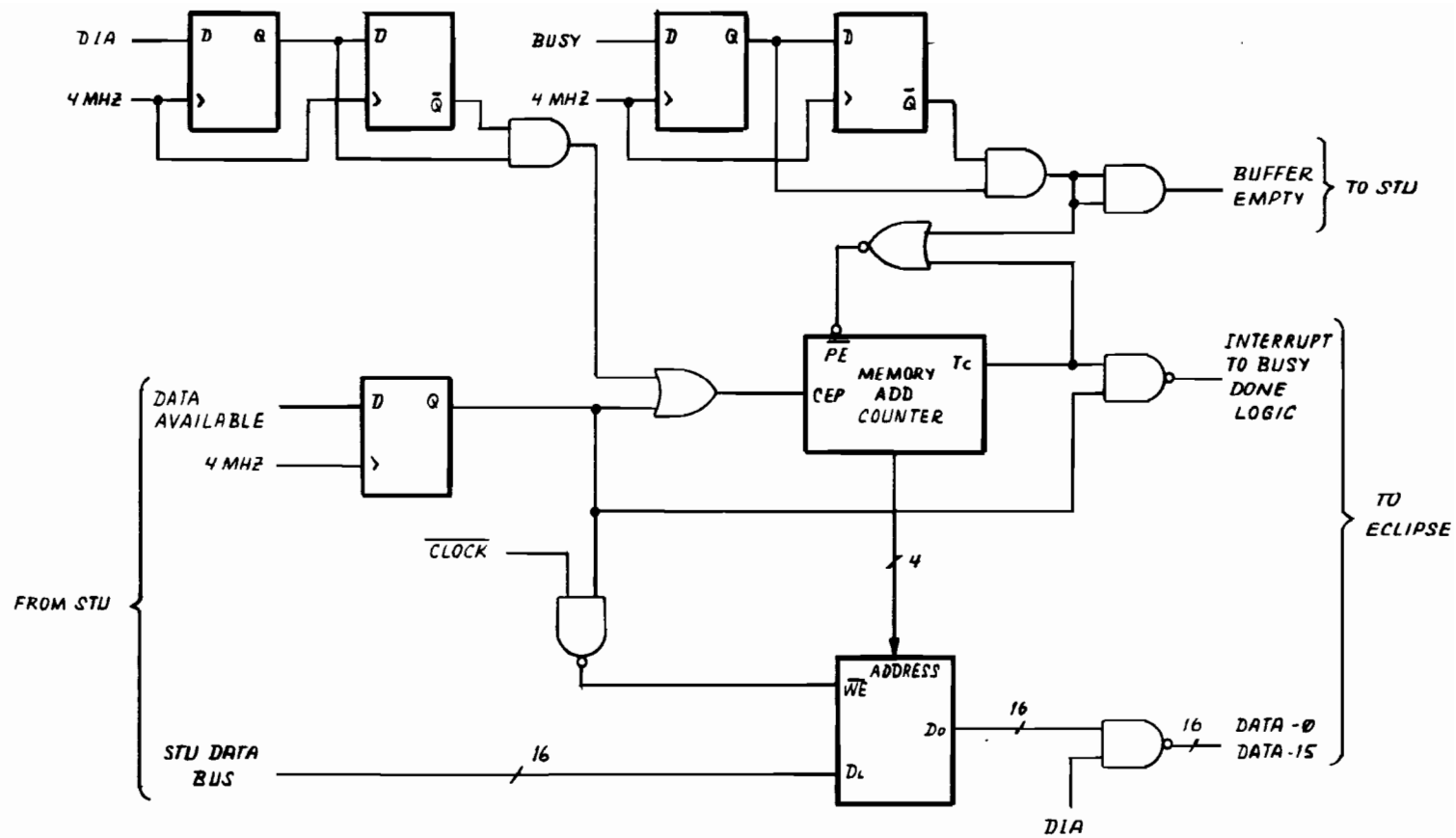


Fig. 2.8-13. 11-word buffer.

memory address and word count registers are shown in Fig. 2.9-1. To begin the data channel transfer operation, a START instruction is issued (i.e., the start pulse accompanies the instruction) setting the busy and clearing the done bit. As a result the DCH sync flip-flop is set, causing a data channel request to be sent to the CPU (see Fig. 2.9-2). When the data channel is made available the CPU transmits two control signals to the interface, DCHA followed by DCHO. DCHA outputs the contents of the memory address register to the data bus and increments the memory address and word count registers. DCHO latches the data from the addressed memory location into the data register (see Fig. 2.9-1).

Since operation of the data channel and the external clock are asynchronous some form of synchronization is required in order for the Interface to operate properly. This is the purpose of the Sync Flip-flop in Fig. 2.9-2. Each time DCHO is output from the CPU, the Sync Flip-flop will be set. This allows the multiplexer to pass data from the data register to the shift register. Note that Bit 4 - Bit 15 and Bit 0 from the data register go to a parity encoder before the shift register is loaded. Bit 0 is used to select even (1) or odd (0) parity. When the shift register finishes outputting the current word, shift/load from the "÷ 13 divider" will be set, allowing the new data from the multiplexer to be loaded into the shift register. Note that the output from the divider also resets the Sync Flip-flop and initiates another data channel request. Thus while the current word is being transferred, the next word will be fetched from memory. This process will continue until the end of the data block is reached. At that time, the word count register will contain zero, causing the done bit to be set, the busy bit to be reset, and an interrupt generated. Resetting the busy bit will prevent further data channel requests from being generated.

Note that when the Sync Flip-flop is reset to zero, as in the case of no data channel operations, the shift register will constantly receive an idle character (000 111 111 111 1) from the multiplexer, and consequently all data sent will consist of idle characters.

The data formats for the Radar Report Interface may be found in Vol. 3 of this document.

## 2.10 Random Number Generator Interface

This device provides the CPU with 16 bit uniformly distributed random integers. It consists of three T flip-flops and 14 D flip-flops cascaded as shown in Fig. 2.10-1. With output from the last D flip-flop fed back to the input to the first T flip-flop, a 17 bit maximal length sequence generator is formed. (See The Design of Digital Systems by John B. Peatman, p. 412).



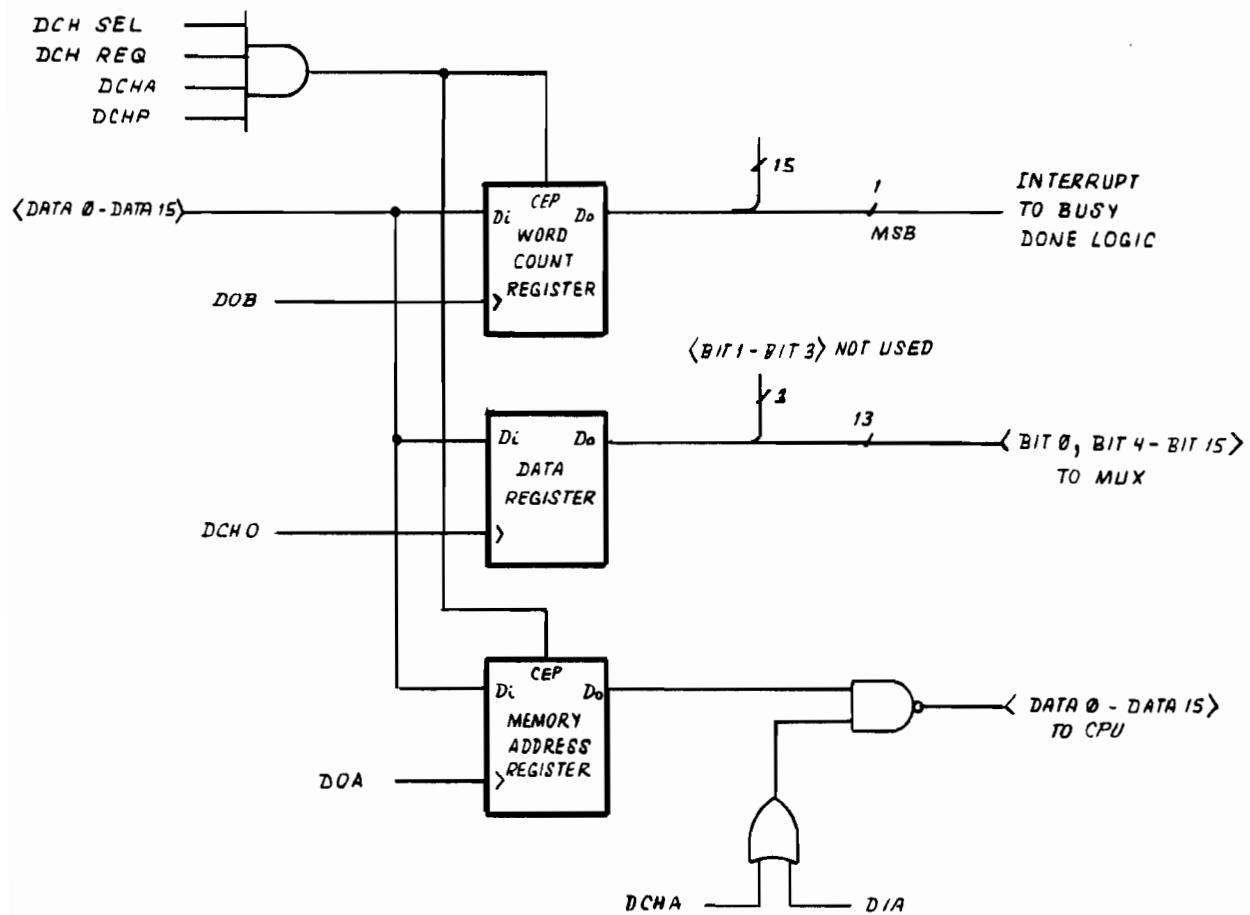


Fig. 2.9-1. Memory address and word count registers.

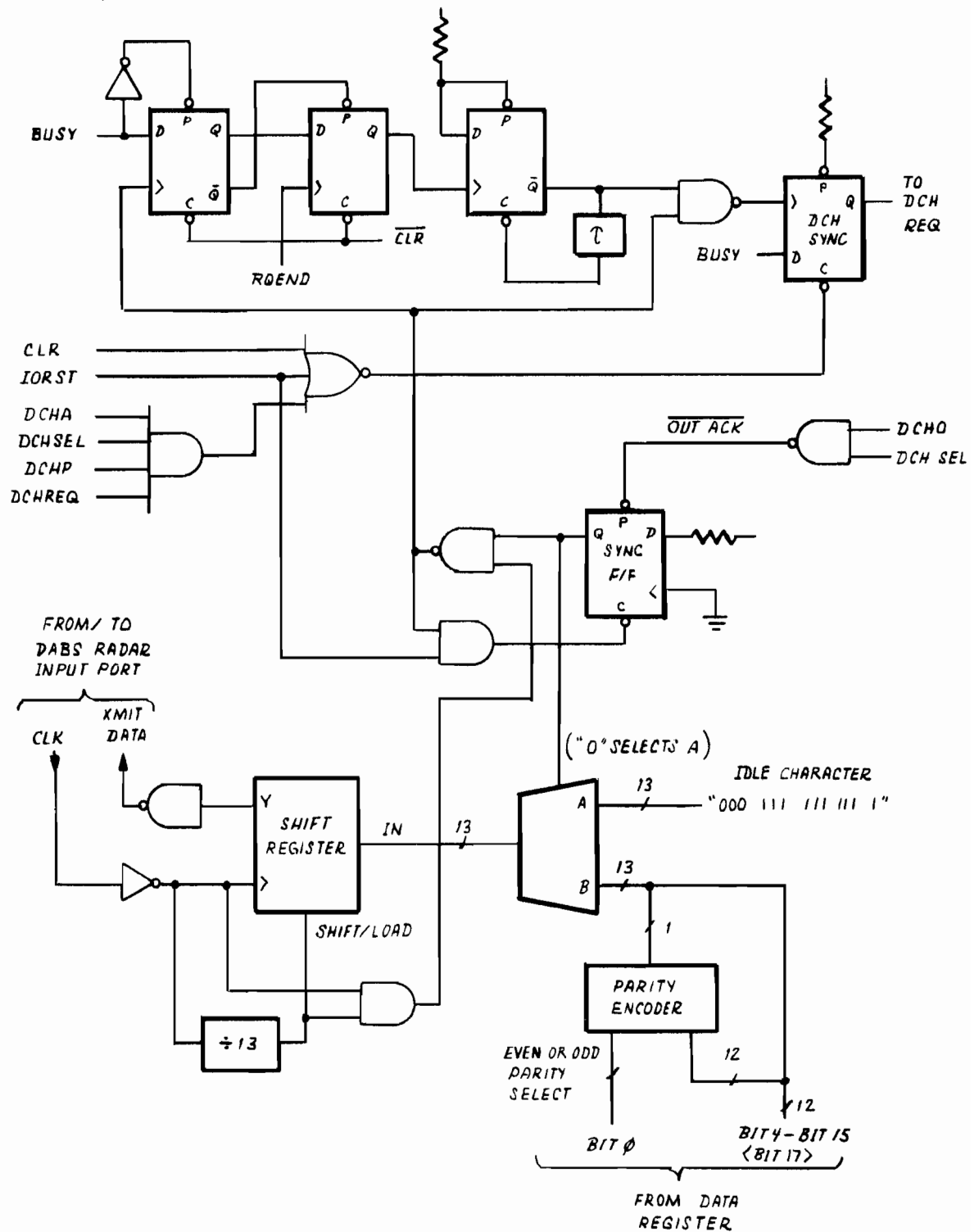


Fig. 2.9-2. Radar report interface.

Fig. 2.10-1. Random number generator interface.

Each time the generator is read by the CPU (DIA), it will be clocked once, and a new number produced.

This generator is capable of generating all  $2^{17}-1$  possible states, except zero. Since zero is an illegal state, (causing a lock-up condition) the generator must not start in this condition. To assure this, the generator is reset to state "111 000 ----00" during power up, or at any time by the IORST or NIOP instructions.

### 3.0 OPERATIONAL SOFTWARE

#### 3.1 Major Software Functions

ARIES' major software functions consist of:

1. Interrogation Processing
2. Pre-Interrogation Processing
3. Traffic Model Input
4. Inter-ARIES Communication
5. Timer Countdown
6. Operator Communications
7. System Initialization
8. Data Recording

These major functions communicate with one another and access global data structures as shown in Fig. 3.1-1. The operation of each function is summarized here and explained in detail in the following subsections\*.

Interrogation Processing performs several functions based on interrogation data input by the receiver, as follows:

- a) It generates reply data blocks using information in the Track File and transmits these to the controlled reply generator.
- b) It uses the data from ATCRBS/All-Call interrogations to provide antenna azimuth and azimuth rate information that can be used by the rest of the system.
- c) Whenever the antenna moves into a new azimuth sector Interrogation Processing looks up the fruit rate for that sector and initializes the fruit reply generator for the correct fruit rate. There are 32 sectors for fruit generation purposes.
- d) Whenever the antenna passes the starting azimuth specified for the next group of radar replies to be transmitted, Interrogation Processing releases them for transmission. It may also start the radar interface if it is not already active.

---

\* Detailed format descriptions of the data structures are given in Appendix E, although details of the use of these structures are described only under the various software functions. Detailed descriptions of the I/O formats and protocols for the ARIES devices are given in Volume 3. A more detailed description of hardware operation is given in Section 2.0.

Fig. 3.1-1. ARIES software system.

The Track File is the central store of information concerning all aircraft visible to the sensor. Most other files are merely indices for this file. In particular, the DABS ID Lookup Table allows a program to quickly locate the track record for a target given only the DABS ID. The Azimuth Sorted Index allows a program to find all track records for targets that lie between two given azimuths. The Reply Time Sorted Index allows Interrogation Processing to find all track records for targets which could possibly reply to an all-call interrogation at the current antenna azimuth. Furthermore, these are sorted by reply time so that no further sorting needs to be done to place the replies in the reply buffer in correct time order.

Pre-Interrogation Processing has the function of creating the Reply Time Sorted Index for all aircraft which could be in the beam during the next 0.1 second interval. It also updates all track positions for such targets to the time at which they will next be on the antenna boresight. Finally, it generates simulated radar reports for these aircraft.

Data for the Track File are obtained from the traffic model data stored on the disk. These data are processed by Traffic Model Input. Each track update from the disk has a time associated with it, and the disk file is time ordered. Thus Traffic Model Input simply compares the update time of the next track update record with the current system time and processes the record at the specified time.

The Inter-ARIES communication functions\* consist of Adjacent ARIES Input, Modem Input and Modem Output. They provide the capability for several ARIES systems to run in synchronism to generate a multi-site environment. This can be used to load the various multisite network functions of the DABS system. Messages are provided which allow several sites to start synchronously and maintain synchronism. Other messages provide the means of keeping the state data for all targets consistent among all sites (e.g., if a target receives a lockout command from one sensor, the target should appear locked out at all sites).

The Timer Countdown function counts down the clocks in the Track File which simulate the transponder timeout function. This causes the simulated transponders to reinitialize their states if they have not been interrogated within about 16 seconds, as do real transponders.

Operator Communications provides the system operator with teletype commands by which system operation can be controlled. These include positioning the model to a specific time, starting the simulation, stopping the simulation, and checking system status.

---

\*To be implemented.

System Initialization controls the positioning of the model file and the initialization of all the system files. It can also be used to reinitialize the system after it has been halted by operator command.

The system includes the capability for the various functions to record data onto either disk or tape for the purpose of off-line analysis and debugging.

### 3.2 Interrogation Processing

#### 3.2.1 Assumed Interrogation Pattern

Fig. 3.2-1 is a time line for the DABS channel. As shown, the assumed pattern consists of single all-call interrogations separated by zero or more cycles of DABS interrogations. The all-call interrogations occur either at fixed intervals or follow a known, fixed pattern of intervals. The interrogation mode (A or C) is also assumed to follow a known pattern, as is the use of the front or back antenna in sensors equipped with a back-to-back antenna system. The combination of the mode pattern, the front/back pattern, and the interval pattern gives rise to an overall periodic interrogation pattern. Knowing this pattern, and the place of the last interrogation in the pattern, it is possible to determine the time of the next interrogation to within a few microseconds, and also the mode and front/back status of that interrogation. ARIES determines this interrogation pattern during its initialization procedures. At each all-call interrogation it then predicts the time and mode of the next all-call interrogation.

Each all-call interrogation is followed by a listening interval sufficient for receiving replies from the maximum sensor range, plus an extra interval to allow the channel to be clear of more distant replies.

The DABS interrogations which fall between the all-call listening intervals also are assumed to be structured. The arrival of these interrogations is not predictable. It is assumed, however, that if aircraft A receives an interrogation before aircraft B, that A's reply will be received by the sensor before B's reply. The discrete interrogations and replies are assumed to have a cyclic structure, with a series of interrogations followed by all the replies to those interrogations. No further interrogations are sent until all the replies are received. This is diagramed in Fig. 3.2-1. It may be that several of these cycles occur between all-call interrogations.

#### 3.2.2 Interrogation Processing Overview

Processing of discrete and all-call interrogations is handled differently due to the order of magnitude difference in the minimum round trip times as well as the fact that a single all-call interrogation can elicit replies from



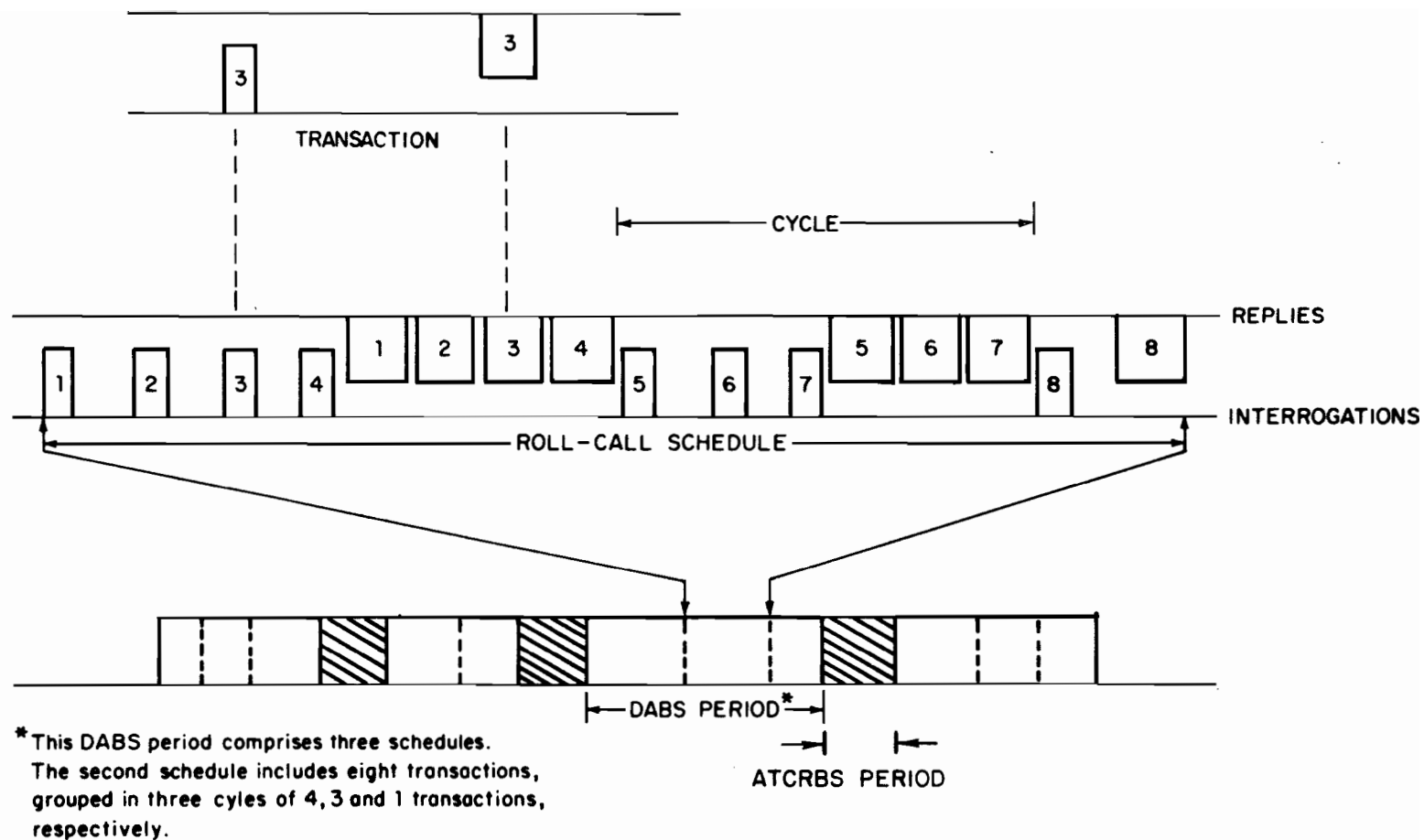


Fig. 3.2-1. Channel time allocation.

many aircraft while a single discrete interrogation causes only a single reply. It is possible to process discrete interrogations as they arrive. The total time available to the software in the worst case is 64  $\mu$ sec (see the analysis in Section 3.2.3.7). By efficient assembly language coding and use of a user-microprogrammable CPU the total processing time can be reduced to just under this limit.

For ATCRBS-equipped aircraft at the minimum range (1 nmi), the appropriate ATCRBS reply must be returned to the sensor within about 15  $\mu$ sec of the detection of the interrogation (12  $\mu$ sec for out-back travel time, 3  $\mu$ sec for transponder turnaround). This is barely sufficient time for a computer to respond to an interrupt, let alone perform any processing. Therefore, it is not possible to process all-call interrogations in the way discrete interrogations are handled, i.e., waiting until the interrogation is received to start generating the replies.

Fortunately, the nature of all-call scheduling is such that the sensor simply goes repeatedly through a short cycle of interrogations. Given a knowledge of this cycle, the position of the last interrogation in that cycle, and the time of that interrogation it is possible to predict the time and mode of the next interrogation. This cyclic pattern can be recognized very quickly by ARIES during an initialization period. It is then stored in the Interrogation Pattern File (one of the Site Characterization Tables in Fig. 3.1-1).

Once this pattern has been recognized and 'locked onto' it is possible to prepare the replies to an interrogation in the time interval preceding the interrogation. Furthermore, an external interval timer can be set to generate an interrupt to the computer several microseconds before the actual interrogation. This allows the computer time to change state and to enter the all-call processing programs in anticipation of the coming interrogation. Since the replies to that interrogation are already computed at that point, it can release these replies immediately to the reply generation hardware.

Thus, the processing cycle for all-call interrogations is inverted, in that the first thing done is to transmit the already computed replies back to the sensor. Following that, the replies for the next anticipated interrogation are calculated and the interval timer is set to interrupt slightly in anticipation of that interrogation.

Note that the success of this scheme does not depend on timing the interrogation intervals accurately to better than a few microseconds. The predicted time of the next interrogation is used solely to set the interval timer and to predict the antenna azimuth at the next interrogation (which does not require timing accuracy better than about 200  $\mu$ sec for a 4 second scan rate and 14 bit azimuth resolution.) The reply time data in the predicted replies is not referenced to this anticipated time, but to an interrogation time of zero. This can be done because the reply time clocks in the reply generators are always set to zero upon receipt of an all-call interrogation (see Section 2).

Various hardware features of the receiver, controlled reply generator, and interval timer are designed specifically to aid in this mode of operation and to eliminate certain timing problems that arise. See Section 2 of this manual.

Fig. 3.2-2 is a flowchart of Interrogation Processing. The following sections describe the various portions of that flowchart. Note that Interrogation Processing really consists of two separate interrupt handlers. One of these is triggered by the receiver interrupt and handles discrete interrogations and also the acquisition and verification of the all-call interrogation pattern. The other handler processes the interval timer interrupt and handles ATCRBS and all-call reply generation. These are described together because of the close relationship between them.

### 3.2.3 Discrete Interrogation Processing

#### 3.2.3.1 Inputs, Outputs, and Timing

##### Inputs:

Interrogations from the receiver input buffer

Track File records for the interrogated aircraft

DABS ID Lookup Table

Antenna Pattern Lookup Table (one of the Site Characterization Tables)

Azimuth Correction Table (one of the Site Characterization Tables)

##### Outputs:

Discrete replies to the reply buffer (a hardware buffer on the controlled reply generator interface)

Modified transponder state bits in the Track File records for the interrogated targets

Records of the interrogations to a data recording buffer. These include copies of the inter-ARIES track status messages.

##### Timing:

This procedure is triggered by the arrival of a discrete interrogation (hardware interrupt).

#### 3.2.3.2 Location of the Track File Record

The first step in processing a discrete interrogation is to locate the correct Track File record by using the DABS ID in the interrogation. The

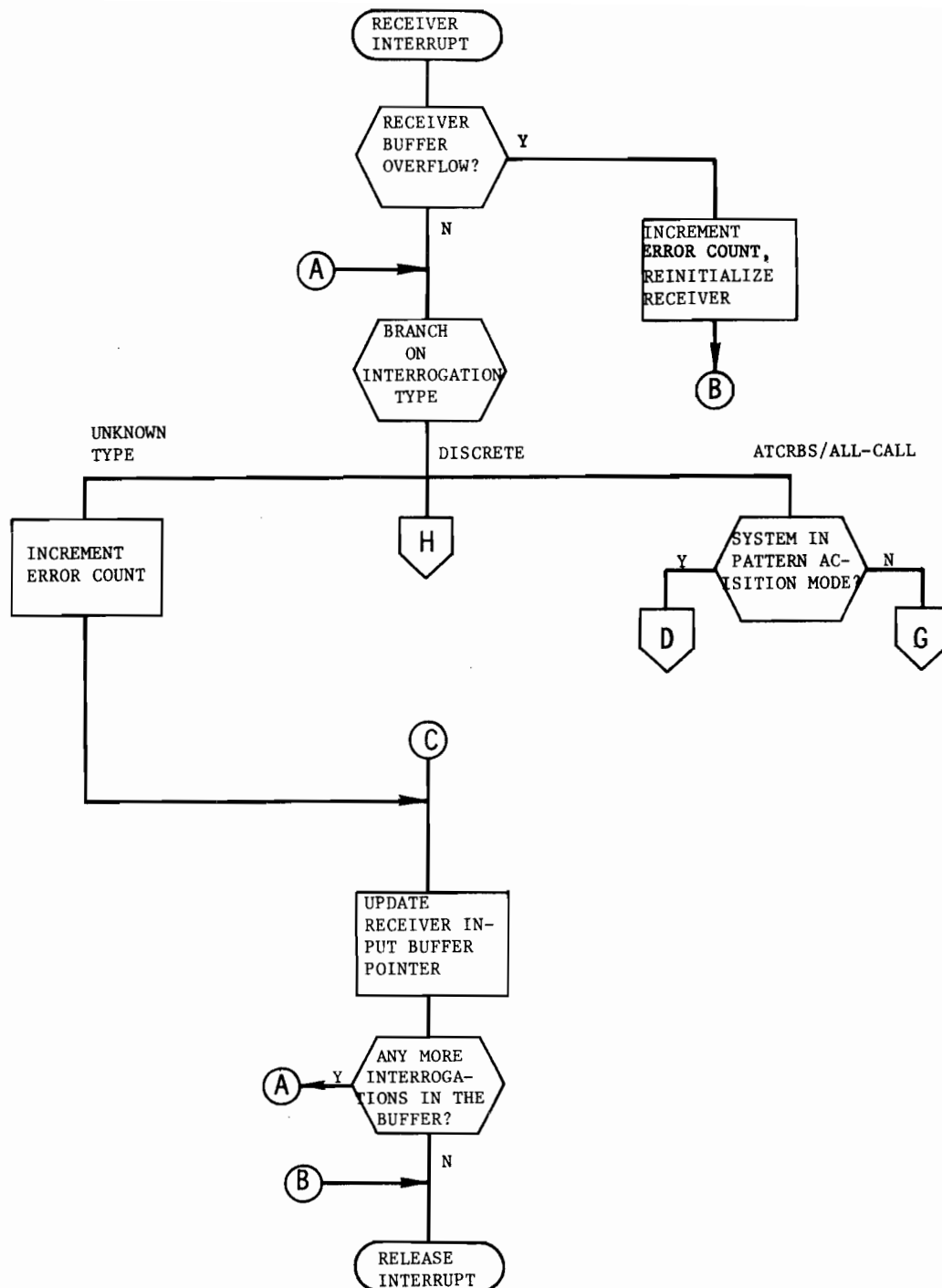


Fig.3.2-2. Interrogation processing flowchart.

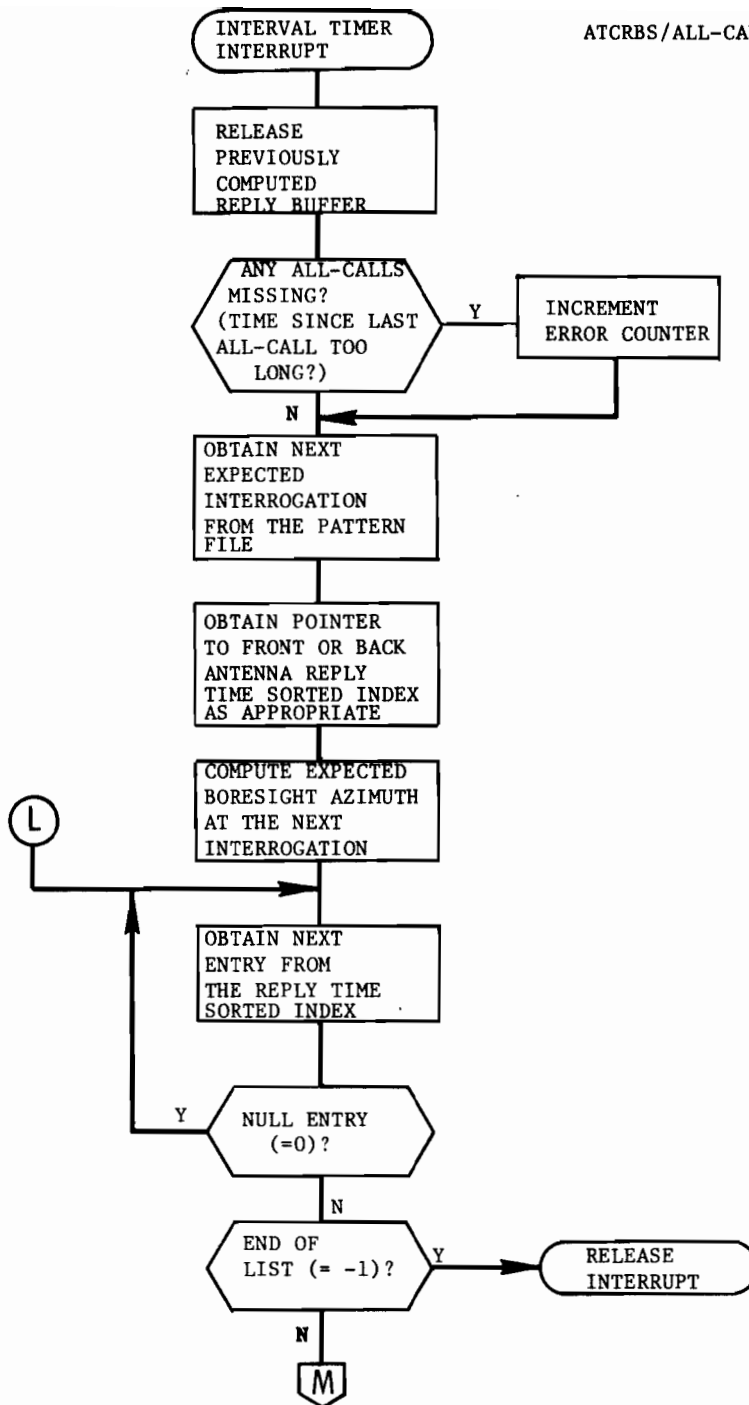


Fig.3.2-2. Continued.

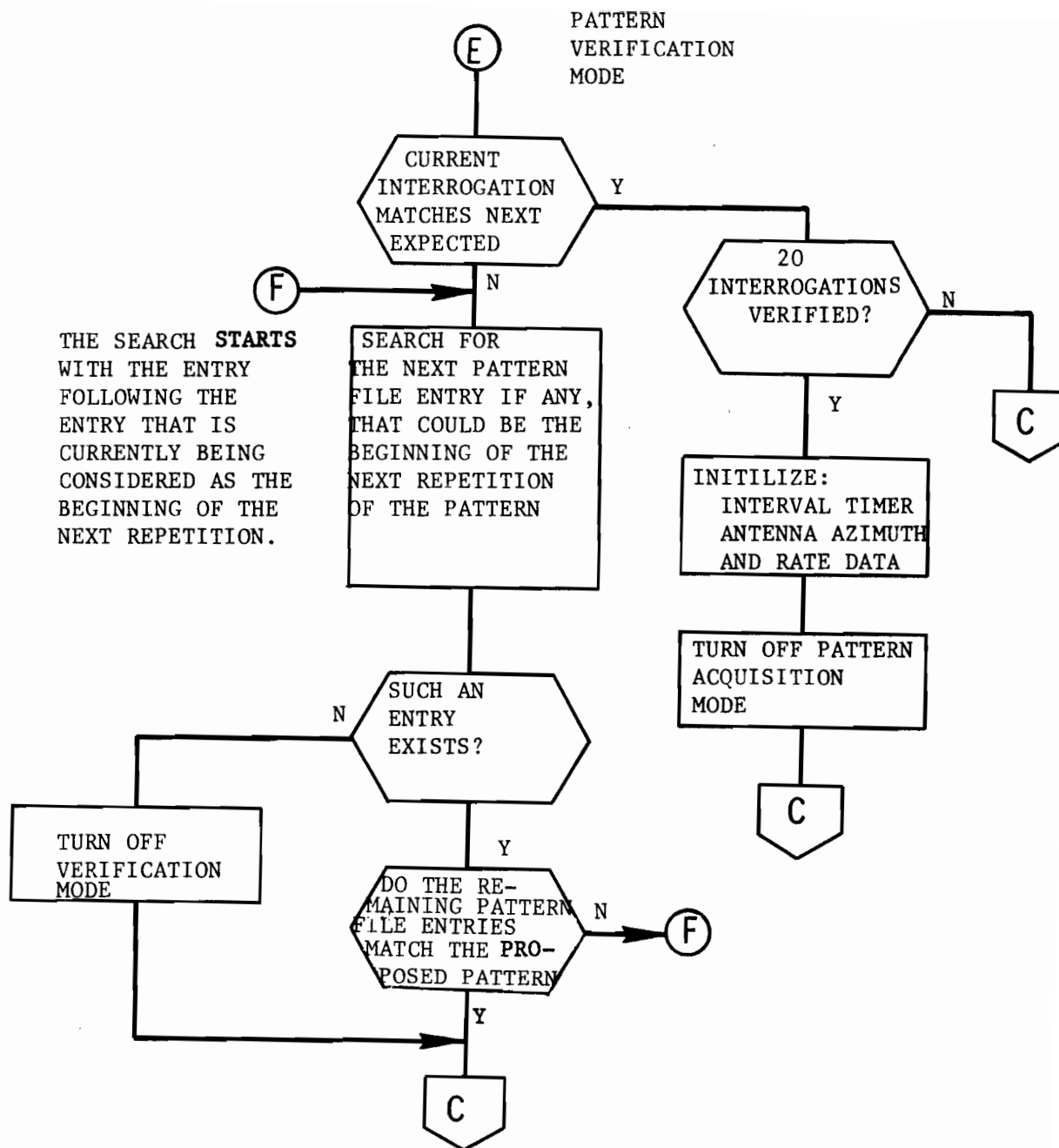


Fig.3.2-2. Continued.

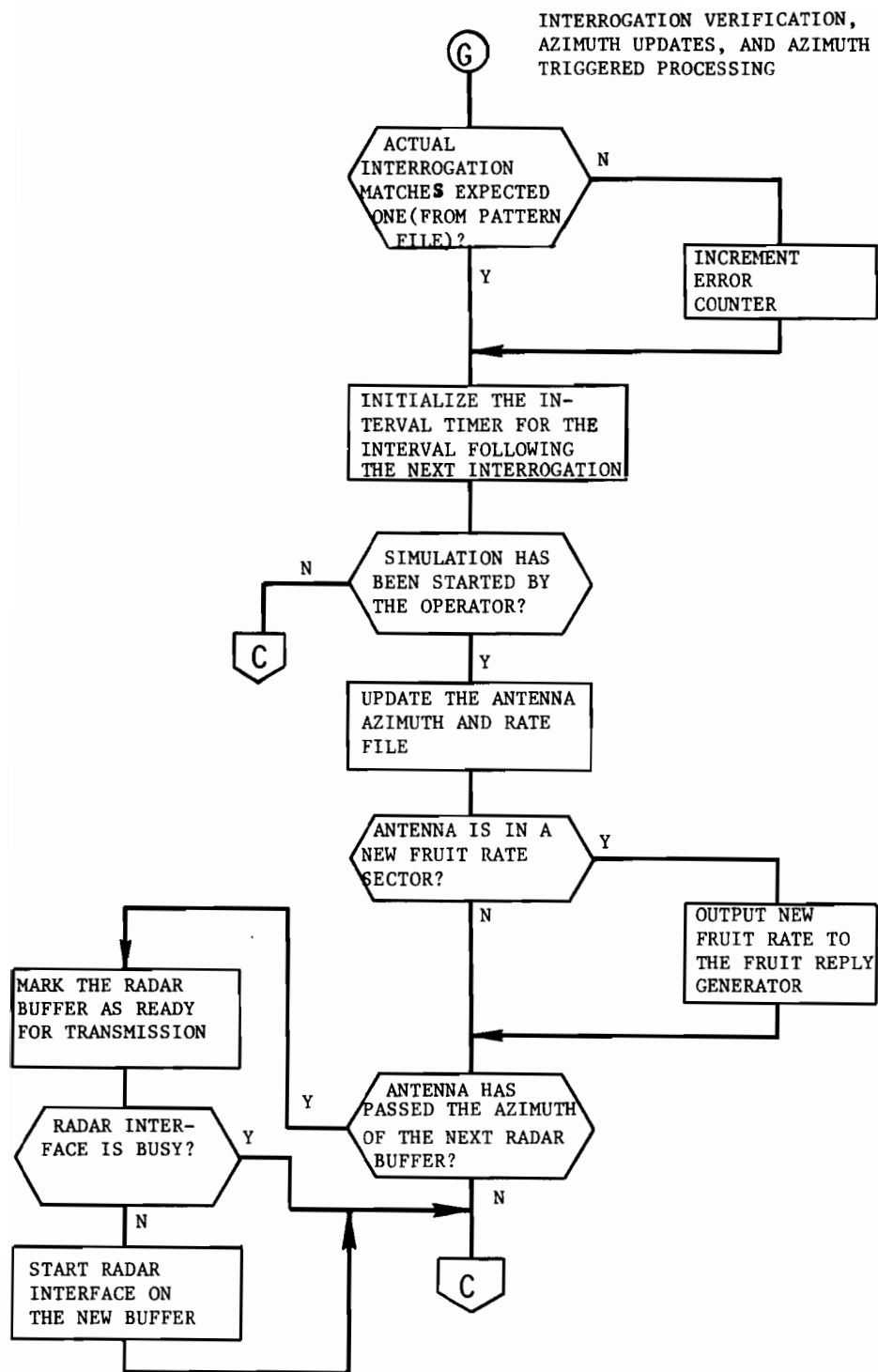


Fig. 3.2-2. Continued.

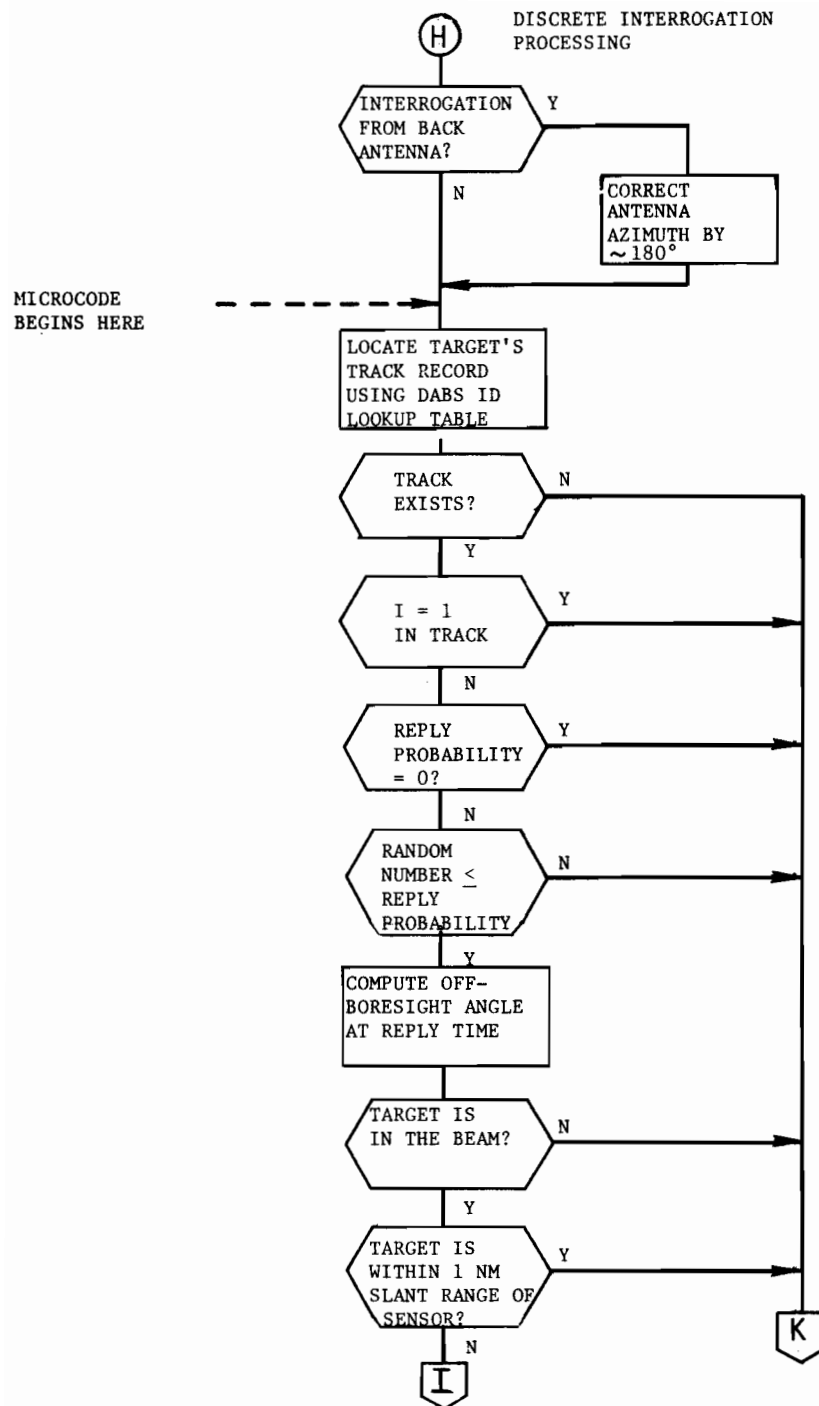


Fig.3.2-2. Continued.



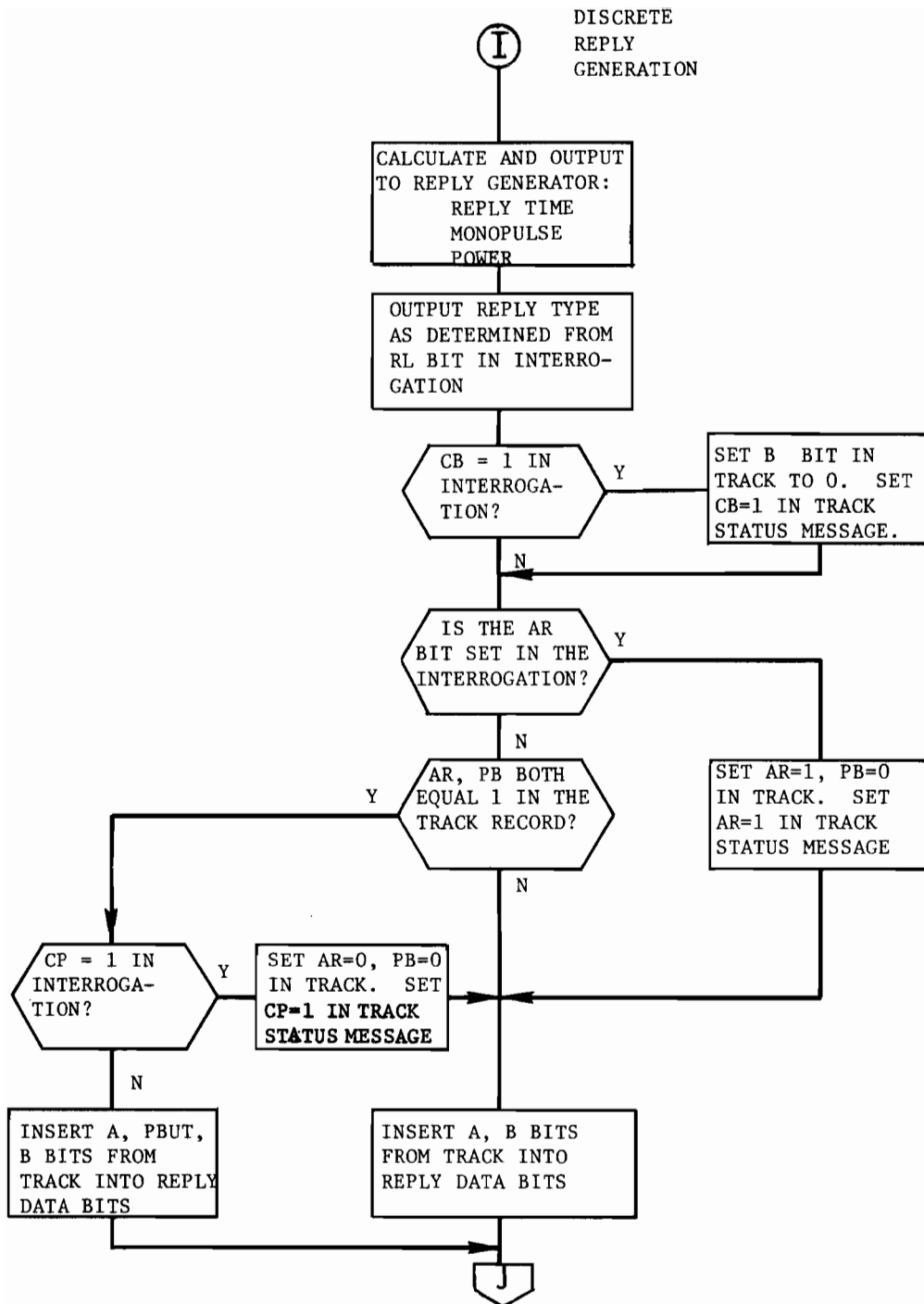


Fig.3.2-2. Continued.

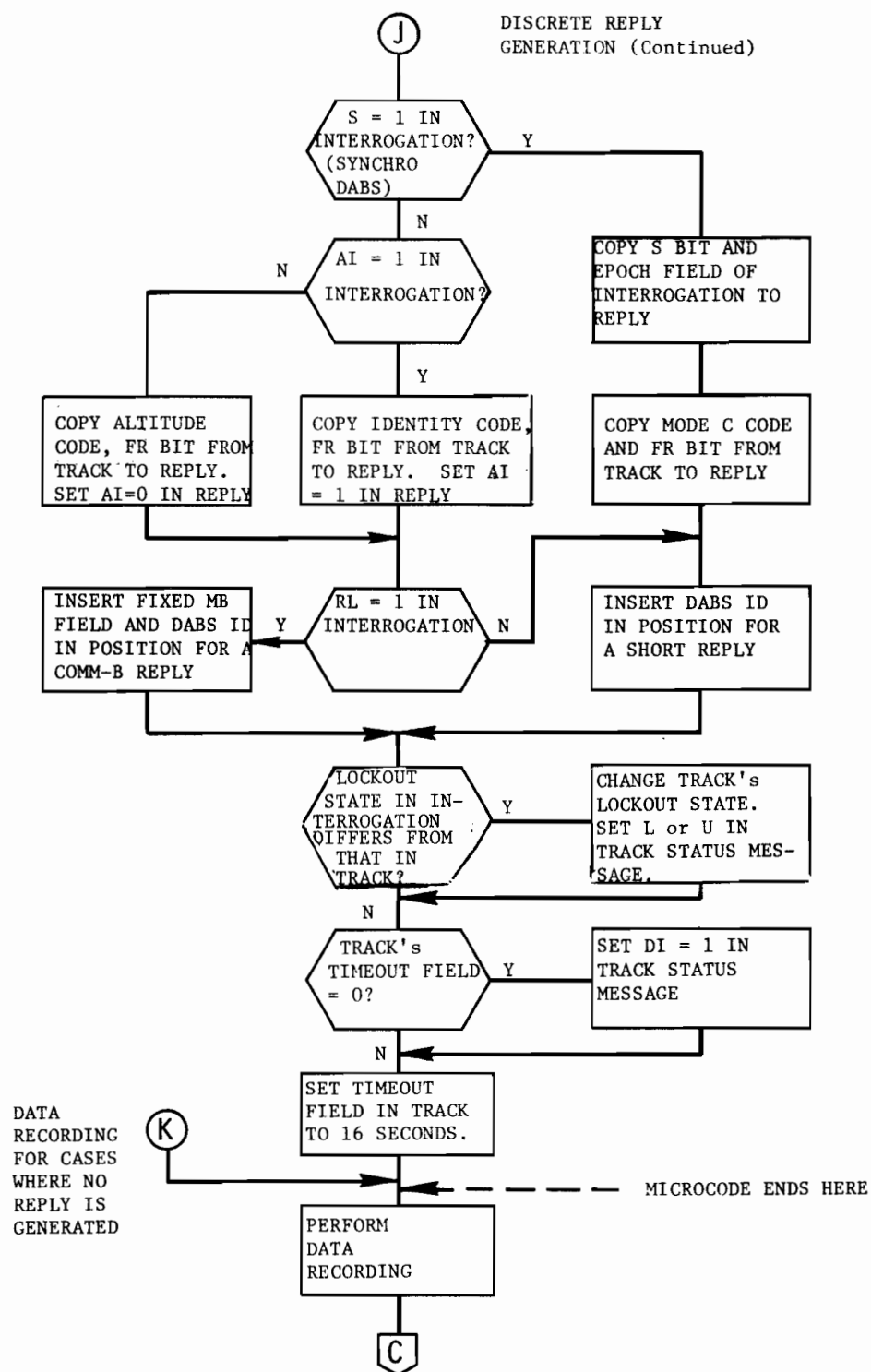


Fig.3.2-2. Continued.

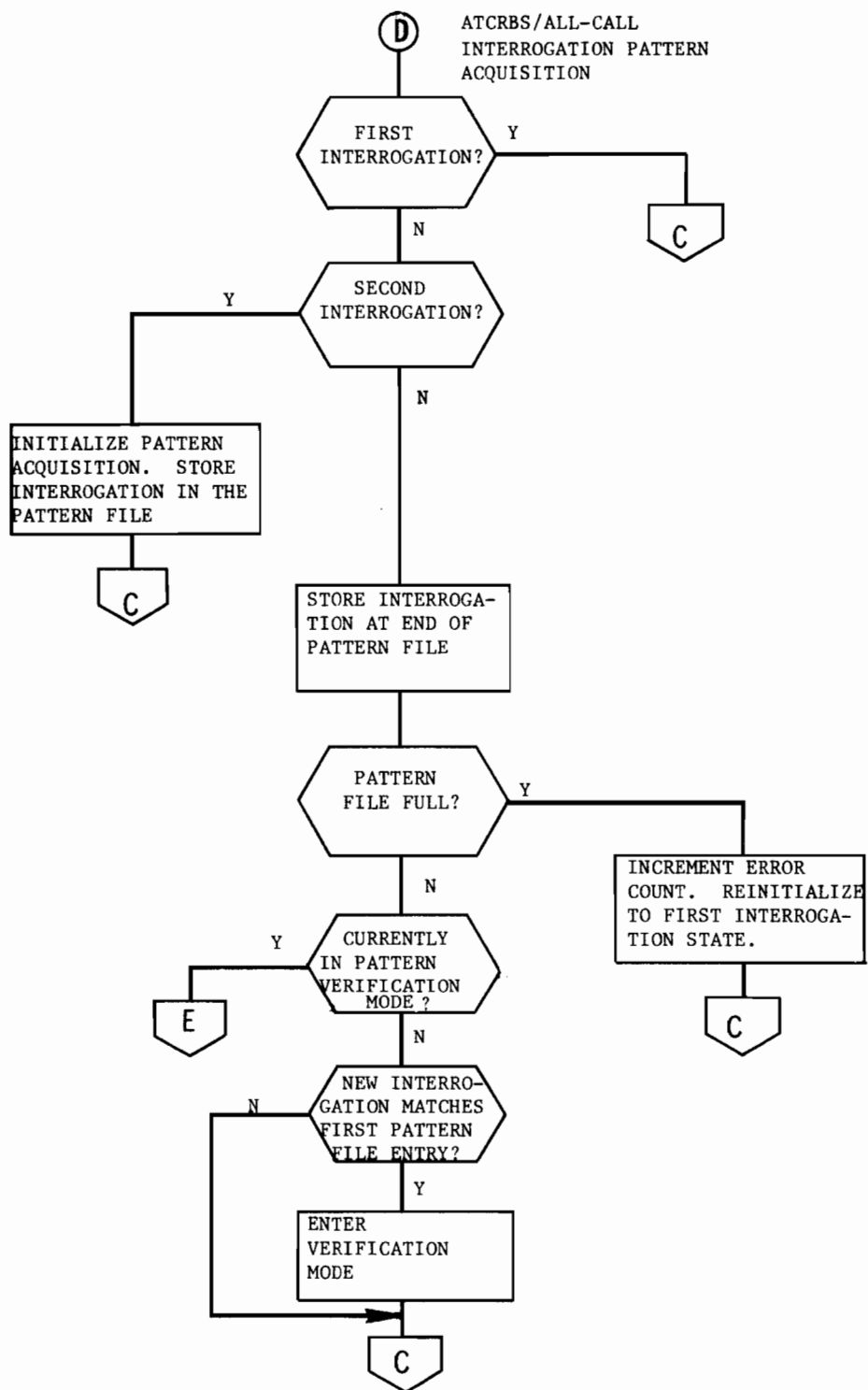


Fig.3.2-2. Continued.

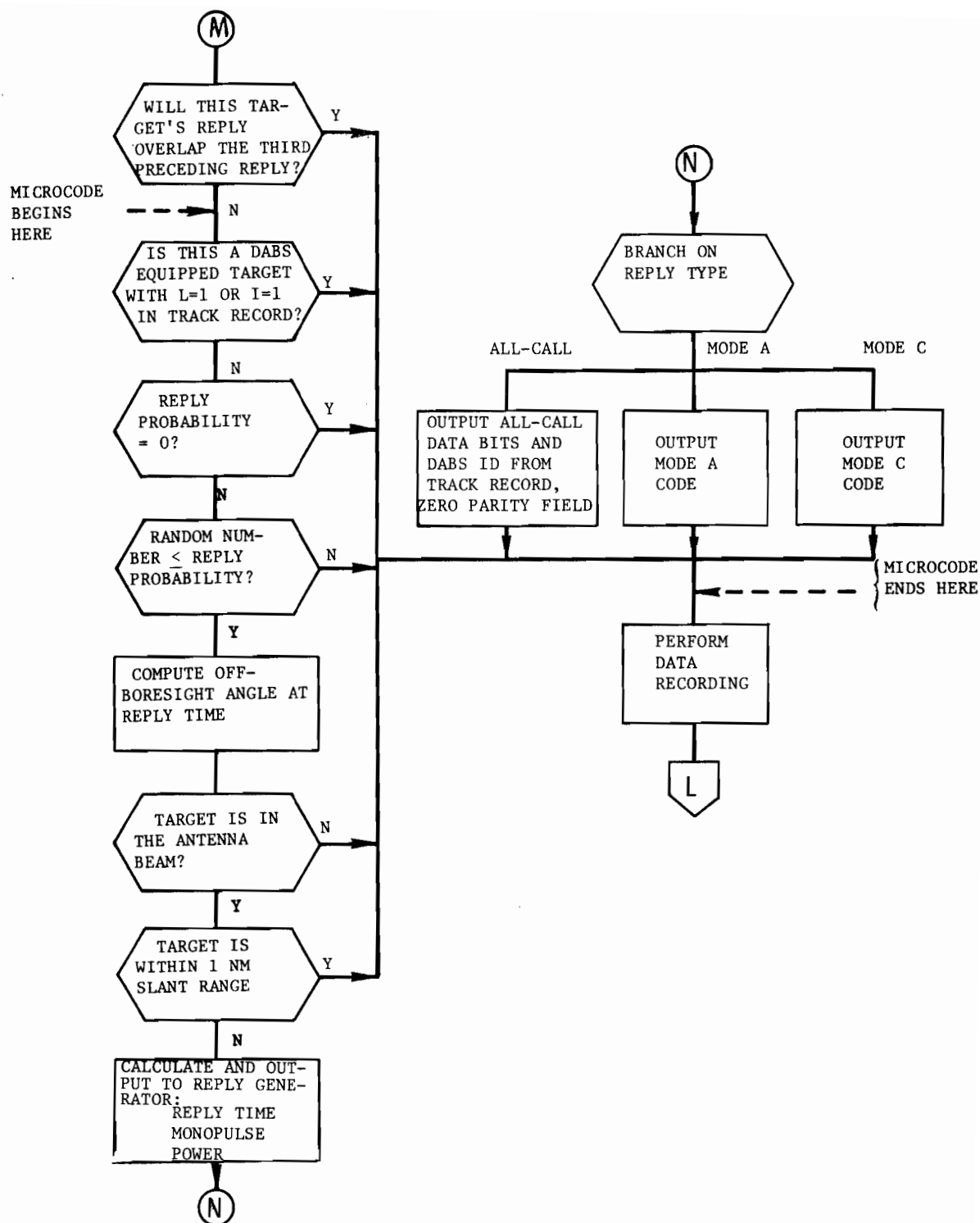


Fig.3.2-2. Continued.

low order ten bits in the ID are used as an index into the DABS ID Lookup Table. If the corresponding table entry is zero, the target does not exist. Otherwise, the table entry is a pointer to the first word of the track file record for that target.

Note that this procedure places certain restrictions on the DABS ID's provided by the model file. These ID's must be unique in the bit positions used for table lookup or the above procedure will not result in a unique track record for each aircraft. It is assumed that the model generation procedures will be free to assign ID's in order to satisfy this constraint. The bits of the ID used to index the DABS ID Lookup Table constitute what amounts to a track number for a target and are the only means of locating a track record. For this reason, ATCRBS targets are also assigned DABS ID's which are used solely as track numbers to uniquely identify all targets in the simulation.

#### 3.2.3.3 Determining Whether the Target is to Reply

The next step is to determine whether the interrogated target is to respond to the interrogation. In the real world, this is affected by several factors. The received power level determines whether or not the transponder can detect the interrogation. This is in turn determined by the position of the target within the sensor's antenna pattern, the position of the sensor in the target's antenna pattern, and the range of the target. A second effect is the possibility that interference on the uplink channel might prevent detection of the interrogation or cause certain bits to be altered in the message. The latter condition will result in an incorrect DABS ID being seen by the transponder due to the error detection coding on the uplink (see FAA-RD-75-61, FAA-RD-75-62) and no reply will be generated. Finally, if the target is far enough from boresight, the P2 pulse transmitted over the omnidirectional antenna will have a large enough amplitude compared with P1 and P3 to cause the transponder sidelobe suppression (SLS) logic to prevent a reply.

Even if the target does reply, the reply may be rejected by the sensor for several reasons. One is a low signal level, another is interference. Both of these conditions are simulated by the ARIES reply stream. However, if a target is so far off boresight that its difference beam antenna signal is greater than its sum beam antenna signal, replies will be rejected as being too far off-boresight by the Receive Sidelobe Suppression (RSLS) circuitry of the DABS sensor. ARIES cannot simulate this condition, as the design prevents the difference signal from being greater than the sum signal. Thus, the ARIES software must simulate this condition by preventing a reply.

It is not necessary for ARIES to simulate all of these effects in detail. The actual reply model used depends on only two parameters, a reply probability stored in the track record and the off-boresight azimuth of the target.

The reply probability encompasses three effects: the target's range, the target's antenna pattern, and the likelihood of interference. The traffic model determines the probability of the reply. The software which generates this model can use this probability to achieve various effects. Setting the probability to its maximum value guarantees that the target will reply if within the beam. Setting it to zero guarantees no reply. Alternating between these values on a scan to scan basis can be used to simulate any scan to scan reply pattern. Setting an intermediate value of reply probability will give a random pattern of replies. Under that circumstance, the given probability will be compared against the next output of the hardware random number generator. If this value is less than the given probability, a reply will be generated if the target is in the beam. If greater, no reply will be generated.

The off-boresight angle is used to model the effect of the sensor's antenna pattern. If the target is within some specified angle of the antenna boresight, the target will reply. The cutoff angle is the angle at which the sensor's difference beam gain equals the sum beam gain, this being the cutoff point beyond which ARIES cannot simulate the monopulse off-boresight signals.

The off-boresight angle of interest is the angle at the time of reply. This can be computer by:

$$\text{OBA} = \theta_T - \theta_A + c(\rho)$$

where:

OBA = off-boresight angle at the time of reply

$\theta_T$  = Target azimuth from the track record

$\theta_A$  = Antenna boresight azimuth at the time of interrogation, as received in the interrogation data block.

$c(\rho)$  = A correction factor which accounts for antenna rotation during the signal's round trip time. This is computed by using the most significant four bits of  $(\rho)$  to index the Azimuth Correction Table.

$(\rho)$  = The round trip time for the target, composed of the signal out-back time plus the transponder turnaround time.

Pre-Interrogation Processing updates  $\theta_T$  for all tracks to be the azimuth at the time the target is on boresight. For most targets, the azimuth rate will be small enough that they do not move during the beam dwell time. The

time critical nature of discrete interrogation processing makes it necessary to ignore any corrections due to target motion during the beam dwell. The linear position error resulting will not be more than 31 feet (assuming 64 msec beam dwell, 960 ft/sec target speed), but for short range targets this can result in a noticeable azimuth difference.

Note that these calculations must be done modulo 360 degrees. Note also that the antenna boresight azimuth in an interrogation is for the front antenna. For sensors with front and back antennas an offset must be added to this value to obtain the boresight of the back antenna whenever the interrogation is from the back antenna (as indicated by the front/back bit in the interrogation.) The offset is nominally 180 degrees, but may be slightly different due to mechanical alignment factors.

There is another reason for a target not to reply which is not related to any failure mechanism in real transponders. As discussed in section 3.4.3 the I bit of the track record may be set during the interval between the creation of a new track and the receipt of initialization information from other ARIES sites. During this time the target should not reply. Therefore, Interrogation Processing tests this bit for each reply to be generated (both discrete and all-call) by a DABS target.

#### 3.2.3.4 Reply Time, Power, and Monopulse Calculations

These are all the reply parameters, with the exception of the data content. The processing of uplink and downlink data bits are discussed in subsequent sections.

The reply time is calculated by adding the round trip time in the track record to the time of the interrogation as received in the interrogation data block. The round trip time ( $\rho$ ) is provided by the traffic model, and includes the transponder turnaround time as well as the out-back propagation time. The traffic model also provides ( $\hat{\rho}$ ), which is used by Pre-Interrogation Processing to predict ( $\rho$ ) to the time at which the target will be on the antenna boresight. Assuming a maximum target velocity of 960 ft/sec (568 kt), it is not possible for a target to move as much as one range unit (approximately 32 ft) during one-half the maximum assumed beam dwell time (64 msec.) Therefore, it is not necessary for Interrogation Processing to correct ( $\rho$ ) for the difference in time between boresight crossing time and the actual interrogation time.

The reply power is a function of the inherent transponder power, the range of the target, and the aircraft and sensor antenna patterns. The first three are modeled by a single track record parameter, the reply power. This is obtained from the traffic model, and is updated from the model periodically along with all the other target state variables obtained from the model. Since the power does not change significantly for small changes in range, Pre-Interrogation Processing will not adjust the power when it corrects the track's round trip time. It will be left to the traffic model generation software to update the track record whenever the power changes.

It is considered more important to correctly model the sensor's antenna pattern on the reply link than on the interrogation link. The reason for this is that part of the purpose of ARIES is to test the sensor's response to interference. The effect of interference on a given reply is determined in part by the signal to interference ratio. Therefore it becomes important to correctly simulate reply signal levels.

To calculate the sensor antenna's effect, the off-boresight angle at reply time is used to index the Antenna Pattern Lookup Table which gives the power correction to be added to the track record power (both are in logarithmic units.)

The same access to the Antenna Pattern Lookup Table which gives the correction to the power also gives the correct monopulse information to be inserted into the reply. This table is obtained by calibration procedures prior to running the environment simulation and matches the ARIES monopulse angle generation equipment to the sensor's monopulse angle processing circuitry such that the sensor will in fact measure the desired off-boresight angle.

### 3.2.3.5 Interrogation Data Bit Processing

Every discrete interrogation includes either 56 or 112 bits of information. Certain of these bits can affect the internal state of the transponder and its response to interrogations. Some of them can affect the data bits in the response to the interrogation in which they are received, and therefore must be processed before the reply bits are generated. Others only affect future interrogations and therefore can be processed after reply generation. This proves advantageous, as discrete interrogation processing is by far the most time critical function in the system. All of these state changes are discussed in this section.

Fig. D-1 (Appendix D) shows the data formats on interrogation and reply for all interrogations and replies. Figure E-1 shows how the interrogations appear in the receiver data blocks, and Figure E-2 shows the format of the reply data blocks sent to the controlled reply generator. ARIES simulated transponders will not reply to DABS-only all-call or to Comm-C interrogations, and as a result cannot generate Comm-D replies (see Appendix D for a definition of these formats.) All other modes can be handled, although on Comm-A the uplink message field is not used except for the Acknowledgement Request (AR) bit and on Comm-B replies a fixed MB field is returned by all simulated transponders.

The following interrogation data bits are ignored by ARIES:

IT: Interrogator type

For ARIES use this will always be 1 (standard interrogator) and is shown as such in Fig. E-1.

AL: ATRBS lockout

ARIES will never see ATRBS-only interrogations (i.e., without the P4 pulse which defines an all-call interrogation) and so need not process this bit.



MSRC: Source of MB field

Defines the source of the MB field to be included in the reply. Since ARIES will always reply with the same MB field, MSRC need not be processed.

SP: Spare data bits

These have no function and need not be processed.

SD: Special data

Normally contains the Altitude Acknowledge (ALEC) field for display to the pilot. It does not affect transponder state and may be ignored.

MA: Message bits

These are for distribution to external devices attached to the transponder and may be ignored by ARIES with the exception of the Acknowledgement Request (AR) bit. This bit affects the pilot acknowledgement logic, which is simulated by ARIES.

The S, AI, and RL bits affect only the reply data content and have no effect on transponder state. Therefore, they are discussed in the section on generation of reply data bits (3.2.3.6). The use of the parity/address field, also called the DABS ID, has been described in the section on locating Track File records (3.2.3.2).

CB, AR, CP, and the DL bits are discussed here. All of these affect the transponder internal state. Since a simulated transponder should have a consistent transponder state at all ARIES sites, state changes caused by these bits at one site must be transmitted to other sites. Therefore, part of the function of Interrogation Processing is to format an inter-ARIES track state message whenever a simulated transponder's internal state is changed. The format for such messages is given in Fig. E-10. Inter-ARIES protocols are discussed in greater detail in Section 3.5.

CB (Clear B) = 1 resets the B bit in the track record. (The track record format is shown in Figure E-6.) This function must be performed before a reply is generated. CB is set in an inter-ARIES track status message. The B-bit indicates that the transponder wishes to initiate an air-ground data transfer, and is requesting channel time from the sensor. The CB bit indicates that the sensor has successfully received the message and that the transponder can terminate its request.

If AR=1 in an interrogation, the AR field in the track record should be set, and PB should be cleared. This must be done before a reply is generated, as these bits affect the PBUT field of the reply. Pre-Interrogation Processing will set PB the next time it processes this track. When both AR and PB are

set, Interrogation Processing will include the pilot acknowledgement (PBUT) in the reply. The reason for the AR->PB->PBUT sequence is to approximate the time delay that occurs in the transponder between the time an AR is received and the time that the pilot acknowledgment buttons are enabled. Receipt of AR always causes an inter-ARIES track status message to be sent with AR=1.

CP (Clear PBUT) = 1 resets the AR and PB fields in the track record if both are set, and thereby inhibits further transmission of PBUT to the sensor. This function must be performed before a reply is generated. If AR and PB were set before receipt of this bit, CP is set in an inter-ARIES track status message. AR takes precedence over CP in that if AR=1 is received in the same interrogation the processing for CP should not be performed. Thus, AR and CP cannot both be set in an inter-ARIES message.

DL affects the transponder's response to all-call interrogations. Values of 01 or 11 in this field prevent ("lock out") all replies to all-call interrogations until such time as DL=00 is received. DL=10 does not change the lock-out state. Initially the transponder is in an unlocked state, and therefore will reply to all-calls. The only processing required for this field is to set the L-bit in the Track File record for the target if DL=01 or 11, and reset it if DL=00. This may be done after a reply has been generated. A change to 1 causes the L bit to be set in an inter-ARIES Track Status Message, and a change to 0 causes the U bit to be set in this message.

The Timer field in the track record is related to all of the above functions except the B-CB pair. The Timer field is set to its initial value (nominally 16 seconds) whenever any discrete interrogation is received. It is counted down by Timer Countdown. If it ever reaches zero, Timer Countdown simulates the transponder's 'loss of contact' function by resetting the L, AR, and PB fields in the track record, and informing the other ARIES system(s) via a track status message with the T bit set. As part of the timeout protocol, if a discrete interrogation is ever received when the timer field's value is 0, an inter-ARIES message with the DI bit set must be sent to inform adjacent ARIES that this sensor has started interrogating the target. See Section 3.5 for further details.

#### 3.2.3.6 Reply Data Generation

The following is a list of all the reply fields and the values they are to receive under various conditions. If the value for a field is not specified, it should be set as shown in Fig. E-2, which is the reply block format for the controlled reply generator.

- A: Copied from the A bit in the track record. (Fig. E-6)
- S: Copied from the S bit in the interrogation.  
If S=1, the Epoch field is also copied from the interrogation.
- AI: Copied from the AI bit in the interrogation.

PBUT: If the AR and PB bits in the track record are set, the PBUT field from the track record is copied to the PBUT field in the reply; otherwise this field is set to zero.

B: Copied from the B bit in the track record.

FR: Copied from the FR bit in the track record.

Altitude/Identity: If S=1, the mode C code from the track file is inserted;  
If S=0, then:

AI=0 causes the Mode C code to be inserted, and  
AI=1 causes the Mode A code to be inserted.

MB: If RL=1, this is obtained from a fixed format downlink message.

DABS Address: The DABS ID is obtained either from the track record or from the interrogation, whichever is more convenient.

Note that S=1 (a synchro-DABS reply) has an effect upon several other values, and AI and RL need not be checked. A synchronous reply is always a short reply with Mode C code.

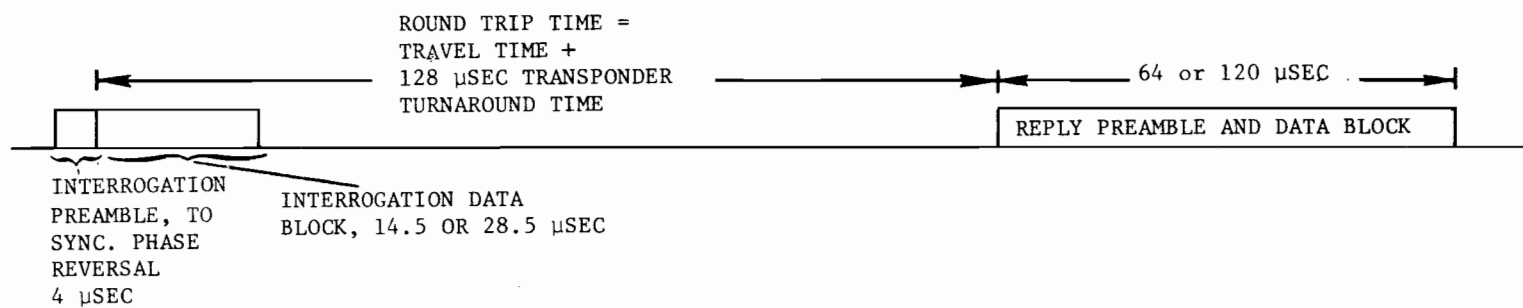
The addition of the reply data bits to the reply time, power, and mono-pulse information completes the reply generation. Interrogation Processing enters the reply in an output buffer in the ARIES digital hardware on a word by word basis as each reply word is generated.

### 3.2.3.7 Timing Considerations for Discrete Interrogations

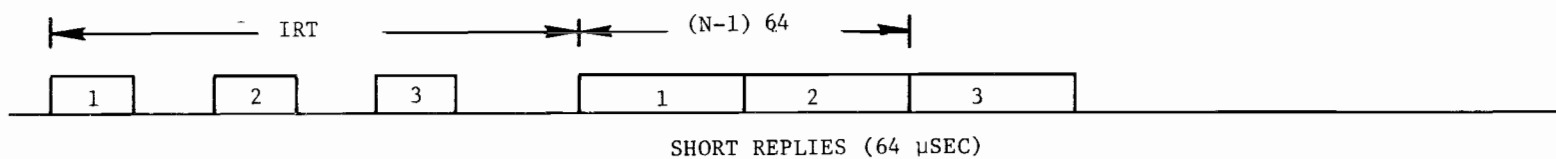
The processing of discrete interrogations is the most time critical task in the ARIES system. This section derives the worst case interrogation pattern for this task. It is found that an interrogation cycle consisting of interrogations to two targets at the minimum range constitutes the worst case. For this case, an average of about 64  $\mu$ sec per interrogation is available to the software.

DABS discrete interrogations are transmitted in cycles, as shown in Fig. 3.2-1. The timing involved in an individual interrogation/reply pair is shown in Fig. 3.2-3. For purposes of this analysis, long interrogations and short replies are the worst case, as these give rise to the shortest time between the receipt of the last bit of the first interrogation in a cycle and the beginning of the last reply. This is the total interval available for processing the interrogations in the cycle.

The total interval from the detection of the first interrogation in the cycle (upon receipt of the sync phase reversal) to the leading edge of the



A) TIMING FOR A SINGLE INTERROGATION/REPLY PAIR



B) DERIVATION OF WORST CASE TIMING

Fig.3.2-3. Derivation of worst case timing for processing discrete interrogations.

first pulse of the last reply is:

$$\text{IRT} + (n-1)64 \text{ } \mu\text{sec}$$

where IRT is the round trip time of the first interrogation/reply pair (including transponder turnaround time), n is the number of interrogations, and 64  $\mu\text{sec}$  is the length of a short reply.

The DABS scheduler generates a schedule with back-to-back replies, as shown. Aircraft are scheduled in decreasing range order, and so it is guaranteed that if replies are scheduled back-to-back the interrogations will not overlap and will appear in the same order as the corresponding replies. Scheduling for a cycle stops when an attempt is made to schedule an interrogation that would overlap any of the replies.

Given the available time calculated above, the average time per interrogation is

$$\frac{\text{IRT} + (n-1)64}{n} = \frac{\text{IRT} - 64}{n} + 64$$

The worst cases arises when n is a maximum, i.e., when the cycle is densely packed. The densest packing occurs when all targets are at the same range, in which case the interrogations have the same spacing as the replies, 64  $\mu\text{sec}$ . In that case,  $n = \lfloor \text{IRT}/64 \rfloor$  where  $\lfloor \quad \rfloor$  indicates the greatest integer less than the given quotient. Ignoring the integer quantization:

$$\frac{\text{IRT} - 64}{n} + 64 = \frac{\text{IRT} - 64}{\text{IRT}/64} + 64 = 128 - \frac{(64)^2}{\text{IRT}} = \text{Avg. time/interrogation}$$

Thus, the average time is a minimum when IRT is smallest, i.e., for targets at the minimum range. For a minimum range ARIES target (1 nmi), IRT is about 140  $\mu\text{sec}$  (128 for transponder turnaround plus 12 for out-back time). A maximum of two discrete interrogations is possible in this interval. Average available time is then:

$$\frac{140 + 64}{2} = \frac{204}{2} = 102 \text{ } \mu\text{sec}$$

However, for a long interrogation, the first 30  $\mu\text{sec}$  are required for receipt of all the data for the first interrogation (some of this time is required in any case to allow the CPU to handle the interrupt and save its state). Also, approximately 15  $\mu\text{sec}$  are required to allow time for the last reply generated by the software to be transmitted to the hardware reply generators. Thus, only 159 of the total 204  $\mu\text{sec}$  are available to the software (the interrogation time for the second interrogation and the lead time for the first reply have no effect on this value).

This available time must then be further reduced to account for the fact that data channel I/O operations steal time from the Data General Eclipse CPU used in ARIES. There are three sources of data channel activity: the receiver, the PCD radar interface, and the disk. The receiver will use channel time to input the second interrogation of the pair, using 8  $\mu$ sec of CPU time. The radar interface will output at most one word during the 204  $\mu$ sec interval, using 1.6  $\mu$ sec of CPU time. Finally, the disk will use at most 20  $\mu$ sec if transferring data at its maximum rate during the entire interval. The total time used by the data channel could therefore be as large as about 30  $\mu$ sec, leaving 129  $\mu$ sec for the software, or an average of 64.5  $\mu$ sec per reply.

This is obviously a very short period of time in which to perform all the processing described above. The only way ARIES is able to do this is by taking advantage of the user-microprogrammable feature of the Eclipse CPU.

### 3.2.4 ATCRBS/All-Call Reply Generation

#### 3.2.4.1 Inputs, Outputs, and Timing

Inputs: Reply Time Sorted Index  
Track File records for targets in the Reply Time Sorted Index  
Azimuth Correction Table (one of the Site Characterization Tables)  
Antenna Pattern Lookup Table (one of the Site Characterization Tables)  
Interrogation Pattern File (one of the Site Characterization Tables)  
Antenna Azimuth and Rate Data

Outputs: An ATCRBS/All-Call Reply Buffer for the next interrogation.

Timing: This procedure is triggered by a hardware interrupt from the interval timer.

#### 3.2.4.2 Processing

The all-call portion of Interrogation Processing is entered upon receipt of the interval timer interrupt. The first action is to release to the reply generation hardware the previously computed reply buffer.

The next function is to obtain from the Interrogation Pattern File (one of the Site Characterization Tables, acquired during system initialization, see Section 3.2.5.2):

- The time to the next interrogation
- The mode (A or C) of the interrogation
- Whether the front or back antenna will be used (for sensors with back-to-back antennas)

Also, the Antenna Azimuth and Rate Data is sampled before it is updated by ATCRBS/All-Call Interrogation Processing (section 3.2.5).

The interrogation time is used to calculate the antenna boresight azimuth at the time of the next interrogation, using the Antenna Azimuth and Rate Data. The front/back indication is used to select the appropriate Reply Time Sorted Index. This index contains a pointer to the Track File record for each target that could conceivably respond to an ATCRBS/All-Call interrogation from the specified antenna at that time. Furthermore, these are ordered by increasing reply time starting with the first target to reply. All-Call Reply Generation consists largely of accessing successive targets on this list, preparing the appropriate replies for those targets which will actually reply, and inserting these replies (still reply time ordered) into a reply buffer. Note that, as described in Section 3.4.5, an entry in the Reply Time Sorted Index may be zero due to a track being dropped after it was entered in the index.

It is possible, with all-call and ATCRBS replies, to have several targets reply at the same or nearly the same time. It is not desirable to have more replies in progress at one time than there are reply generators. Therefore, the software must check each reply time and compare it against the reply time of the third preceding reply (there are three reply generators). If the third preceding reply would still be in progress at the time the current reply would be starting, the current reply should be discarded to prevent overloading the reply generation capability. If these two replies do not overlap, the current reply can be added to the reply buffer.

For each target, it is next decided if it is a DABS target which is locked out to all-call. A DABS target is locked out whenever the L bit is set in the track record. If a target is locked out, the program can proceed to process the next target.

Next, it is determined whether the target is to reply or not, based on the reply probability and whether it is in the antenna beam. This processing is identical to that described in Section 3.2.3.3.

The reply time, power, and monopulse values for a reply are determined as in Section 3.2.3.4, the only difference being that the interrogation time is taken to be zero. This can be done due to the hardware feature which causes all ARIES interrogation and reply time counters to be set to zero upon arrival of an ATCRBS/All-Call interrogation.

Finally, the data bits are determined by the anticipated interrogation mode and whether the target is DABS or ATCRBS equipped. For ATCRBS targets, either the Mode A or Mode C code is inserted in the reply, corresponding to the anticipated interrogation mode. For DABS targets, the all-call data bits from the Track File (the two words containing the capability and DABS ID fields) are combined with a zeroed address/parity field.

This completes the all-call processing for a given target, and the program then returns to the Reply Time Sorted Index to obtain the next track record to be processed.

### 3.2.5 ATCRBS/All-Call Interrogation Processing

#### 3.2.5.1 Inputs, Outputs, and Timing

**Inputs:** Receiver data blocks for ATCRBS/All-Call interrogations.  
Interrogation Pattern File (one of the Site Characterization Tables).  
Antenna Azimuth and Rate File.  
Fruit Rate Table (one of the Site Characterization Tables).  
Radar Report Buffer.  
Current system time (from the System Status File).

**Outputs:** Interrogation Pattern File (at system initialization)  
Updated Antenna Azimuth and Rate File  
New interval timer count.  
New fruit rates to fruit reply generator.  
Initializing commands for the radar report interface.

**Timing:** Triggered by the interrogation interrupt from the receiver. Control is sent to this procedure only for ATCRBS/All-Call interrogations.

#### 3.2.5.2 Acquiring the All-Call Interrogation Pattern

While this is strictly part of ARIES initialization, it is based on the processing of interrogation inputs from the hardware and is therefore programmed with Interrogation Processing. System Initialization merely sets a flag for Interrogation Processing which alters the way all-call interrogations are processed.

When an interrogation arrives, a test is made to see if the system is in the pattern acquisition mode. If so, the processing that is performed attempts to find a cyclic pattern of interrogations followed by the sensor. It is assumed that a single cycle contains no more than 20 distinct interrogations, where an interrogation is characterized by the interval between it and the previous interrogation, its mode (A or C), and the antenna used (front or back). The basic range clock resolution of the receiver for ATCRBS/All-Call interrogations is 1  $\mu$ sec, but for Interrogation Pattern File purposes this is reduced by a factor of 4, so that the least significant bit of the inter-arrival times in that file represents 4  $\mu$ sec ticks.

The cycle location algorithm starts by simply ignoring the first interrogation received, as the initial interval may be incorrect. The time interval, mode, and front/back status of the second interrogation processed are stored as the first entry of the Interrogation Pattern File. Note that the interrogation time in the receiver data block is the time since the last ATCRBS/All-Call interrogation, as the interrogation time counter is set to zero at each such interrogation. As successive interrogations are received, they are first



compared against this initial entry to see if the cycle might be starting again. The mode and front/back status must be identical and the time interval must be within one 4  $\mu$ sec count of the parameters of the initial entry.

If the new interrogation does not match, it is entered at the next location in the table.

If the new interrogation does match, the algorithm goes into a verification mode where, in addition to adding the interrogations to the end of the table, it compares successive interrogations against the current entries in the Interrogation Pattern File. If it can successfully match the file against the next 20 interrogations it assumes that it has discovered the cycle. If any match fails, the algorithm has retained all interrogations that were received before finding the mismatch. It then attempts to find a new cycle of at least the length of the unsuccessful cycle, using the interrogations already received as a basis. If the available table space overflows, an error condition is signaled, and the algorithm reinitializes itself to try again.

Once the pattern has been acquired, this program starts the interval timer, thus initializing the ATCRBS/All-Call Reply Generation process described above. After that point, ATCRBS/All-Call interrogations are processed only for purposes of re-initializing the interval timer, verifying the interrogation pattern and providing various azimuth dependent services, described below.

#### 3.2.5.3 ATCRBS/All-Call Pattern Verification

At each ATCRBS/All-Call interrogation, the interval, mode, and front/back indicators for the interrogation received are compared against those expected. The interval is acceptable if it is within one 4  $\mu$ sec count of the expected value. If the actual interrogation does not match the expected one, an error counter is incremented.

#### 3.2.5.4 Initializing the Interval Timer

At each interrogation, the interval timer is preset with the interval expected between the next interrogation and the following interrogation, less 60  $\mu$ sec. The 60  $\mu$ sec constitutes the amount of "early warning" time the ATCRBS/All-Call Reply Generation software will get before the latter interrogation. This preset value does not affect the current count. Instead, an interval timer feature which causes the preset value to be loaded as the current count upon receiving an ATCRBS/All-Call interrogation is used. Thus, at the next interrogation the appropriate interval is loaded to cause an interrupt just prior to the following interrogation. By triggering the load on the arrival of the next interrogation more precise timing is maintained than would be the case if the software directly loaded the count.

#### 3.2.5.5 Update to the Antenna Azimuth and Rate File

This function is not logically related to All-Call Interrogation Processing, but it is convenient to include it in that algorithm due to the availability of a new antenna azimuth measurement in the receiver data block. All that is involved is to calculate a new antenna rate by the formula:

$$\text{New Rate} = \frac{\text{Interrogation Azimuth} - \text{Azimuth File Azimuth}}{\text{Interrogation Time} - \text{Azimuth File Time}}$$

That is, the antenna rate is computed from the last two sample points with no smoothing performed. The new rate thus computed is stored along with the new interrogation azimuth and interrogation time to complete the update.

#### 3.2.5.6 Update of the Fruit Rate

This is performed here for similar reasons to the previous update. The new antenna azimuth is checked against the current sector index to the Fruit Rate Table (one of the Site Characterization Tables). If the antenna has moved into a new sector (as determined from the high order 5 azimuth bits) then the new sector number is used as an index into the Fruit Rate Table, the specified rate value is retrieved, and is output to the fruit reply generation hardware.

#### 3.2.5.7 Transmission of Radar Data

If the antenna has moved past the starting azimuth for the next radar reply buffer, then the PCD radar interface is started to transmit that buffer, unless it is still busy with previous buffers. In the latter case, the buffer is marked as ready for transmission, and will be started by the interrupt handler for the radar interface when the previous buffers are completed.

### 3.3 Pre-interrogation Processing

#### 3.3.1 Purpose

Pre-interrogation Processing locates all those tracks in the Track File which can be interrogated by either the front or back antenna of the sensor during the next 0.1 second. It prepares these for interrogation by updating their positions to the time at which they will cross the antenna boresight. It then enters a pointer to each such track into one or both of two new copies of the Reply Time Sorted Index, one each for the front and back antennas. Tracks are entered into the index corresponding to the antenna that will interrogate them, and the pointers are then sorted by reply time. The two new Reply Time Sorted Indexes then replace the current copies of these tables for use by Interrogation Processing.

Pre-interrogation Processing also generates simulated radar reports for each target. These reports are in a format which is easily transformed to Production Common Digitizer (PCD) format by the radar report interface. They are grouped in radar reply buffers, each buffer corresponding to one azimuth bin of the Azimuth Sorted Index.

### 3.3.2 Inputs, Outputs and Timing

Inputs: Azimuth Sorted Index  
Track File  
Antenna Azimuth and Rate Data  
Current system time (from the System Status File)  
Radar blip/scan ratio (associated with the Radar Report Buffers)

Outputs: Updated Track File records  
Modified Azimuth Sorted Index  
Reply Time Sorted Indices  
Radar Report Buffers

Timing: Performed every 0.1 second, and  
initiated by the real time clock  
interrupt handler

A functional flowchart for Pre-interrogation Processing appears in Fig. 3.3-1. The data structures are described in Appendix E. The rest of this section discusses some of the details of this task.

### 3.3.3 Locating Tracks to be Processed

The task must first determine where the antenna will be during the next 0.1 second. The Antenna Azimuth and Rate Data plus the system real time (millisecond) clock provide sufficient information for this purpose.

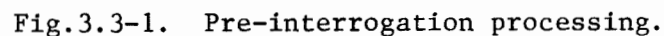
The following equation determines the boresight azimuth at the start of the interval:

$$\theta_c = \theta_A + w_A * (t_s - t_m)$$

where:

$\theta_c$  = current antenna azimuth  
 $\theta_A$  = antenna azimuth, from the Antenna Azimuth and Rate Data  
 $w_A$  = antenna rate, from the Antenna Azimuth and Rate Data  
 $t_s$  = time at start of 0.1 second interval = current system clock value  
 $t_m$  = time of measurement of the antenna azimuth from the Antenna Azimuth and Rate Data

Substituting  $t_s + 0.1$  second for  $t_s$  gives the antenna boresight at the end of the interval.



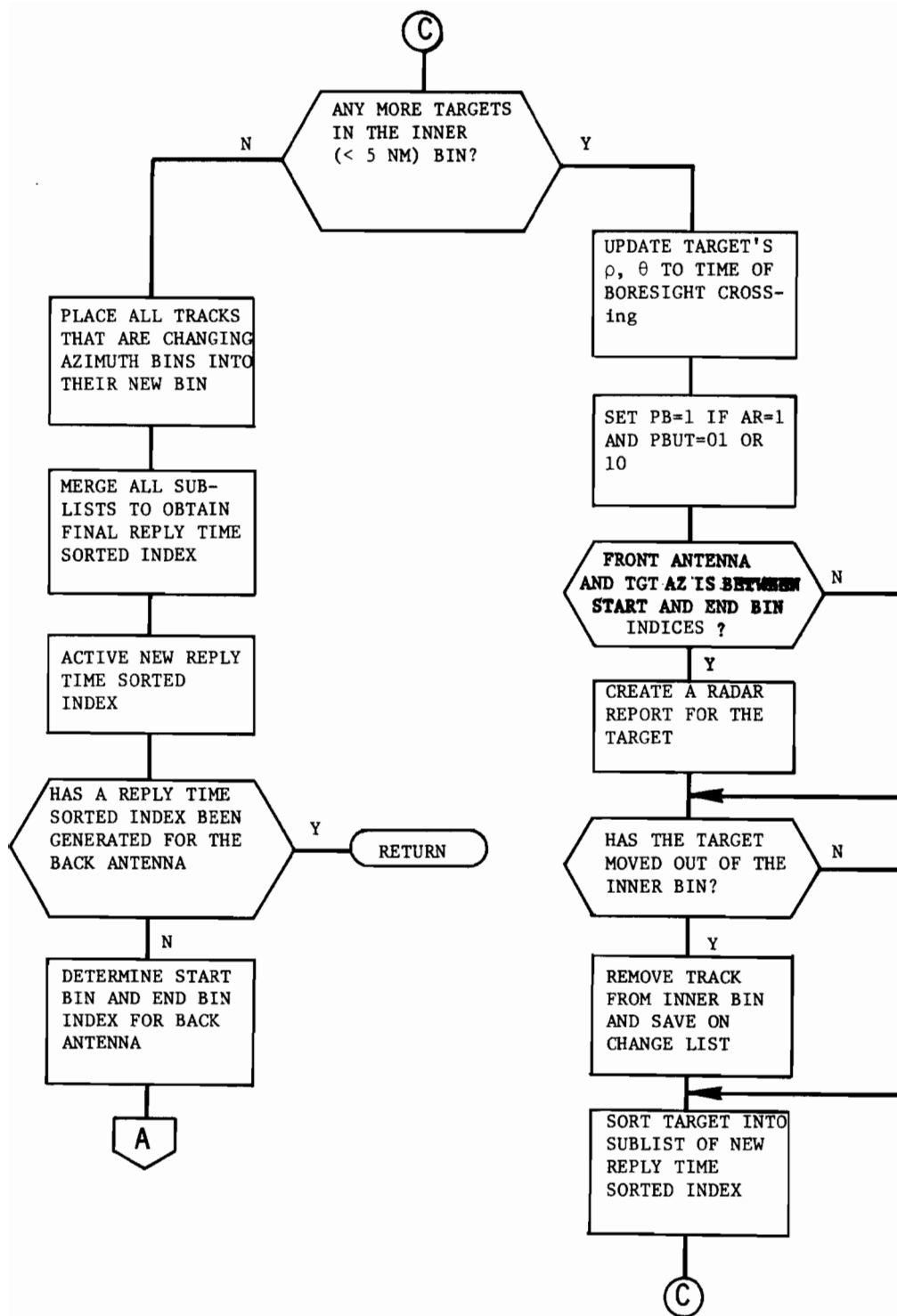


Fig.3.3-1. Continued.

The cutoff angle (identical to that used by Interrogation Processing, Section 3.2.3.3) must be subtracted from the current azimuth so that all targets within the beam at the start of the interval are included. Similarly, this quantity must be added to the final azimuth, for a similar reason. Thus all azimuths between the trailing beam edge at the current time, and the leading beam edge 0.1 second in the future are considered. A further factor is added to the leading edge of the beam to provide a small safety factor (1 azimuth bin).

The high order 8 bits of the resulting azimuths are then used as indices to the Azimuth Sorted Index. The purpose of this index is to make it easy to locate all tracks which lie at particular azimuths. It divides the coverage area into 256 azimuth bins, each of which contains a linked list of all tracks in the corresponding azimuth sector. There is also a special bin for close range targets (those within 5 nmi) which may change azimuth bins so rapidly that it is difficult to maintain them in the correct list.

Pre-interrogation Processing will prepare all tracks in the bins lying between the starting azimuth bin and the final azimuth bin, as calculated above, plus those tracks in the short range bin. All azimuth bins that lie within this interrogation area are marked to indicate this by setting the Start and End Bin indices of the Azimuth Sorted Index. This is used to signal the Traffic Model Input task to carefully update tracks in those azimuth regions, as this could destroy the track preparation performed by Pre-interrogation Processing. When it sets these indices for one set of azimuth bins, it automatically removes the previous set of bins from the 'in-the-beam' region, thus releasing those tracks. Further discussion is contained in the section on Traffic Model Input, Section 3.4.

Due to the buffer zone provided at the leading edge of the beam, some bins will be processed more than once. By use of the previous value of the End Bin Index in the Azimuth Sorted Index, double updating of most tracks is avoided and only bins newly entering the beam have track updates performed. However, all tracks in the beam are entered in the Reply Time Sorted Index.

Note that all the processing described here is repeated for the back antenna, except that radar report generation is only performed for the front antenna.

#### 3.3.4 Update to the Track File

The tracks' range, azimuth, and time must be updated to the time of boresight crossing. Interrupts are disabled during the time new values are stored to prevent interaction with Interrogation Processing. An exact expression for the time of boresight crossing given the antenna and target positions at time  $t$  is:

$$\Delta t = \frac{\theta_T - \theta_A}{w_A - w_T}$$

$$t_B = t + (\Delta t)$$

where:

$t_B$  = time of boresight crossing  
 $\theta_T$  = target azimuth,  $w_T$  = target azimuth rate  
 $\theta_A$  = antenna azimuth,  $w_A$  = antenna azimuth rate

Unfortunately, the track and antenna state information will almost never be for the same instant of time, and so the track must first be updated to the time of the antenna measurement. Then  $(\Delta t)$  is calculated, and finally the range and azimuth are updated. The full set of equations is:

- 1)  $t_1 = t_A - t_T$
- 2)  $\theta = \theta_T + w_T * t_1$
- 3)  $t_2 = (\theta - \theta_A) / (w_A - w_T) + t_1$
- 4)  $\theta_T \leftarrow \theta_T + w_T * t_2$
- 5)  $\rho_T \leftarrow \rho_T + \dot{\rho}_T * t_2$
- 6)  $t_T \leftarrow t_T + t_2$  (time of antenna boresight crossing)

where:

$t_A$  = time of antenna state data  
 $t_T$  = time of track state data  
 $\theta_T, w_T$  = the track's azimuth and azimuth rate, from the Track File  
 $\theta_A, w_A$  = the antenna's azimuth and rate from the Antenna Azimuth and Rate Data  
 $\rho_T$  = round trip time from the Track File  
 $\dot{\rho}_T$  = rate of change in round trip time from the Track File  
 $\leftarrow$  represents assignment to the Track File record

If this update causes the track to move to a new bin in the Azimuth Sorted Index, Pre-interrogation Processing will remove the track from its current bin and save it in a list of similar targets. Targets on this list will be placed in their new bins after all bins are processed. The move to the new bins must be delayed to prevent the possibility of some targets being processed twice.

In addition to this position update, Pre-Interrogation Processing sets the PB bit in the track if the AR bit is already set and PBUT = 01 or 10. This supports the simulation of the pilot acknowledgment process as described in Section 3.5.4.

### 3.3.5 Preparing the Reply Time Sorted Index

As Pre-interrogation Processing updates each track, it will store a pointer to that track record in a new copy of the appropriate Reply Time Sorted Index. These pointers will be grouped in sublists of 5 entries each, preceded by a count of the number of entries in the sublist (the last sublist may have fewer than five). As each new pointer is added, it is sorted into its sublist based on the reply time of the track it points to, with shortest reply times appearing at the beginning of each sublist. The format appears to the left of Fig. 3.3-2.

The final reply time sorted list is produced by successive merging of these sublists. The first step in this merge operation is also shown in Fig. 3.3-2. Note that a count of -1 is used to mark the end of the sublists.

The end result of all these operations is a new Reply Time Sorted Index for either the front or back antenna. The first word of this list will be a count of the number of tracks in the list, and the rest of the words will be reply time sorted pointers to the tracks which might be interrogated. The last pointer is followed by a -1 word. The count and -1 word are redundant information.

After such a list is created, it must be passed to Interrogation Processing. This must be done carefully, as Interrogation Processing is interrupt driven, and therefore may look at this list at any time. Pre-interrogation Processing simply overwrites a pointer to the old table with a pointer to the new table. This involves a single uninterruptable store operation and is therefore safe (i.e., Interrogation Processing can never access the data in an 'in-between' state). Whenever Interrogation Processing is running, it has priority over all other system tasks, and therefore Pre-interrogation Processing can never change the pointer while Interrogation Processing is using the Reply Time Sorted Index. Thus, the entire operation is safe, and Interrogation Processing will always receive one unique Reply Time Sorted Index each time it is run.

### 3.3.6 Radar Report Generation

As shown in Appendix E, Figure E-11, radar reports are placed in a circular buffer area. Reports are grouped by azimuth bin, and all targets which appeared in a given azimuth bin appear in the corresponding group of radar reports, even though Pre-interrogation Processing may have predicted them to a different bin in the course of updating the track. Each group of reports is preceded by two



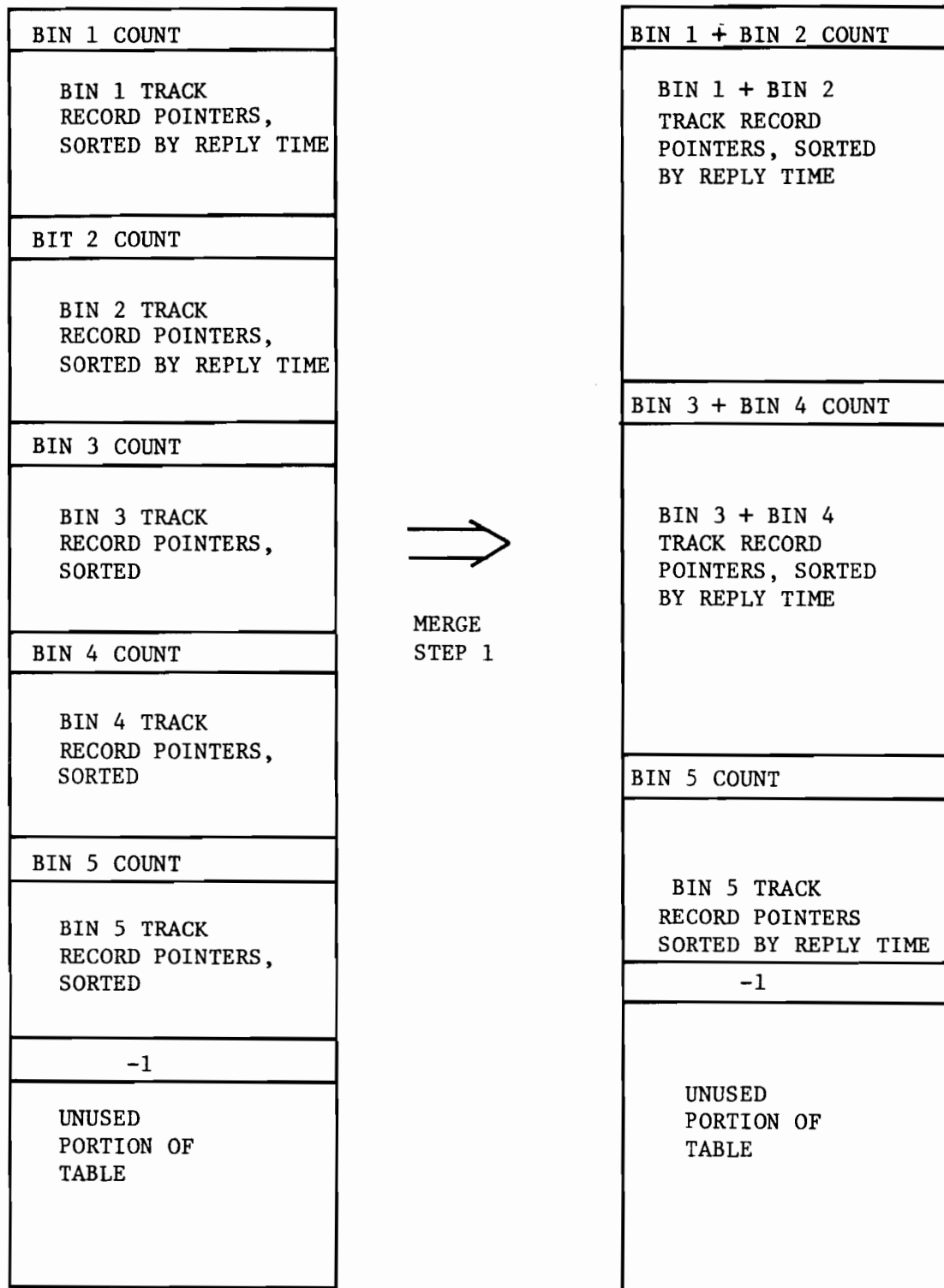


Fig.3.3-2. Merging reply time sorted lists.

words, the first of which specifies the antenna azimuth after which a group of reports is to be transmitted, and the second of which specifies the number of reports in the group. The azimuth word is set to the initial azimuth of the azimuth bin from which the reports were derived plus 164 azimuth units (one azimuth unit is  $2\pi/2^{14}$  radians). This assures that targets are out of the radar beam at the time their reports are sent.

Radar report generation is straightforward, given the updated round trip time and azimuth. Round trip time must be converted to one way range in the appropriate units by subtracting the transponder turnaround time (3  $\mu$ sec for ATCRBS, 128  $\mu$ sec for DABS), dividing by two, and converting units. Azimuth conversion requires only rounding off to a 12 bit azimuth. Time in storage is set to a constant 0.25 seconds.

To make the radar simulation more realistic, there is a radar blip/scan ratio. A 16 bit integer is used to represent a system-wide blip/scan ratio. For each radar report, a random number is obtained from the hardware random number generator, and the report will actually be generated only if the random number is less than or equal to the constant. If the blip/scan ratio is zero, no reports are generated.

For the azimuth bin at the North point, two additional fixed messages are placed at the head of the buffer. These are a radar status report (the message indicates that the status is satisfactory) and a search Real Time Quality Control (RTQC) report. The latter is a report for a fixed known target. Both of these are checked by the receiving ATC system for purposes of radar failure detection. Formats for these messages are shown in Appendix E, Figure E-12.

Note that radar reports are to be generated only for the front antenna.

The creation and transmission of radar reports is controlled by three indices into the radar report buffer. The first of these points to the next free location in the report buffer at which a radar report group will be created. It is moved whenever one group is completed and a new group begun. The second follows the first around the circular buffer, never being allowed to pass it. It points to the next group to be released for transmission. It is moved by ATCRBS/All-Call Interrogation Processing whenever the antenna passes the transmission azimuth specified for the report group. If the radar report interface is not busy with an earlier report group at that time, it is started on the newly released group. The third points to the next group to be transmitted (or the group currently being transmitted). It follows the second index around the circular buffer, never passing it. It is moved by the interrupt handler for the radar report interface whenever an interrupt is received indicating complete transmission of a radar report group. If it does not equal the second index after being moved, report transmission is started for the next report group.

### 3.4 Traffic Model Input

#### 3.4.1 Purpose, Input/Output, and Timing

This procedure has the function of updating track records in the Track File from the traffic model on the disk. Tracks are updated at times specified in the model records which cause the updates. In addition to updating the Track File, Traffic Model Input may update the track's position in the Azimuth Sorted Index, add tracks to, or delete them from that index, add tracks to or delete them from the DABS ID Lookup Table, and delete tracks from the Reply Time Sorted Index.

Inputs:   Disk input buffers  
          System Status File  
          Antenna Azimuth and Rate Data  
          Start and End Bin indices associated with the  
              Azimuth Sorted Index  
          DABS ID Lookup Table  
          Track File

Outputs:   Updated Track File  
          Updated Azimuth Sorted Index  
          Updated DABS ID Lookup Table  
          Deletion of dropped tracks from Reply Time Sorted Index  
          Updated System Status File (at system initialization only)

Timing:   Triggered by the real time clock interrupt handler whenever  
          the current system time becomes equal to the time of the next  
          model record.

A functional flowchart for this task is presented in Fig. 3.4-1. The data structures are described in Appendix E.

#### 3.4.2 Input Processing

The ARIES model data comes from a file on the system disk. The file logical record format appears in Appendix E. These logical records are grouped into physical records of 256 words for storage on the disk. There are three disk buffers, each one large enough to hold two physical records. At any time, one of these is being processed by Traffic Model Input, and the others are available to Disk Input. The latter task attempts to keep these buffers full. Buffers are used in round-robin fashion. After emptying a buffer, Traffic Model Input must inform Disk Input that a buffer is to be filled. It meanwhile starts processing the next buffer in rotation.

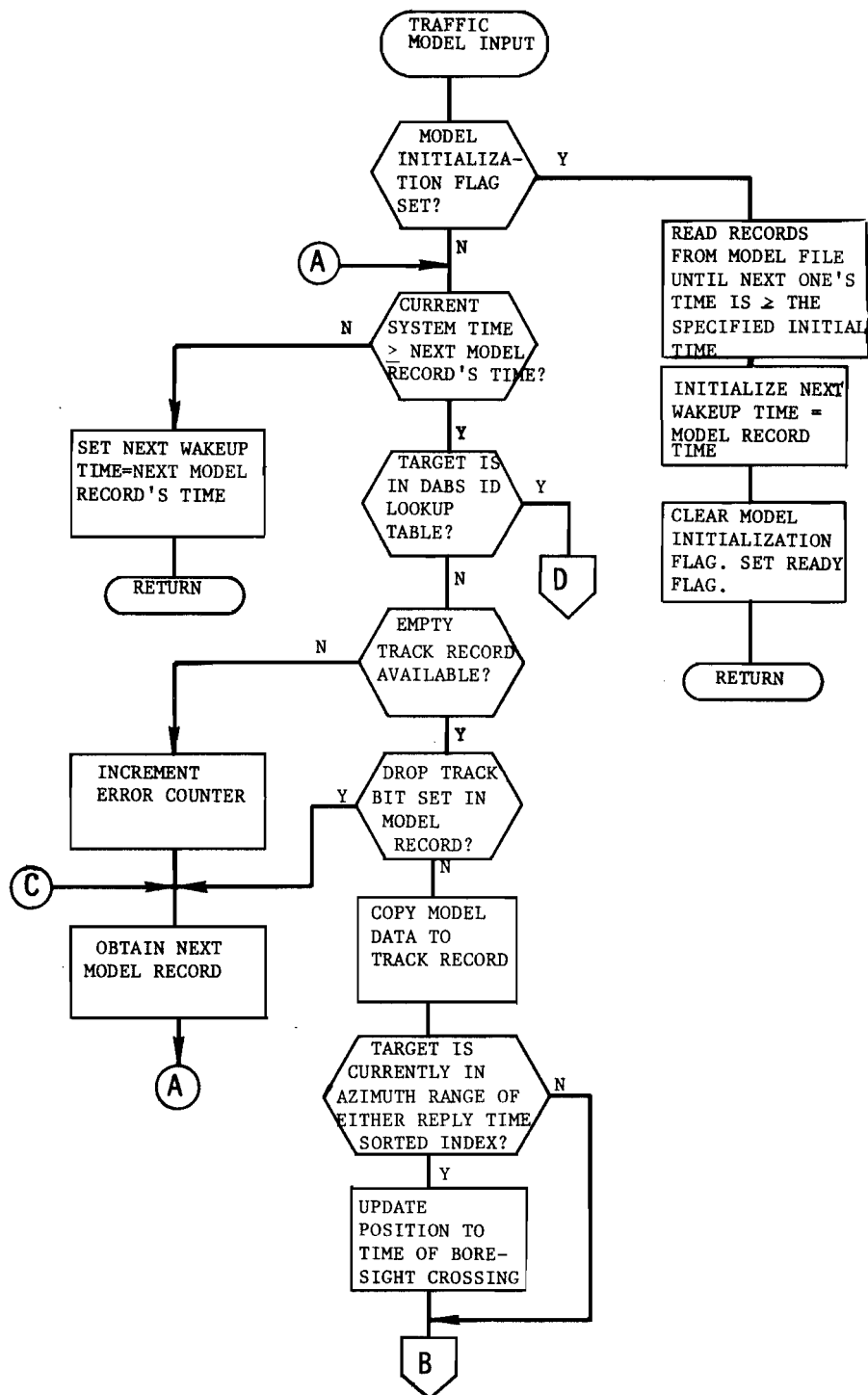


Fig.3.4-1. Traffic model input.

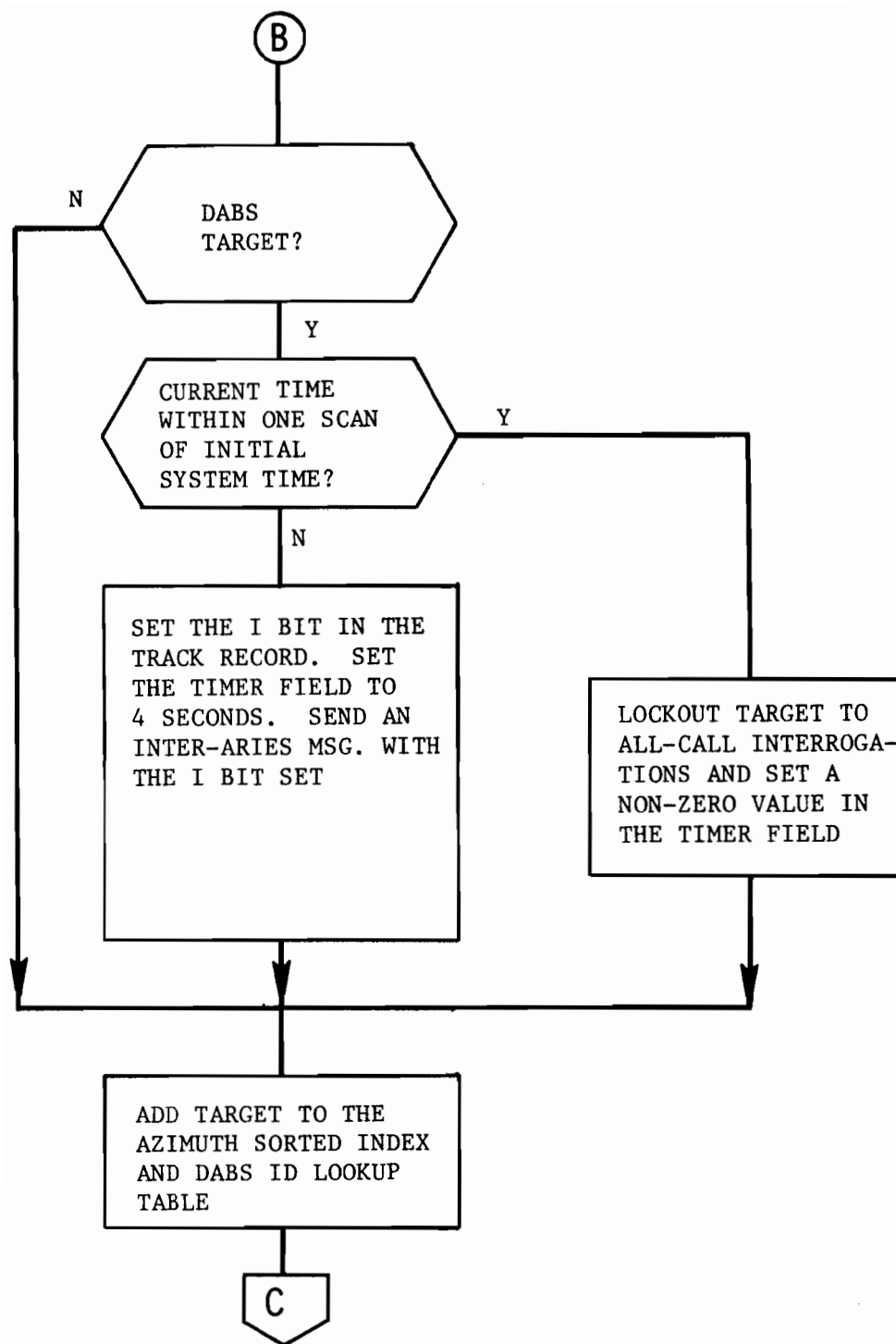


Fig.3.4-1. Continued.

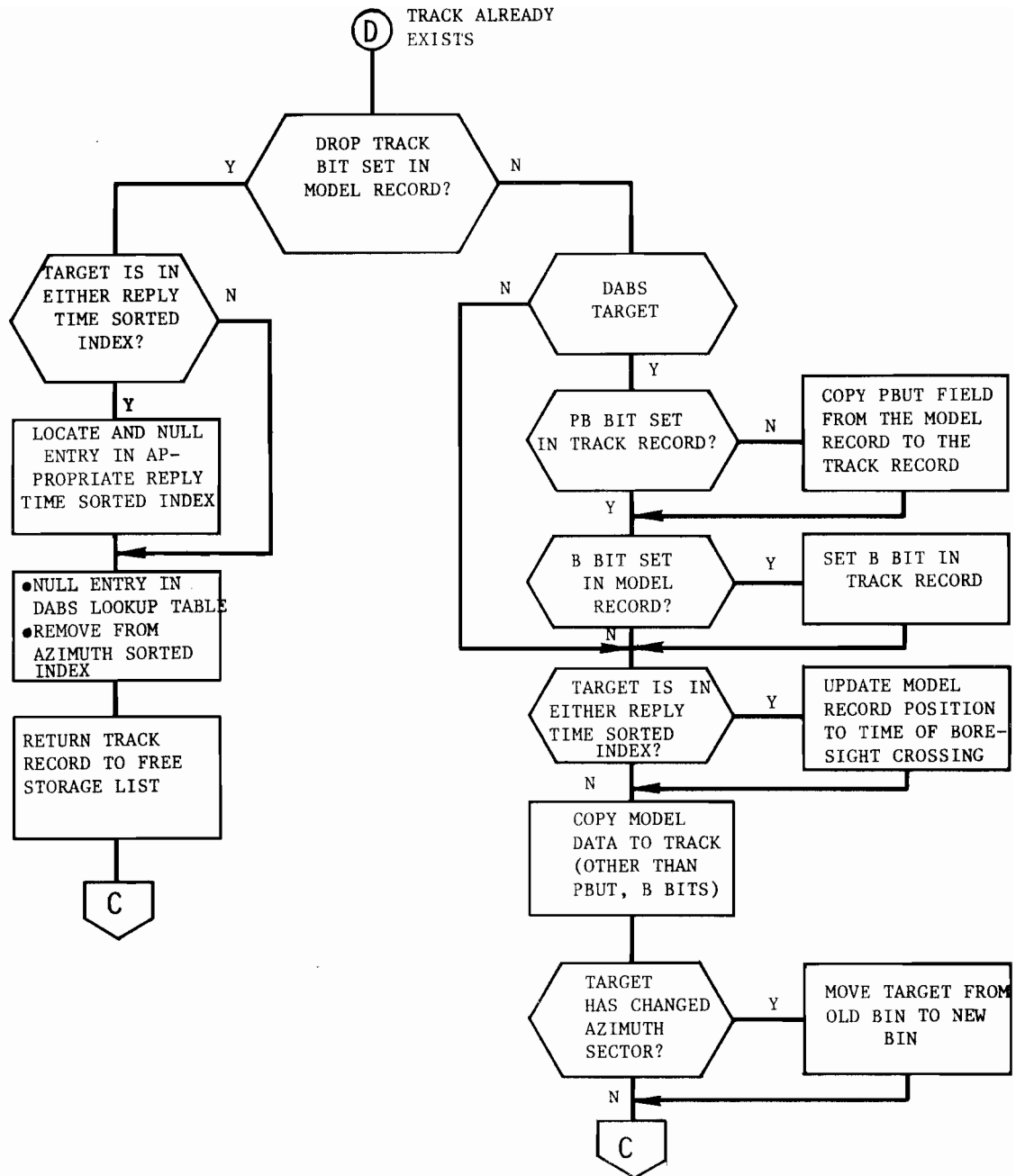


Fig.3.4-1. Continued.

Each logical record contains a time tag as its first words. This is the system time at which this data is to be inserted into the Track File. Traffic Model Input processing involves comparing the time of the next record with the current system time from the system clock. When the system time is greater than or equal to the record time, the record is processed. If the next record's time is greater than the system time, then the real time clock interrupt handler is initialized to awaken Traffic Model Input at the desired time. Traffic Model Input gives up the CPU in the interim. If the MSB of the time field is set, that record is treated as the end of the model file.

### 3.4.3 Starting a New Track

The first action upon starting to process a model record is to use the DABS ID to locate the track record via the DABS ID Lookup Table. ATCRBS targets are assigned DABS ID's for this purpose even though they are not DABS equipped. The low order bits of the DABS ID are a unique identifier for all tracks in the system, and can be thought of as a track number. The reason for using a lookup table to access the Track File, rather than just using these bits as an index directly into the track file array, is that it is possible that not all tracks are visible to all ARIES sites simultaneously. With the lookup scheme it is possible to have up to a total of 1024 uniquely identified tracks in a multi-ARIES system, while any one ARIES needs to store track file information on only 400 or fewer of these. The net result is a considerable reduction in the amount of storage required by the Track File.

If the Lookup Table does not point to a track record for the given ID (i.e., the pointer value at that location is null) then no track record exists for that target, and a new track must be started. Note that no indication is given in the model record itself that this is a new track, and this decision is based solely on whether the specified target is currently in the track file. This feature allows the model file to be started at any point in time, and not just at the beginning. Targets will automatically be entered into the system at their first appearance.

At startup of the model, the appearance of a large number of targets at once can cause a severe garble situation which prevents the sensor from acquiring the targets on all-call. To avoid this, a special initialization indication is set for the first scan. Traffic Model Input tests this condition. If set, it arbitrarily locks out the DABS targets to all-call and sets the track timeout timers to various values (it simply increments a counter for each new target processed and stores the low order bits of this counter in the track timer). Thus the targets will gradually come out of lockout and be acquired sometime during the initial 32 seconds of operation. Other than this precaution, new track processing proceeds as at all other times.

Upon recognition that a model record describes a new target, a new track file record must first be obtained from the pool of free track records. The data from the model record are inserted. All fields not initialized from the model record are zeroed, except in the case of the model startup procedure described above.

Next, the target position is checked to see if it is currently in the antenna beam (as determined from the Start and End Bin indices in the Azimuth Sorted Track Index) in which case there is some possibility of it being immediately interrogated. To insure correct operation, it is safest to always update new tracks which fall in the beam to time of boresight crossing before insertion into the Azimuth Sorted Index and DABS ID Lookup Table. This update is done as described in Section 3.3.4. The actual insertion is trivial, involving linking the track record onto the beginning of the azimuth bin list corresponding to its updated azimuth, and inserting a track record pointer into the previously null location in the DABS ID Lookup Table.

Both of the above operations are safe. The Azimuth Sorted Index is looked at only by Pre-interrogation Processing, which is non-preemptive (i.e., it cannot interrupt while Traffic Model Input is running). The Lookup Table insertion consists of a single store operation, and is therefore safe even though this file is used by the interrupt driven Interrogation Processing program. No other part of the system knows about the track until these insertions are made.

For reasons described in Section 3.5.7, the new track must be marked as uninitialized by the I bit being set in the track record. A track status message must then be sent to adjacent sites to request the current transponder status (I=1 in the track status message) and a count of 4 seconds set in the track's timer. Upon receipt of a reply containing this status information, or if no reply is received within 4 seconds, the track will be set to active (I=0) status by Adjacent ARIES Input or by Timer Countdown, respectively. The I bit is not set for new tracks within one scan of system startup, in order to avoid a large and unnecessary transient load on the communications functions. No discrete interrogations will be sent during this interval and so no transponders will have changed state.

#### 3.4.4 Update to an Existing Track

If a pointer exists in the DABS ID Lookup Table for the given ID, then the track record needs to be updated. If the old and new positions of the target are not in the antenna beam, then the model data are simply copied directly to the corresponding fields of the track record. Exceptions to direct copying are that the PBUT field should only be copied if the PB bit = 0, and the B bit should only be copied if it is 1 (B in the track is reset only by CB in an interrogation). If the track moves to a new azimuth bin in the process (or moves in or out of the 5 nmi inner bin) it must be deleted from its current position in the Azimuth Sorted Index, and reinserted at the proper location. This is safe due to the non-preemptive character of Pre-interrogation Processing, the only other user of that file.



If the track's old or new position falls between the Start and End Bin indices of the current antenna position, this simple update cannot be done. The reason is that tracks within the beam of the antenna have been predicted to time of boresight crossing, and an update might disturb that condition. Furthermore, tracks within the beam are subject to access by Interrogation Processing, an interrupt driven task. Since the time, round trip time (range), and azimuth parameters of the track are related and involve multiple store operations for update, this gives rise to a non-safe file interaction between the two programs. Parameters other than these three are either not used by Interrogation Processing or else are not related to other parameters and can be updated by safe single store operations. The following protocol between Interrogation Processing and Traffic Model Input eliminates the unsafe aspects of this operation.

First, Traffic Model Input adjusts the time, round trip time, and azimuth in the model record to the time of boresight crossing, as described in Section 3.3.4. It then disables interrupts, thus effectively locking out Interrogation Processing, and copies the new data into the track file. It then enables interrupts again. The few microseconds required for this do not exceed the worst case interrupt latency already expected.

Note that all modifications to word 10 of DABS tracks (the transponder state bits) also must be made either with interrupts disabled or by the uninterruptable bit setting and clearing instructions on the Eclipse. In particular, it is not safe to load this word into a register with interrupts enabled, as it is written into by Interrogation Processing.

The track record at the end of this operation still satisfies the condition that it is updated to the time of boresight crossing. There is one potential problem, in that the Reply Time Sorted Index is based on the old range value and not the updated range value. Therefore, the list may no longer be perfectly sorted. However, the new position cannot be more than a few range units away from the old one, or else the model is not updating the track frequently enough to provide smooth continuous tracks. Any difference this could cause will not affect the generation of replies except in those cases where a multiple garble situation for ATCRBS or all-call replies overloads the available reply generators and forces the software to drop some replies. The effect will be that different replies may get dropped than would have been the case if the list had been re-sorted.

After updating the track record, the Azimuth Sorted Index may need to be updated. This is done exactly as in the case of a target which is not in the beam, and is safe for the same reason.

#### 3.4.5 Deleting a Track

The traffic model records may also indicate that a track is to be dropped. This will happen if a target leaves the airspace that could

conceivably be covered by the sensor for which that model data is intended. This also allows for aircraft which land during the exercise. It is necessary to remove the corresponding track from the DABS ID Lookup Table and the Azimuth Sorted Index in order that it not be interrogated. The track record is returned to the free storage pool for reuse.

All of these can be easily done if the track is not in the antenna beam, as specified by the Start and End Bin indices of the Azimuth Sorted Index. The entry in the Lookup Table is set to null, the track record is deleted from its azimuth bin and is linked to the free storage list. The operations must be performed in that order to assure safety.

It is not safe to delete a track which may be interrogated. Discrete interrogations can be safely stopped by nulling the entry in the DABS ID lookup Table. A similar result can be obtained for ATCRBS/All-Call interrogations by nulling the track's entry in the Reply Time Sorted Index. Both these operations should be performed for tracks which are to be deleted while in the beam before any other deletion operation is performed. Once it is assured that interrogations will not be directed to the track, the track can be safely returned to free storage. Deletion from the Azimuth Sorted Index is not critical and can be done either before or after stopping interrogations.

### 3.5 Inter-ARIES Communication\*

#### 3.5.1 Purpose

The purpose of inter-ARIES communication is two fold. First, a multi-ARIES simulation is simulating a single aircraft environment as seen by several sensors. These sensors communicate with each other about the aircraft visible to them, and interact with the aircraft based on the assumption that each sensor is seeing the same environment. To assure that the ARIES environments look to the sensors like a single environment, they must be correctly coordinate converted to reflect the sensor's position (a function of model generation) and must be kept in synchronism. This latter is the first function provided by inter-ARIES communication. Messages are provided which allow the models at all sites to be positioned to a particular point in time, to be started and stopped simultaneously, and which verify this synchronization during operation.

The second purpose served by inter-ARIES communication is to perform a similar synchronization function for the simulated transponders' hidden state variables. Much of the target state is determined by the traffic model, and therefore is synchronized between sites as long as the models remain synchronized. However, certain DABS transponder states are determined at least partly by received interrogations. These are the lockout state, the state of the Acknowledgement Request (AR) - Pilot Acknowledgement (PBUT) cycle, the state of an air initiated Comm-B request, and the transponder timeout status (which

---

\* These functions have not been implemented. They are documented here to assist future implementation.

affects lockout and the AR-PBUT cycle). It is desirable that the states of all the ARIES' models of a given target be consistent as to that target's internal state. In order to achieve this, each ARIES informs its neighbors whenever one of its targets changes internal state. Corresponding changes can then be made in the neighboring ARIES' track records.

It is, of course, not possible to have all ARIES in precisely the same state all the time, due to delays in transmitting data over telephone lines. In the discussions of individual protocols below any situations where this has an effect on the replies will be mentioned. Appendix H contains a more detailed analysis of message delay effects.

### 3.5.2 Message Formats and Line Protocols

Each pair of ARIES will communicate over a full duplex synchronous line at a data rate of at least 2400 bits per second. This data rate is sufficient to allow for the transmission of one track status message for each of 400 tracks in one 4 second scan time, ignoring line protocol overhead. This is considered to be a worst case loading of the communication system, and corresponds to the possibility of all tracks receiving DABS lockouts during initial acquisition. While this worst case is considerably worse than the anticipated average load, this line speed will reduce the delay time in synchronizing the state of a track at all sites. The best case delay for a single message will be approximately 1/10 second.

The formats for track and system status messages, and for the transmission blocks in which they are included, are presented in Appendix E, Figures E-9 and E-10.

A line protocol must provide for two basic failure modes, which in turn can lead to several recognizable error conditions. These two basic failure modes are:

- 1) a transmission block is received in error, and
- 2) a transmission block never reaches its destination

To handle the first failure mode, each transmission block includes error detection coding. Each transmission block contains a unique sequence number, and the receiver must acknowledge each block received. The transmission block format also provides space for this acknowledgement. At the time a transmission block is sent to a given neighboring ARIES, the transmitting ARIES either includes an ACK (acknowledgement) field and the block sequence number of the last block correctly received up to that time from that neighbor (indicating acceptance of all messages up to and including that one), or a NACK (negative acknowledgement) field is sent with the same block sequence number (indicating the same acceptance condition plus a request for retransmission of all subsequent blocks).

Complete line failure, either transient or permanent, requires additional considerations. The block sequence numbers provide protection against transient failures. A receiver expects to see successive sequence numbers over a given line. If the expected sequence number does not appear, it is assumed to have been lost and a NACK is sent back to the transmitter. This assumes that there is a following message which is received.

More permanent failures are detected by three means:

- 1) When a transmission block is created, a time is associated with it. If it is not accepted by all other ARIES systems within 4 seconds, an error condition is signalled in the transmitting ARIES.
- 2) If the block sequence numbering for a given line wraps around and a sequence number of a block not yet accepted by all other ARIES must be reused, an error condition is signaled.
- 3) If the buffer space provided for messages overflows (due to failure of one or more adjacent ARIES systems to accept the corresponding transmission blocks), an error condition is signaled.

A receiver will acknowledge a block even if it itself has no messages to transmit. It is therefore legal to send a transmission block containing no message bytes, solely for acknowledgement purposes.

### 3.5.3 Overview of the All-Call Lockout Protocol

This and the following sections provide an overview of the processing of the various items of information handled by the inter-ARIES communication system.

There are two bits in the track status message which report lockout state. L=1 indicates that the track's lockout status changed to lockout, and U=1 indicates that the status changed to unlocked. Track status messages with either of these bits set are normally generated by Interrogation Processing in response to an interrogation. This program simultaneously changes the local lockout status for that track (the L bit in the Track File) accordingly. Adjacent ARIES Input may also generate such messages, as described in Section 3.5.7.

The receipt of these bits by Adjacent ARIES Input causes the L bit in the local track file record for that target (if such a track exists) to be set if L=1 and reset if U=1. Thus, after a message transmission delay time, all ARIES will agree as to the lockout status of the target, and this status will correspond to the last lockout command received from the sensors.

There is an exception to the previous statement, however, due to the message delay time. Under certain circumstances it is entirely possible for

two sensors to transmit conflicting lockout states to a target. Typically this can happen at airspace boundary regions where, due to the quantization introduced in the sensor's coverage maps, two sensors do not agree as to the precise boundary location. In the case of a real target, this would result in the lockout state of the transponder oscillating to correspond to the status desired by the last sensor to interrogate it.

In the case of ARIES' simulated targets, the result will be similar except that under certain circumstances the final state will always be the opposite of the initial state irrespective of which interrogation was received last. This will happen if two sites receive conflicting interrogations within about 1/10 second of one another. Please see Appendix H for details.

If conflicting commands are not simultaneous to within the inter-ARIES message delay at the time of interrogation, the simulated situation will exactly mirror the real world situation except during message transit times during which the two sites will disagree as to the transponder state.

#### 3.5.4 Overview of Acknowledgment Request (AR) Processing

There are two bits in the track status message dealing with this protocol; the AR and CP bits. Track status messages with these bits set are normally generated by Interrogation Processing (Adjacent ARIES Input may also generate them - see Section 3.5.7). The AR bit is sent in a track status message whenever the AR bit is set in an interrogation for a track. This bit is set in an interrogation in order to solicit a pilot response to an uplink message.

The CP bit is set in a track status message whenever it is set in an interrogation, the AR bit is not simultaneously set in the interrogation, and PB=1 in the track file (indicating an active pilot response - see Section 3.2.3.5). CP is sent by the sensor whenever it has received the desired pilot response and wishes to inform the transponder that it can clear the pilot response (PBUT) bits.

The pilot response protocol of a real DABS transponder is shown in Fig. 3.5-1. On receipt of an AR bit, the transponder enters State 1 and begins a short timeout after which it enters State 2. The purpose is to inhibit pilot response (perhaps to a previous message) until the new message has been displayed to the pilot. From this state, the protocol goes to State 3 or 4 depending on whether the Yes or No response buttons were pressed. The PBUT bits in all subsequent replies are set to indicate which response was made, until such time as a CP bit is received, sending the transponder back to State 0 and clearing the PBUT field. At any time in this protocol, receipt of further AR bits resets the

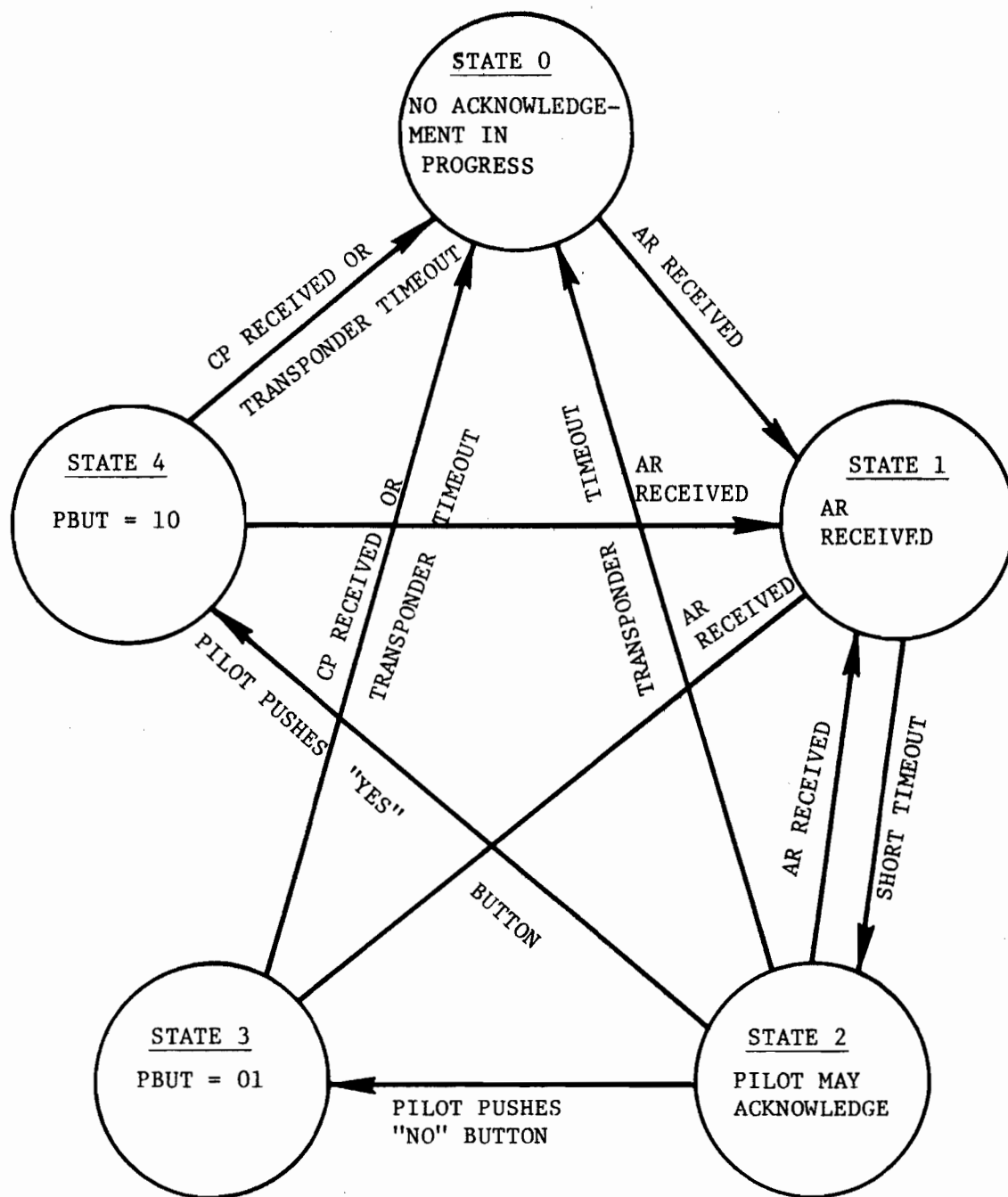


Fig.3.5-1. Pilot response protocol.

transponder to State 1. CP has no effect except in States 3 and 4, and pressing the Yes or No buttons has no effect except in State 2. Transponder timeout also resets the protocol to State 0 (see Section 3.5.6).

The AR, PB, and PBUT fields in the track record are used to approximate this behavior. The AR bit in the track record is set by either Interrogation Processing or Adjacent ARIES Input on receipt of an AR bit in an interrogation or track status message respectively. Simultaneously PB is set to 0. This corresponds to State 1 above.

PB is set by Pre-Interrogation Processing when it prepares a track for interrogation and finds the AR bit set and PBUT=01 or 10. (AR, PB) = (1, 1) corresponds to State 2 above. The setting of PB by Pre-interrogation Processing approximates the timeout function, assuring that a pilot response is not made to the same sensor in the same scan that AR was sent.

The PBUT field in the track record is set from the model tape by Traffic Model Input as long as PB=0. If AR=1, PB=1, and PBUT=01 or 10, the track state corresponds to transponder State 3 or 4. The PBUT field from the track record is then copied into all discrete replies from that target generated by Interrogation Processing. The restriction that PB must be 0 to change PBUT arises from the corresponding feature of the transponder that the acknowledgement buttons have no effect in States 3 or 4, i.e., the pilot cannot change his mind. It is a requirement on the traffic model generation programs that any changes in PBUT occur simultaneously at all ARIES sites.

If CP is received in an interrogation while PB is set, AR and PB are reset, corresponding to State 0. If CP is received via an inter-ARIES message, AR and PB are reset irrespective of their current state as the presence of CP in the message indicates that the reset conditions were satisfied at an adjacent site.

The AR and PB bits can also be reset by the transponder timeout function described in Section 3.5.6.

ARIES does not support the transponder test mode (PBUT = 11).

Notice that by the above protocol Interrogation Processing need only look at PB to determine whether or not to insert the PBUT field in a reply.

The effects of message delay on this protocol are relatively minor, as long as changes in the value of PBUT are not occurring at the same time. Under some circumstances an extra PBUT response may be sent that would not have been sent in a live aircraft test. Under other conditions the ARIES first receiving the AR may "lose" it, although the other ARIES will respond. Please see Appendix H for details.

A final, undesirable, case can occur if the traffic model attempts to change the PBUT field while an acknowledgement cycle is in progress. There can exist an interval, due to message delay and the difference in the times a target will be processed by Pre-Interrogation Processing at different sites, when PB is set at some sites and not others. If the model update occurs at that time, sites with PB=0 will accept the new value and the others will not. Until the end of that acknowledgement cycle this discrepancy will remain. For this reason, it is recommended that the model not try to alter PBUT at points where acknowledgement requests are expected in a scenario.

### 3.5.5 Overview of Downlink Request Processing

The CB bit in the track status message is the only track status bit concerned with this protocol. It is set only by Interrogation Processing upon receipt of a CB bit in an interrogation.

In the real transponders, an air initiated downlink is started by the transponder setting the B bit in a reply, thereby requesting channel time for a long downlink. The sensor eventually allocates this time and receives the message. It then sets the CB bit in an interrogation in order to clear the B bit in the transponder, completing the transaction.

The B bit for a target is set simultaneously at all ARIES sites from the traffic model file. This simultaneity is a requirement on the model generation programs. Unlike other track record fields, which are always updated by a new traffic model record, the B bit can only be turned on by the traffic model. It can be turned off only by receipt of a CB bit in an interrogation or in an inter-ARIES message. Unlike the preceding two protocols it is not affected by transponder timeout.

Message delay usually will have no effect due to the "primary sensor" feature of the DABS system. Normally only one sensor, the primary sensor for that aircraft, will respond to a B request, and only it will send CB. Thus inter-ARIES messages play little part in the normal protocol. Some pathological cases can occur however. In one, the sites are sufficiently out of time synchronization that a CB arrives at the adjacent site before B is set. By unlikely chains of logic, an extra copy of the message might be sent. The synchronization protocols should prevent this occurrence.

Other cases can occur if the designation of which sensor is primary and which secondary changes during the protocol. The logic of this is complex, and the sensors avoid switching primary designation during the actual downlink protocol, but due to inter-ARIES delays it is conceivable that the sensors could consider the protocol complete and switch designations before the adjacent ARIES are informed of the CB. An extra copy of the message might then be downlinked. This has no serious effect, as the messages sent are meaningless fixed format messages in any case.



Thus extra Comm-B messages beyond what would be sent by the real transponders are the worst consequence of inter-ARIES message delay for this protocol.

### 3.5.6 Overview of the Transponder Timeout Protocol

DABS transponders have a timeout feature whereby they reset their state if they receive no discrete interrogations for a period of 16 seconds. Resetting involves setting the lockout state to unlocked and setting the AR (acknowledgement request) protocol to State 0. The air initiated downlink (B-bit) is not affected.

To simulate this effect, the ARIES track records contain a counter which is set to 16 seconds at each discrete interrogation and counted down by one each second by Timer Countdown. If this counter reaches zero, a track status message with the T-bit set is transmitted. Receipt of a discrete interrogation while in this timed-out state causes Interrogation Processing to send a track status message with the DI-bit set.

In addition to the timeout counter, each track record contains status bits indicating the track's timeout status at the adjacent sites. For a new track these always start at 0 (timed out). They are set upon receipt of DI-bits and reset upon receipt of a T bit. A transponder is considered to have timed out at a site if and only if all adjacent sites have timed out (their status bits are 0) and the local timer has counted to zero.

Timer Countdown processes each track once a second. When it recognizes the timed out condition (status bits and timer = 0) it resets the L, AR, and PB bits. The I-bit is also reset for reasons described in Section 3.5.7. Adjacent ARIES Input performs a similar timeout function if all the timeout conditions are satisfied after receipt of a T bit from another site. Thus, a transponder times-out only when all sites agree that it has timed out. The timeouts at each site may occur at slightly different times, the maximum difference being the message delay. During this delay, the timed-out site may receive all-call replies while the other sites will not (if the target is locked out), and if the delayed sites do finally interrogate discretely they may find a PBUT response that had been cleared at the timed out site.

### 3.5.7 New Track Initialization

When a track is newly started up at a given ARIES site, as described in Section 3.4.3, it is entirely possible that the same target has existed for some time at one of the other sites. The target may have acquired lockout and other status values different from the initial zero value assigned to the new track. To obtain correct and consistent operation, these status bits must be correctly initiated. To do this, Traffic Model Input sends a track status message with the I bit set to each of the adjacent sites for each new track, and sets the I bit in the track record. While the track is in this state, it will not be processed by Interrogation Processing. The track's timer is also set to a count of 4 seconds.

No messages with the I bit set are sent within about one scan of the initial system time. No other ARIES have received discrete interrogations during this interval, so no track status changes have occurred. This reduces the chances of a large, unnecessary, system startup transient.

The receipt of a track status message with the I bit set causes Adjacent ARIES Input to send back a track status message designed to initialize the new track's state appropriately. The new track is set up by default as unlocked, timed out both locally and at adjacent sites, and with no pilot acknowledgement cycle in progress. If the other sites disagree with any of these settings, they can alter the new track's status by means of appropriate track status bits. No message is returned if the adjacent site has no knowledge of the track or, of course, if the adjacent site is not active.

A consistent state for the downlink request is the responsibility of the model tape generation programs. If a new track is started with a downlink request active, this request should simultaneously be activated at the other sites.

The track will be set to an active, initialized status ( $I = 0$ , track timer = 0) by its local Adjacent ARIES Input upon receipt of the first track status message, and completion of the corresponding state initialization. Further track status messages will be treated normally.

To handle the case where adjacent sites are not active for this simulation run, or where for some reason they do not know about the track in question, a timeout procedure is provided. If Timer Countdown counts the track timer to zero, it will set the I-bit to zero, activating the track.

### 3.5.8 System Synchronization

In order for ARIES to work properly in a multisite configuration, the traffic models at each site must maintain time synchronization. Since the track status messages take about 1/10 second (best case) to travel between sites, this will (somewhat arbitrarily) be taken as the synchronization goal. That is, the clocks at all sites are started in and maintain synchronism within 100 msec. Timing information is passed between sites and used to check for correct synchronization. Failure of these checks causes an error condition to be raised.

System synchronization is initiated and checked by means of system status messages. These messages are sent in separate transmission blocks from track status messages and are treated specially by the inter-ARIES message processing functions.

The system operators initiate system synchronization by requesting the system to initialize itself to a particular time on the model file. This request is processed by Operator Communications, which starts up the local System

Initialization program, which in turn sends a system status message to adjacent ARIES informing them that initialization is taking place and specifying the starting time.

Upon receipt of such a message, Adjacent ARIES Input starts its own local System Initialization program. Notice that this will cause a system initialization message to be sent back to the original ARIES, as System Initialization always sends this message when it starts.

If an ARIES is already initialized to the time specified in a system initialization message, a "system ready" message is sent to the other sites. If the system is in the process of initializing to the specified time, no message is transmitted. These messages are sent by the Adjacent ARIES Input program.

When a site completes initialization, it transmits a "system ready" message to the adjacent sites. The recipients merely record this information in their system status data. When all three sites are waiting at the specified time the system is ready to be started, and the operator is so informed.

The operator at one of the sites then enters a command to start the system. System startup messages are formatted and sent to all adjacent sites. The time it takes for these messages to be received can be calibrated, and is on the order of a few tens of milliseconds. The originating sensor delays its own startup by this period of time in order to assure closer synchronization.

Startup messages are treated as a special case by the communication system's transmission block acknowledgement protocol. Receipt of a request for retransmission (NACK) for these messages raises an error condition and causes a system halt. The operator is informed, and must then reinitiate the system initialization process.

System halt messages are generated by Operator Communications whenever the system is halted by operator command. Receipt of such a message causes the recipient to be halted. Thus total system shutdown is effected.

The procedures above are designed to start all ARIES in synchronism. Under normal circumstances, this synchronism should be easily maintained within the specified 100 msec by the real time clocks at each site (in fact, any errors in synchronism are due far more to errors in start times than to clock drift). However, to check for possible error conditions which might give rise to loss of synchronization, each site periodically transmits a synchronization time message to adjacent sites.

This is another special case message, and is generated by the inter-ARIES communication programs themselves. The real time clock interrupt handler sets a flag for these programs once each second. Whenever a telephone line becomes free and this flag is set a system status message is transmitted over the line before any further track status messages. The system time

and status bits are sampled immediately prior to transmission. The reason for this procedure is that it guarantees a relatively stable known delay between the sampling of the clock at one site and receipt of the message at its destination. This known correction can be applied to the received message time before comparing it to the recipient's system clock. If the two values do not agree within 100 msec, a system error is signaled.

If a synchronization time message is rejected by its recipient (NACK), the original message is not re-transmitted. Instead, the system clock and status are again sampled and a new message created (but with the same transmission block sequence number). This is required in order to assure the known time delay described above.

#### 3.5.9 Adjacent ARIES Input Task

Fig. 3.5-2 is a functional flowchart of Adjacent ARIES Input. The processing performed has been fully described above under the various overview sections.

Inputs: Modem Input Buffer.  
System Status File.  
DABS ID Lookup Table.  
Track File.

Outputs: Modified Track File.  
Inter-ARIES messages.  
Modified System Status File.

Timing: Activated by the modem input handler upon the arrival of messages from other sites.

Note that in manipulating the transponder state bits in track word 10, uninterruptable bit setting and clearing instructions must be used or interrupts disabled in order to prevent undesirable interactions with Interrogation Processing.

#### 3.6 Track File Timer Countdown

The logical function of this task is described in Section 3.5.6. A functional flowchart for this task is presented in Fig. 3.6-1. This task's input/output behavior is as follows:

Input: Track File.  
DABS ID Lookup Table (Used to locate active track records).

Output: Updated Track File.  
Inter-ARIES messages.

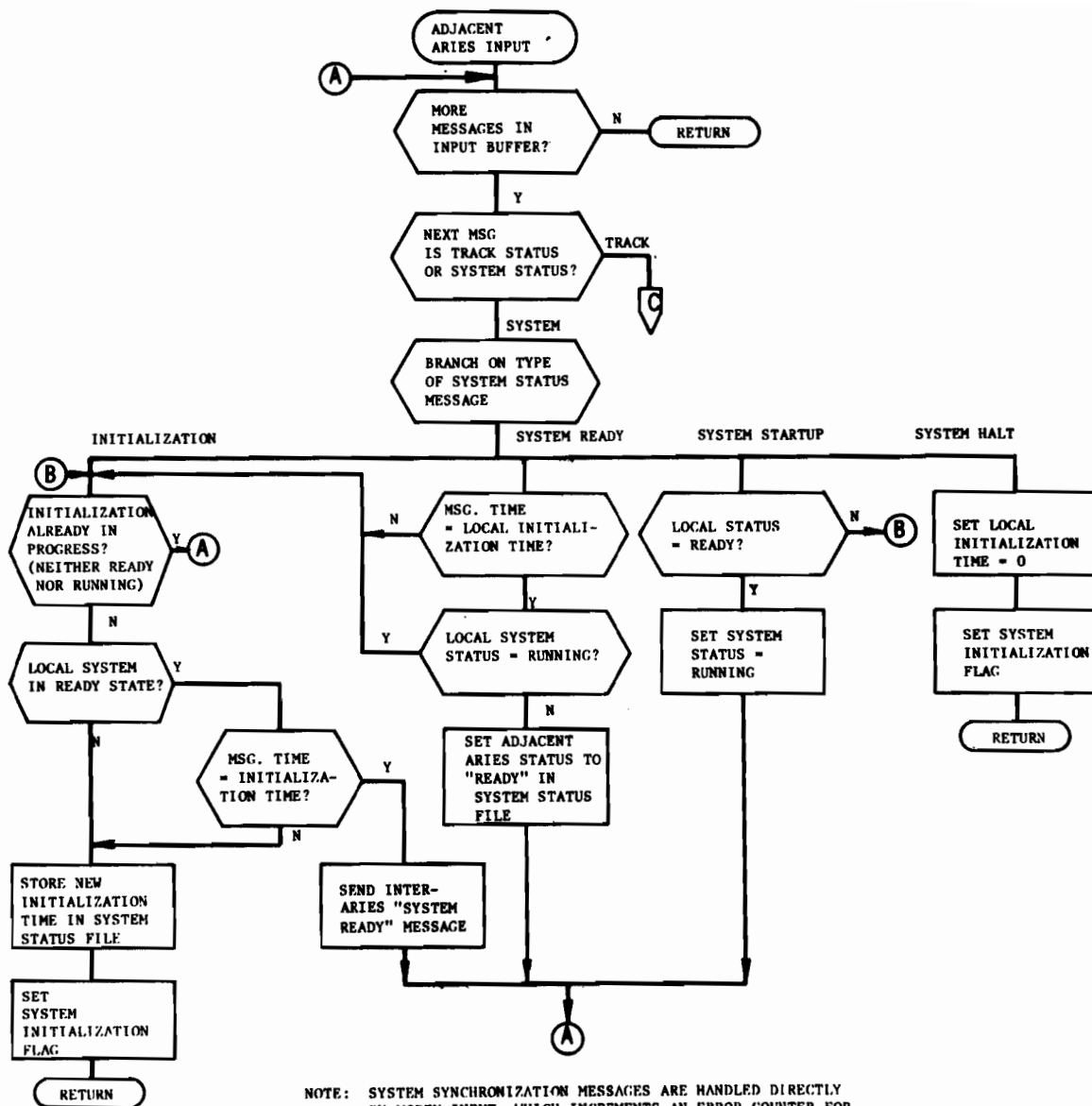


Fig.3.5-2. Adjacent ARIES input.

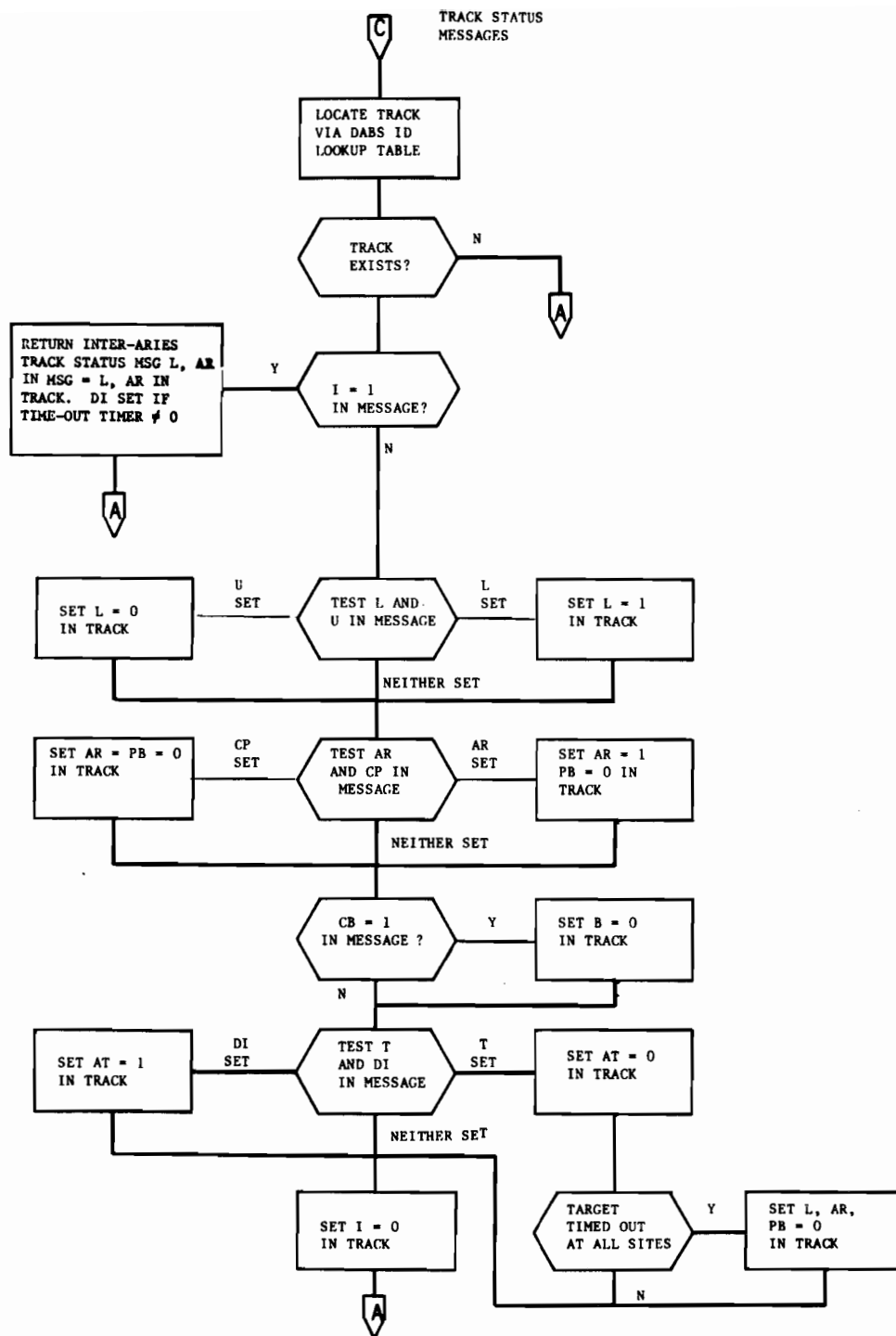


Fig.3.5-2. Continued.

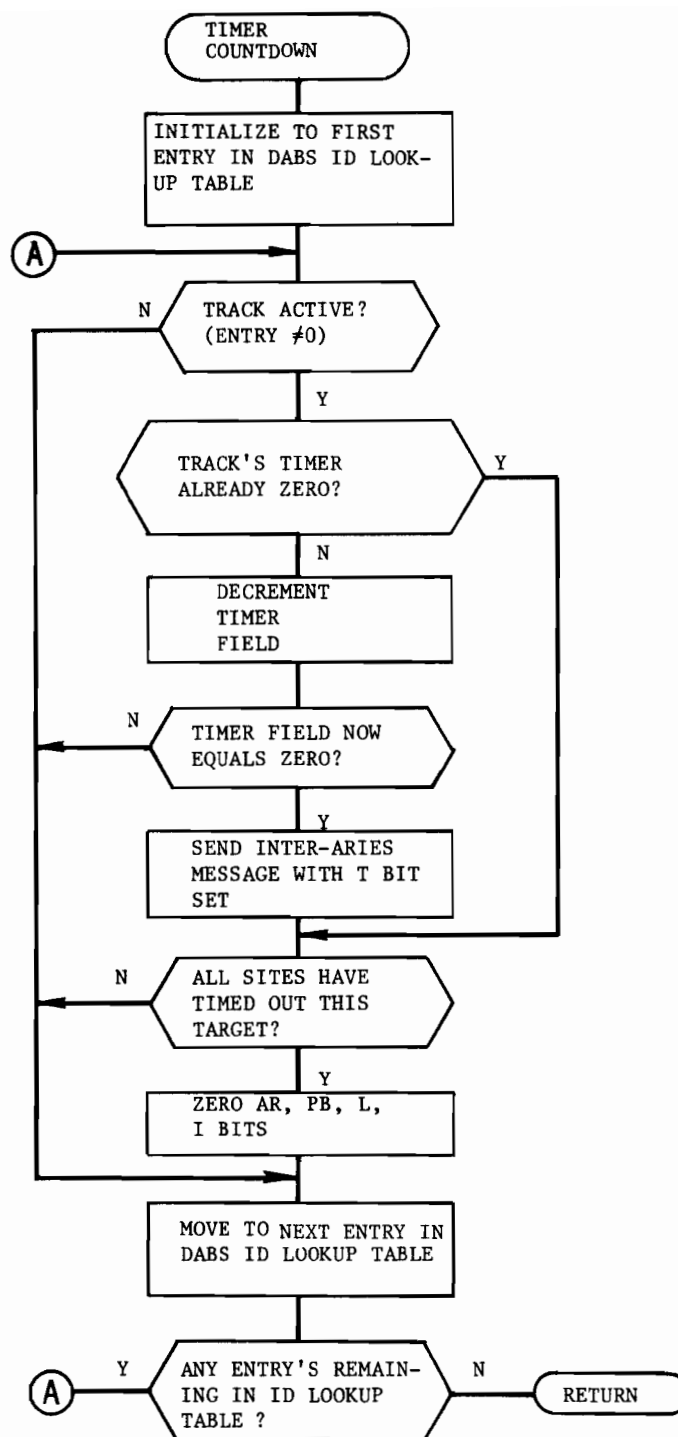


Fig.3.6-1. Timer countdown.

Timing: Started once each second by the real time clock interrupt handler.

Note that interrupts must be disabled while track work 10 is being manipulated, to prevent undesirable interactions with Interrogation Processing.

### 3.7 Operator Communications

Each ARIES site has a terminal by means of which the local operator can interact with the system and control its operation. Operator Communications prompts the operator to enter a command number. Once this is done, it will prompt the operator to provide any needed input parameters. Both the command numbers and all parameters are integer numbers. After receiving all needed parameters, Operator Communications executes the desired function and, unless the operator has requested that the system be halted or re-initialized, again prompts the operator for a new command number.

The available commands and the corresponding command numbers are listed below, and each command is discussed individually in the following paragraphs.

1. Start the simulation running from the current initial time.
2. Halt the simulation and restore the normal operating system.
3. Halt the simulation and reinitialize to a time specified by the operator.
4. Print system status information.
5. Allow the operator to create a target.
6. Allow the operator to drop a target from the system.
7. Allow the operator to change data recording modes for DABS and ATCRBS interrogations.
8. Enable/disable interrogation data recording.

#### 3.7.1 Starting the Simulation

This command requires no operator supplied parameters. It first sends a startup message to the other ARIES sites as described in Section 3.5.8, then clears the system waiting status bit and sets the system running status bit. Both of these are in the status flag word in the System Status File. It then prompts the operator for a new command.

To understand how this starts the system, certain background information is required. First, Operator Communications does not issue the command



number prompt until both the local and adjacent ARIES sites are fully initialized and waiting to begin the simulation, as indicated by status bits in the system status file. This implies that all sites have acquired the interrogation pattern of their local sensor and have positioned their model file to the specified initialization time. At this point, Operator Communications types a "system ready" message, then gives the operator a chance to review the list of available commands, and finally issues the command number prompt.

At this point, the only reason the system is not running is that the real time clock interrupt handler is not incrementing the system clock, or readying any of the timed tasks. It will only do this if the status bit which indicates the system is running is set. When this is done by Operator Communications, the various timed programs begin executing, and traffic model records are processed and stored in the Track File at the appropriate system times.

### 3.7.2 Stopping the Simulation

When this command is given, a system halt message is sent to all other ARIES systems and an end-of-file is written on the data recording tape, if data is currently being recorded on tape instead of disk. A bootstrap procedure is then executed which is equivalent to the standard manual system bootstrap procedure. The usual "FILE NAME?" prompt, will be printed by the bootstrap programs. The operator may then either restart the ARIES software system, or restore the Data General RDOS operating system.

### 3.7.3 Reinitializing to a New Starting Time

This also stops the simulation, except that instead of bootstrapping the operating system the operator is prompted to supply a new starting time. This is inserted in the initial system time field of the System Status File. System Initialization is then run. The model file will be positioned to the new time and the system initialized to begin another run.

### 3.7.4 System Status Command

This prints a variety of useful status information, including the following:

- Current system time
- Current system status word setting
- Number of DABS and ATCRBS targets
- A summary of the status indications from the Status Formatter hardware.
- A list of all system error counts that are non-zero and their current value.
- The percent of time the CPU was idle during the last second, and the minimum value of this one second idle time observed since the last execution of this command.

### 3.7.5 Starting a Track

This command prompts the operator for all necessary information about the target to be created. It then obtains a track record from the free storage list and creates the track, inserting pointers to it into the DABS ID Lookup Table and the Azimuth Sorted Index. It is not as careful as Traffic Model Input in the way it does this, not concerning itself with whether or not the target is in the antenna beam. Therefore, it is possible to get some anomalous behavior on the first scan of the antenna past the target.

If a track already exists with the same track number, Operator Communications will print an error message. It does not allow tracks to be updated from the terminal.

Once a track is created, it will be updated once each scan by Pre-Interrogation Processing. Since this updates the position directly in  $\rho$ ,  $\theta$  coordinates using the  $\hat{\rho}$ ,  $\hat{\theta}$  values in the track record, targets will in general move on spiral paths (of which radials and circles centered on the origin are special cases).

### 3.7.6 Dropping a Track

Operator Communications prompts for a track number. It then removes that track from the DABS ID Lookup Table and the Azimuth Sorted Index, and finally returns the track record to free storage.

### 3.7.7 Changing the Recording Mode

The data recording performed for both ATCRBS/All-Call and discrete interrogations has short and long modes. The long modes provide more information about what was happening, while the short modes provide some useful information while reducing CPU utilization and data recording bandwidths. For more information on data recording and the particular block formats, please see Section 3.9 and Appendix F.

This command allows the operator to change either ATCRBS/All-Call or DABS recording mode (or both). Initially the system records both in short mode. Operator Communications prompts the operator for the desired DABS mode and then the desired ATCRBS/All-Call mode.

Due to the nature of the implementation, it is unsafe to change recording modes if the system is running and interrogations are being processed and recorded. Therefore, this command will not allow either recording mode to be changed once the system has been started by command number 1.

### 3.7.8 Enabling/Disabling Interrogation Data Recording

At each execution the state of interrogation data recording is complemented. Initially recording is enabled. It can then be disabled by executing this command once, and reenabled by executing command 8 again. This command may be executed at any time. Disabling recording slightly reduces CPU utilization. Only interrogation recording is affected. Other data blocks will still be recorded.

### 3.8 System Initialization

This program controls the initialization of the ARIES system. Many of the required operations are performed directly. These include initializing system data structures, initializing the various I/O devices, creating the system tasks, and reading data from the disk to initialize the user programmable micro-code and certain of the Site Characterization Tables. Other initializations are performed indirectly by other system functions once System Initialization has set certain flags. These include ATCRBS/All-Call pattern acquisition, performed by Interrogation Processing, and positioning of the model file to the initial time, performed by Traffic Model Input.

System Initialization also sends an initialization message to all the other ARIES sites to inform them of the fact that the local site is initializing and also to specify the initial model time. As described in Section 3.5.8 this will cause the other sites to initialize to that time as well.

Once System Initialization has completed its initialization functions it reduces itself to being the lowest priority task, and takes on the combined functions of system idle loop and background task scheduler. The latter function is described more fully in Section 3.10. The idle loop consists of a 1 msec delay counter, which counts only when no other system task is running, followed by a few instructions in which an idle time count is incremented (providing a measure of the amount of CPU time being used by the system), and then any background tasks that need to be executed are called. Before returning to the 1 msec delay, the system status flags are checked. If these have all been cleared by Operator Communications due to a command to reinitialize, System Initialization will resume highest priority and reinitialize the system.

Figure 3.8-1 is a functional flowchart of System Initialization.

### 3.9 Diagnostic Data Recording

Data recording by ARIES serves two purposes. Primarily the recordings are used as diagnostic aids for debugging errors in the real time operation of the system. A secondary function is to record enough data about the

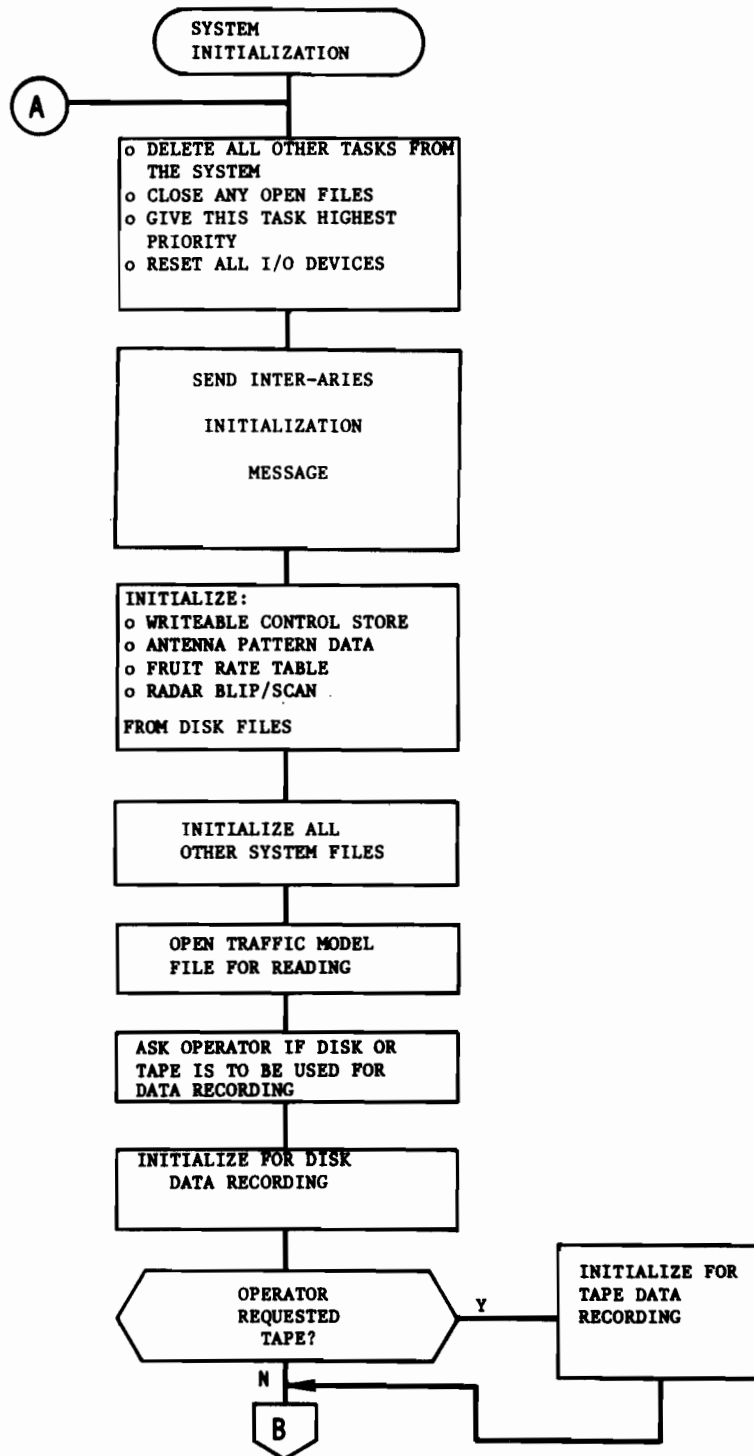


Fig.3.8-1. System initialization.

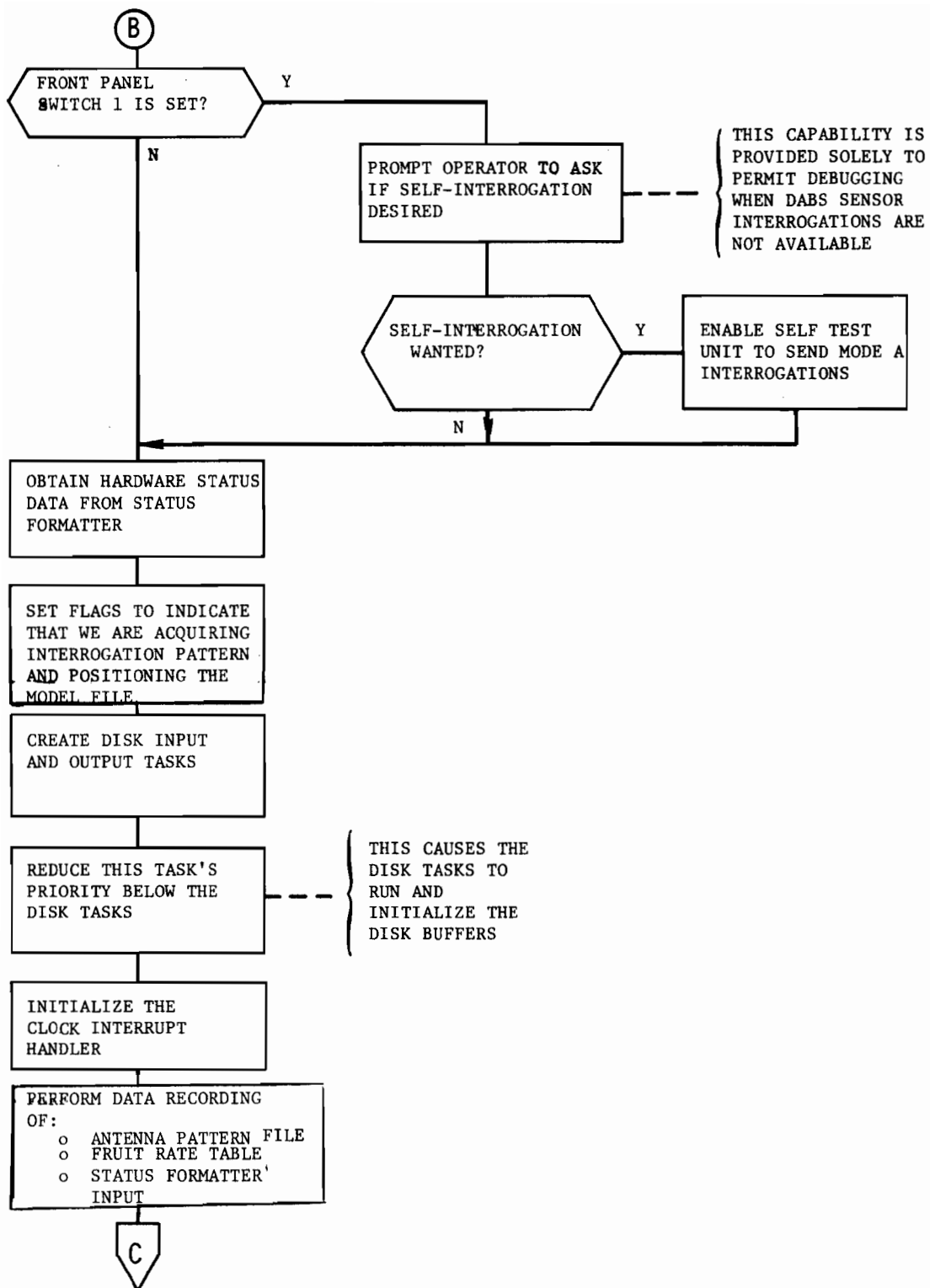


Fig.3.8-1. Continued.

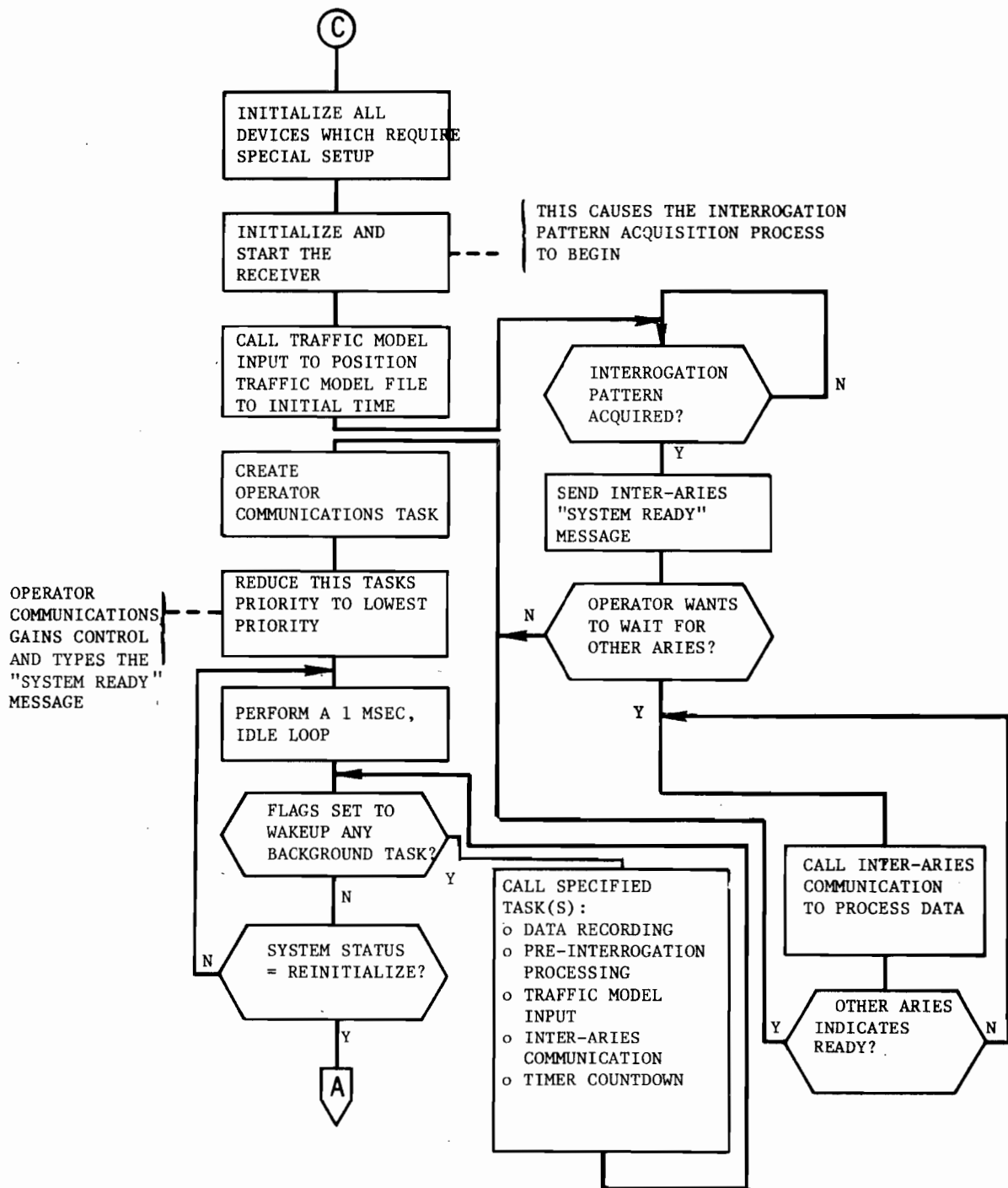


Fig.3.8-1. Continued.

interrogations received that the operation of the DABS sensor can be monitored. The data recorded consists almost entirely of the various hardware inputs. The data paths in ARIES are short enough and direct enough that it is not felt necessary to record internal data. In the case of the input of model data from the disk, the model record is not re-recorded, but only an indication of the disk block number within the file and when it was retrieved is saved. If necessary, later off-line analysis can retrieve the model data from that same file.

There is a Data Output task which writes these data to either a disk file or magnetic tape. System Initialization asks the operator to specify whether disk or tape recording is to be used. A data recording subroutine is also provided which allows any background system task (see Section 3.10 for a definition of background tasks) to record data, and which will format the data into I/O buffers for recording by the Data Output task.

If the recording is made to tape, it continues until an end of tape condition or tape error is sensed, at which time an end of file mark is written and recording is transferred to the disk.

If the recording is made to the disk, a limited amount of recording is supported, determined by the size of the disk file allocated for this purpose. If the end of the file is reached before the system stops, the Data Output task begins overwriting the initial records of the file. Thus, the file will always contain the last few minutes worth of data.

Data items are recorded in the format shown in Fig. 3.9-1. All data blocks are preceded by a three word header. The first word is an identifying block number, the second a count of the number of words in the block (including the three header words), and the third is the value of the low order 16 bits of the system's millisecond clock when this data was recorded.

The individual data blocks are grouped into buffers for recording. 512 word buffers are used, corresponding to two disk blocks. Since the individual data blocks are varying size, and it is not permitted to break a block between buffers, a varying amount of data will appear in each buffer. To indicate the end of the useful data a -1 is recorded in the word following the last data block. This is the location where the identifying block number for the next block would normally be found.

At the beginning of each buffer, block number 1 (a time block containing the current 32 bit system clock time) is recorded so that a full 32 bit mission time is available to eliminate the ambiguity in the 16 bit times recorded with each data block. The overall buffer format is shown in Fig. 3.9-2.

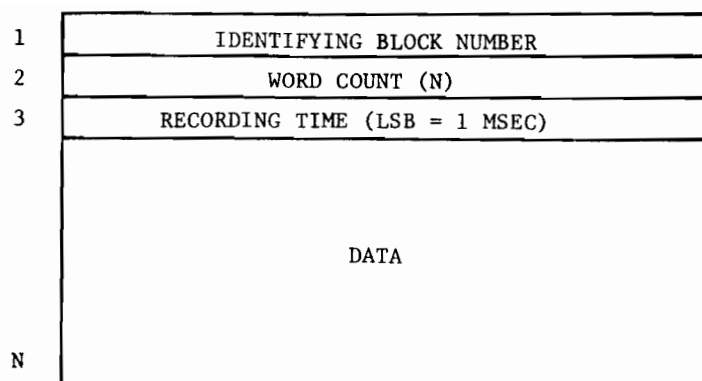


Fig.3.9-1. Data block format.

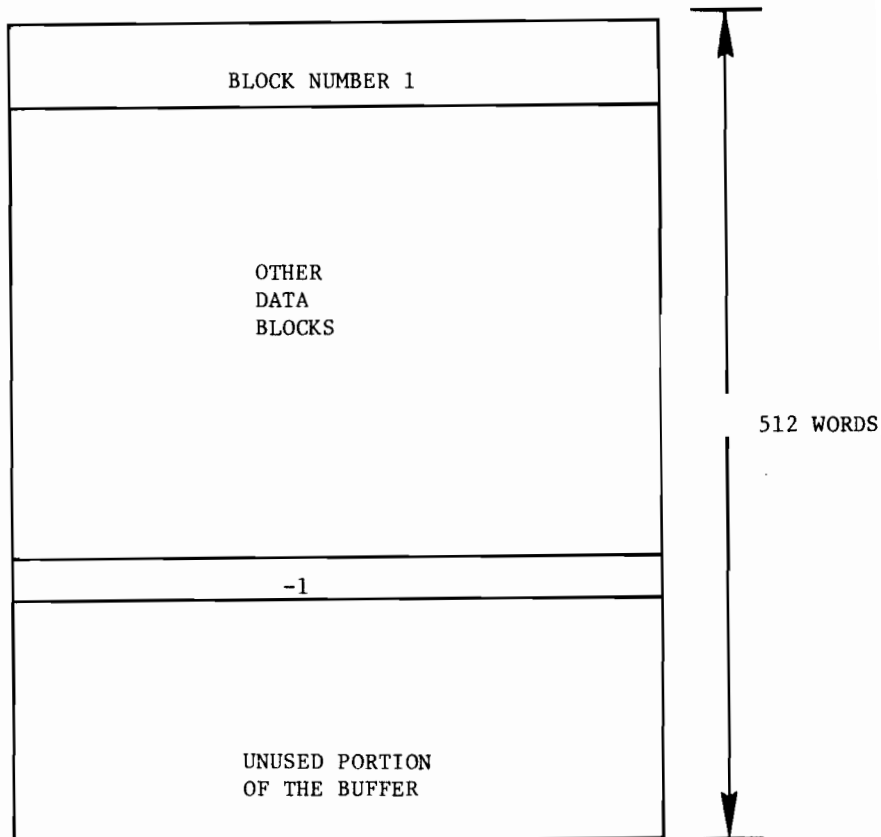


Fig.3.9-2. Recording buffer format.



Detailed data block formats for each of the data blocks recorded by ARIES are given in Appendix F.

As noted above, a utility program is provided to allow any background task to record data. Background tasks have the property that they are non-preemptive, that is, they do not interrupt one another. This is important to the operation of data recording. A task that is storing data in the recording buffer cannot be interrupted by another task which records data, as this could result in one data block being inserted into the middle of another, making the recording unreadable. However, the interrogation and reply data that is to be recorded is generated by Interrogation Processing and ATCRBS/All-Call Reply Generation, which are interrupt handlers. Since they cannot be allowed to directly record the data they instead store it in buffer areas. This data is then later picked up and recorded by background tasks which were specifically created for this purpose.

### 3.10 System Tasks and Task Scheduling

The previous discussions of the system's operation have been largely in terms of major functional units. However, these units also tend to be independent in their timing and can therefore be scheduled independently and treated as tasks. Thus, logically, ARIES consists of quite a few independent tasks, the most important of which have been discussed in the preceding sections. In fact, these tasks are scheduled by a variety of means. The highest priority, most time critical tasks, such as Interrogation Processing and ATCRBS/All-Call Reply Generation, are scheduled by means of a priority interrupt system. Also included as "tasks" under this scheme are handlers for all the ARIES hardware interrupts. All of these except the above two and the real time clock handler are quite short and just increment counters or read small amounts of hardware status data.

The next group of tasks are treated as tasks by the Data General RTOS operating system, and are priority scheduled in such a way that higher priority tasks can take away the CPU from lower priority tasks (just as higher priority interrupts can preempt lower priority ones). Of these there are only a small number, as the operating system requires a considerable amount of memory overhead (in the form of task control blocks and per-task stack space) for such tasks. Included in this group are Disk Input, Data Output, Operator Communications, and System Initialization, in order of decreasing priority.

The next group of "tasks" are in fact treated as subroutines of System Initialization in the portion of that task which forms the system idle loop. Each such "task" has a status flag which, if set, will cause it to be called on the next cycle through the idle loop (see Section 3.8). The idle loop only goes back to its 1 msec delay counter when all such flags have been cleared. The order in which the flags are tested determines the priorities. After completing any task, all the flags are again tested. Many of these

are set by the real time clock handler when the time for a given task to run has arrived. Others are set when either Interrogation Processing or ATCRBS/All-Call Reply Generation has created some data to be recorded and wishes to wake the corresponding data recording "task" (see Section 3.9). These "tasks" are not able to preempt each other (due to the fact that they are treated as subroutines) and are collectively called the "background tasks". They include the following functions, in priority order:

- DABS Interrogation Data Recording
- ATCRBS/All-Call Data Recording
- Pre-Interrogation Processing
- Traffic Model Input
- Adjacent ARIES Input
- Timer Countdown

There are two major advantages to this form of scheduling. One is that the scheduling overhead is greatly reduced, both in terms of CPU time and in terms of memory requirements. Second, since these tasks are non-preemptive, there is no need to worry about non-deterministic interactions on data structures shared between two tasks. Thus the coding is simplified and there is less likelihood of obscure timing related "bugs". However, data structures shared between these background tasks and the other system tasks and interrupt handlers must be studied carefully for unsafe interactions and care taken to avoid them. In particular, the sections of this document discussing Pre-Interrogation Processing, Traffic Model Input, and Timer Countdown present details designed to prevent unsafe interactions with Interrogation Processing on the several files shared by these functions.

## REFERENCES

1. P.R. Drouilhet, "DABS: A System Description", Project Report ATC-42, Lincoln Laboratory, M.I.T. (18 November 1974), (FAA-RD-74-189), DDC AD-A005056/9.
2. D. Karp, and M.L. Wood, "DABS Monopulse Summary", Project Report ATC-72, Lincoln Laboratory, M.I.T. (4 February 1977), (FAA-RD-76-219), DDC AD-A038157/4.
3. Provisional Signal Formats for the Discrete Address Beacon System (Revision 1)", Project Report ATC-30, Rev. 1, Lincoln Laboratory, M.I.T. (25 April 1974), (FAA-RD-74-62), DDC AD-770052/9, specifies the signal formats and protocols under which ARIES operates.
4. J. Welch, and P. Robeck, "Proposed Technical Characteristics for the Discrete Address Beacon System (DABS)", Project Report ATC-71, Lincoln Laboratory, M.I.T. (30 September 1977), (FAA-RD-77-143), DDC AD-A048246/3. This contains modified formats and protocols adopted since FAA-RD-74-62 was published. ARIES does not abide by this specification insofar as it differs from FAA-RD-74-62.
5. J.T. Barrows, "DABS Downlink Coding", Project Report ATC-48, Lincoln Laboratory, M.I.T. (12 September 1975), (FAA-RD-75-61), DDC AD-A016356/8.
6. J.T. Barrows, "DABS Uplink Coding", Project Report ATC-49, Lincoln Laboratory M.I.T. (25 July 1975), (FAA-RD-75-62), DDC AD-A014915/3.
7. D. Reiner, and H.F. Vandevenne, "Provisional Message Formats for DABS/NAS Interface (Revision 1)", Project Report ATC-33, Rev. 1, Lincoln Laboratory, M.I.T. (10 October 1974), (FAA-RD-74-63A), DDC AD-A000257/6.
8. S. Cohen, and F. Maginnis, "Extrapolation Methodology Used in the Los Angeles Basin Standard Traffic Model", (MTR-6386), (April 1973) FAA-RD-73-86.
9. "Programmers Reference Manual, Eclipse Computer", Data General Corp. 015-000024.
10. "Interface Designer's Reference, Nova and Eclipse Line Computers", Data General Corp. 015-000031.
11. "Eclipse-Line Real Time Operating System", Data General Corp. 093-000135.
12. J.B. Peatman, The Design of Digital Systems, (McGraw-Hill, New York, 1972).