1985

A Computer Graphics Head-Up Display for Air-To-Air and Air-To-Ground Flight Simulation

Daryl R. Mair
University of Central Florida

A COMPUTER GRAPHICS HEAD-UP DISPLAY FOR AIR-TO-AIR AND AIR-TO-GROUND FLIGHT SIMULATION

BY

DARYL R. MAIR
B.S., University of South Florida, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1985

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

A computer graphics simulation of an aircraft Head-Up Display was designed using an RDS-3000 Ikonas Graphics Processor and a PDP-11/34 host computer system. The software control and display modules were accomplished using Ikonas microcode and Digital Equipment Corporation Fortran IV-PLUS. The Head-Up Display system consists of the basic flight data, which includes aerodynamic flight information, Roll/Pitch Ladder, and the Velocity Vector or Flight Path Marker. The system was designed for flexibility in modifications and evaluation of various weapons delivery systems. These will be adapted to specific needs by research scientists and engineers at the Visual Technology Research Simulator in Orlando, Florida.

INTRODUCTION

The Head Up Display (HUD) is now used in almost all of the United States aircraft involved in Air-to-Air or Air-to-Ground combat. This capability allows the fighter pilot to concentrate on a target, without having to look down to see the instrument readings. At the same time it provides helpful information on the location of the target and calculates an accurate weapon release time for the pilot.

The flight simulator at Visual Technology Research Simulator (VTRS), Naval Training Equipment Center, Herndon Annex, Orlando, Florida, is presently configured as a T2C aircraft with the flexibility to simulate other trainer aircraft as well. The simulator's system configuration consists of three modules, with the HUD system being an added peripheral device as shown in Figure 1. A Systems Engineering Laboratories (SEL) computer is used as the dynamic information retrieval and update element. This system monitors and updates the information received from the aircraft cockpit and sends it to a CIG system. The Computer Image Generator (CIG) projects images of Air-to-Air and Air-to-Ground databases. These are updated at 60 Hz and

have the capabilities of simulating weapons attacks for rockets, bombs and guns. The CIG uses the dynamic data received from the SEL system to produce the image that results from the pilots actions. The image is then transported to the display module to be projected onto the concave surface of a domed screen surrounding the simulator cockpit.

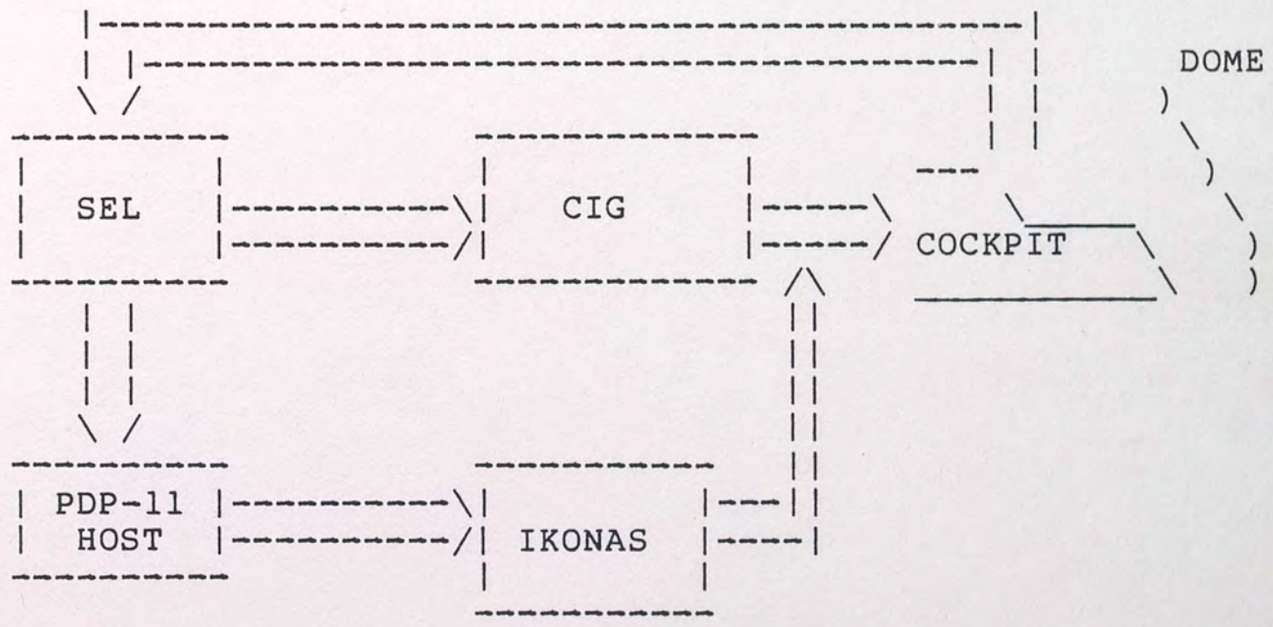


FIGURE 1: General Block diagram of VTRS system.

The SEL Computers also transfer dynamic data pertaining to the HUD to the PDP-11/34 Host Computer via an High Speed Data (HSD) interface. Once received, the host computer will use the new data to update the old, and send the updated

information to the IKONAS GRAPHICS PROCESSOR. The Ikonas will in tern produce an updated display to the HUD projector at a rate of 30 Hz.

The HUD system which consists of a PDP-11/34 Host Computer and the Ikonas Graphics Processor, is explained in detail in Chapter I. The PDP-11 uses FORTRAN FOUR PLUS and MACRO-11 compilers. Virtual addresses in memory can be defined by the Fortran Task Build procedures. Therefore this will enable the control and update portions at the host to be done in Fortran. All other specialized processing can be accomplished using Ikonas Microcode. The Ikonas is based on the AMD 2903 bit/slice architecture, and uses a multiple access Ikonas Bus. The 64 bit microcode instructions are used for Address and Operation Code processing. The Ikonas sytem is equipped with a Matrix Multiplier unit, a Character Generator, as well as image processing memory and control features. These high speed units assist the processors in the generation of objects and characters in a realtime processing environment.

Prior to the development of the HUD, the flight simulator has made use of an Optical Gunsight which when projected, is a static display with hash marks indicating the Milliradians of depression. The Mill depression is used by the pilot as a means of predicting the trajectory that a certain weapon will take. Once the aircraft is in line with the proper Mill depression, then the weapon can be released.

This system is a primitive method and the need for a more accurate and reliable weapons system is evident.

Figure 2 is an illustration of the Basic Flight Control HUD in standard flight. The basic flight data of the HUD consists of three modules:

1. Generation of dynamic flight control data.
2. The Flight Path Roll/Pitch Ladder.
3. Update of the Aircraft Velocity Vector.

The flight control data consists of an alphanumeric display of the Heading, Airspeed, and Altitude located in boxes in the upper quadrant of the HUD display. The Aircraft G, Angle of Attack, Mach Number, and Rate of Climb/Descent are located on the bottom sides. The Roll/Pitch ladder is a dynamic display of the aircraft's viewpoint to the horizon, and gives the flight path in relation to the Velocity Vector. The Velocity Vector or flight path marker is the direction which the aircraft is taking while in flight. The coordination of these three modules basically allows the pilot to fly the aircraft in an upright position only having to look down to the instruments in the cockpit seldomly. From this configuration a number of different weapon status and delivery displays can be constructed. These features are discussed further in Chapter V.

The HUD is designed for a realtime environment, in which the system must be synchronized with the other

functions of the simulator. This requires an analysis of timing as well as a knowledge of the processes of the simulator. The following is a list of the goals included in this research project.

1. Implementation of the Basic Flight HUD as described previously.

2. The design of software for a Host Control Module to:

    1. Receive dynamic flight control information from the SEL 32/75 via a High Speed Data (HSD) interface.

    2. Convert data into visual update and coefficient matrix information.

    3. Send the data and control information to the locations of the Ikonas Processor Memory.

    4. Control execution and synchronization of Ikonas microcode.

3. Develop Ikonas Microcode For:

    1. An executive module to call other microcode modules and to synchronize with the Host Control Module.

    2. A microcode program to create a window to extract portion of roll/pitch ladder to be displayed on the HUD.

    3. Make any modifications to other microprograms to be used.

4. Discuss various forms of weapons control functions.

System requirements, as well as aeronautical specifications must be met, and a detailed definition of

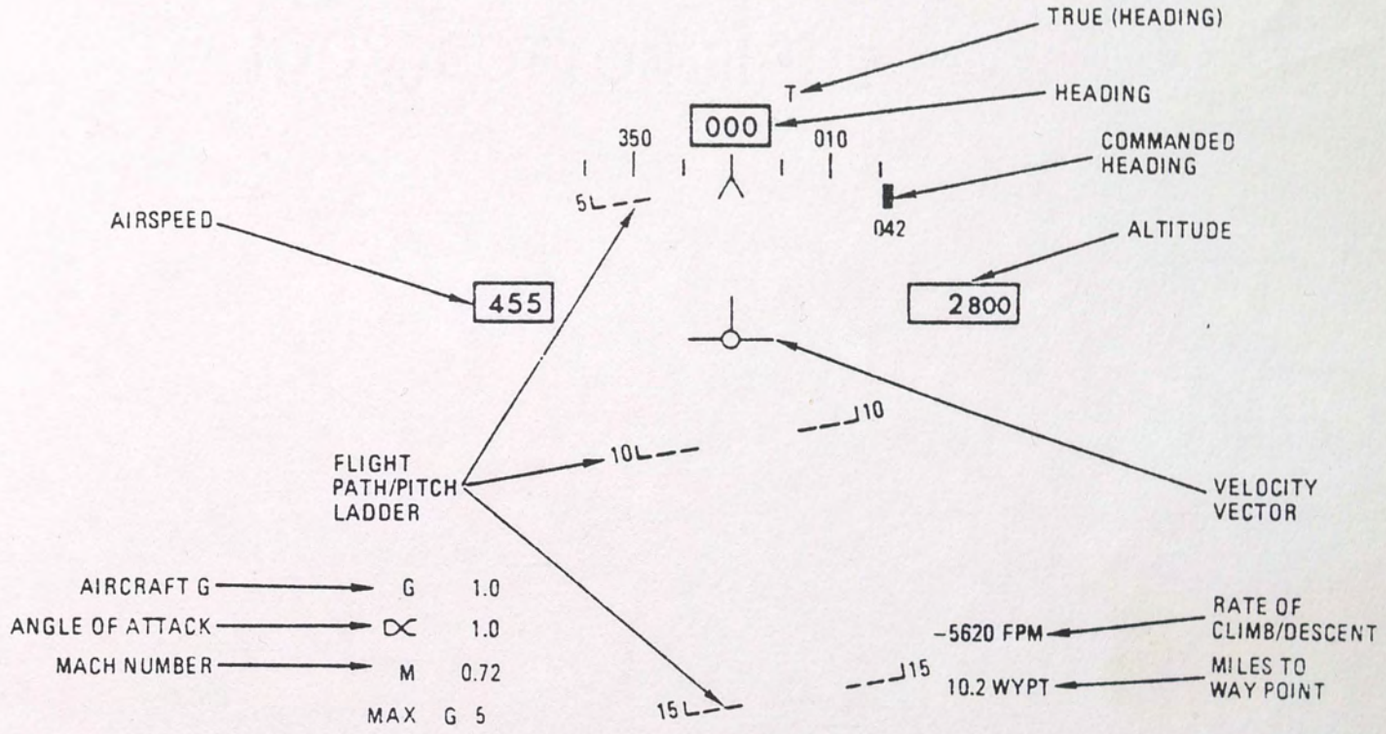


FIGURE 2: Basic Flight HUD

each element of the HUD must be made. The Timing Analysis includes calculation of matrix multiplication rates, and the time required for dynamic character updates and output to the display. These must be synchronized to produce a 30 Hz update rate.

The following chapters in this paper will describe in detail the methods and decisions which were made in the development of the HUD. Chapter I describes the Ikonas Graphics Processor System and its requirements. Chapter II is an explanation of the HUD functions and specification requirements. Chapters III and IV are devoted to describing the firmware and software development, Chapter III explains the Ikonas Microcode and Chapter IV is a detailed description of the host programming. Chapter V is devoted to a discussion of weapons status and release algorithms and their uses. The Results and Conclusions are presented as Chapter VI.

CHAPTER I

IKONAS GRAPHICS SYSTEM

The IKONAS RDS-3000 Graphics System is a 32 bit 200 nsec cycle bipolar processor consisting of several processing units. These all share one common bus called the IKONAS BUS, which is a 100 nsec, 32 bit data, and 24 bit address lines. Figure 3 is a block diagram of the Ikonas System configuration (Ikonas 1980). The BPS32 processor is controlled by 64 bit microcode instructions stored in static memory. The system can be programmed to diplay a variety of graphics applications and can update displays in Image Memory at variable rates of up to 60Hz. The Ikonas Graphics System also includes a PDP-11/34 Host computer, which is interfaced with the Ikonas using a DR11B interface board to the Ikonas Host Interface. This gives the system an external processing power as well as control capabilities from two separate units.

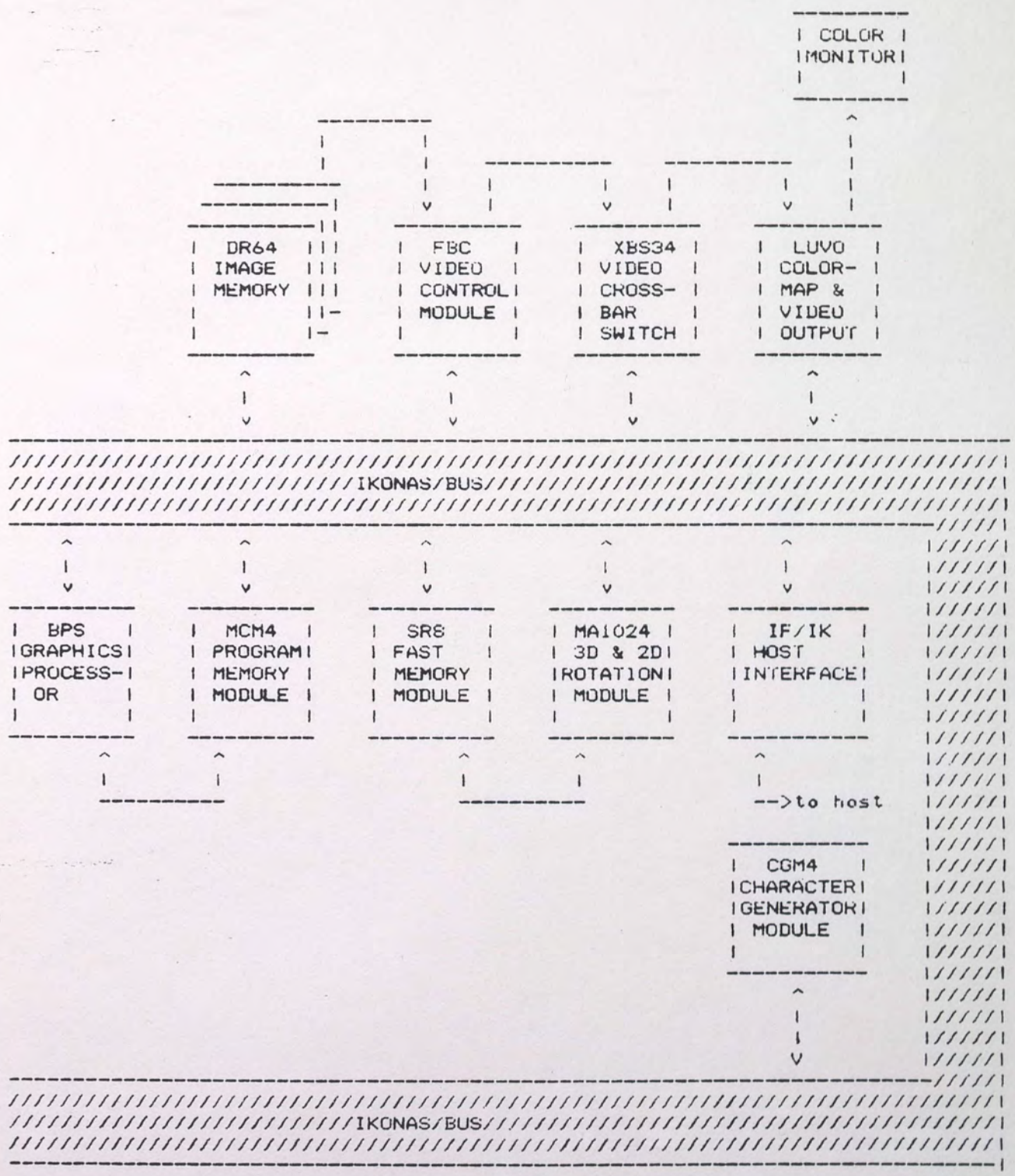


FIGURE 3: Ikonas Processor, Multiplier, Character Generator, Host Interface and Video Output Modules

1.1. Hardware Overview

The processing units for the RDS-3000 system can be isolated into five modules. The BPS32 Processor/Sequencer, MA1024 Matrix Multiplier, CGM4 Character Generator, Host Interface, and the Video Output Chain Modules. These modules all share the Ikonas Bus and require control circuitry or microcode control to interact with each other. The PDP-11/34 Host Computer can also gain access and control of the Ikonas Bus via the Host Interface.

1.1.1 BPS32 Processor/Sequencer

The BPS32 processor and sequencer are based on Advanced Micro Devices Am2903 bipolar bit slice processor and the Am2911 bit slice microprogram sequencer (AMD, 1981). The Ikonas Processor operates with 32 bit registers, and 64 bit microcode instructions. The microcode is used as control and address data for both sequencer and processor modules. Each microcode instruction executes at 200 nsec/cycle, and they can be stored in static RAM microcode memory or in ROM memory. The 64 bit microcode word is separated in two parts. Thirty two bits are used by the MPS 16 Sequencer for next address and immediate data selection. The other 32 bits determine operation selection and ALU functions.

The BMP32 is the processor unit, and carries out the functions selected from the microcode. The MPS 16 sequencer is responsible for fetching microcode instructions, determining next address originations, and selecting some operation fields to be coordinated with the processor. Together these two units function as one module, and after a hardware evaluation of each, will be treated as such.

1.1.1.1 MPS 16 Sequencer

The MPS 16 sequencer consists of a multiplexer bit slice sequencer (Am 2911) and some control circuitry. Figure 4 is a block diagram of the MPS 16 sequencer. The Data Field Register and Microcode Latch are inputs for microcode data to the sequencer unit. From this the specific field(s) are selected. These fields may consist of next address, Ikonas Bus functions, or Data Field direction operation codes. The fields which can be activated are as follows:

OP CODE: The Op Code function selects an appropriate PROM CONTROL address, which determines the path from which the next microcode address will come from. The address can originate from one of four inputs, the data field, program counter, the on chip four word stack pointer, or a latch delay from the data field. When a path is chosen, that line

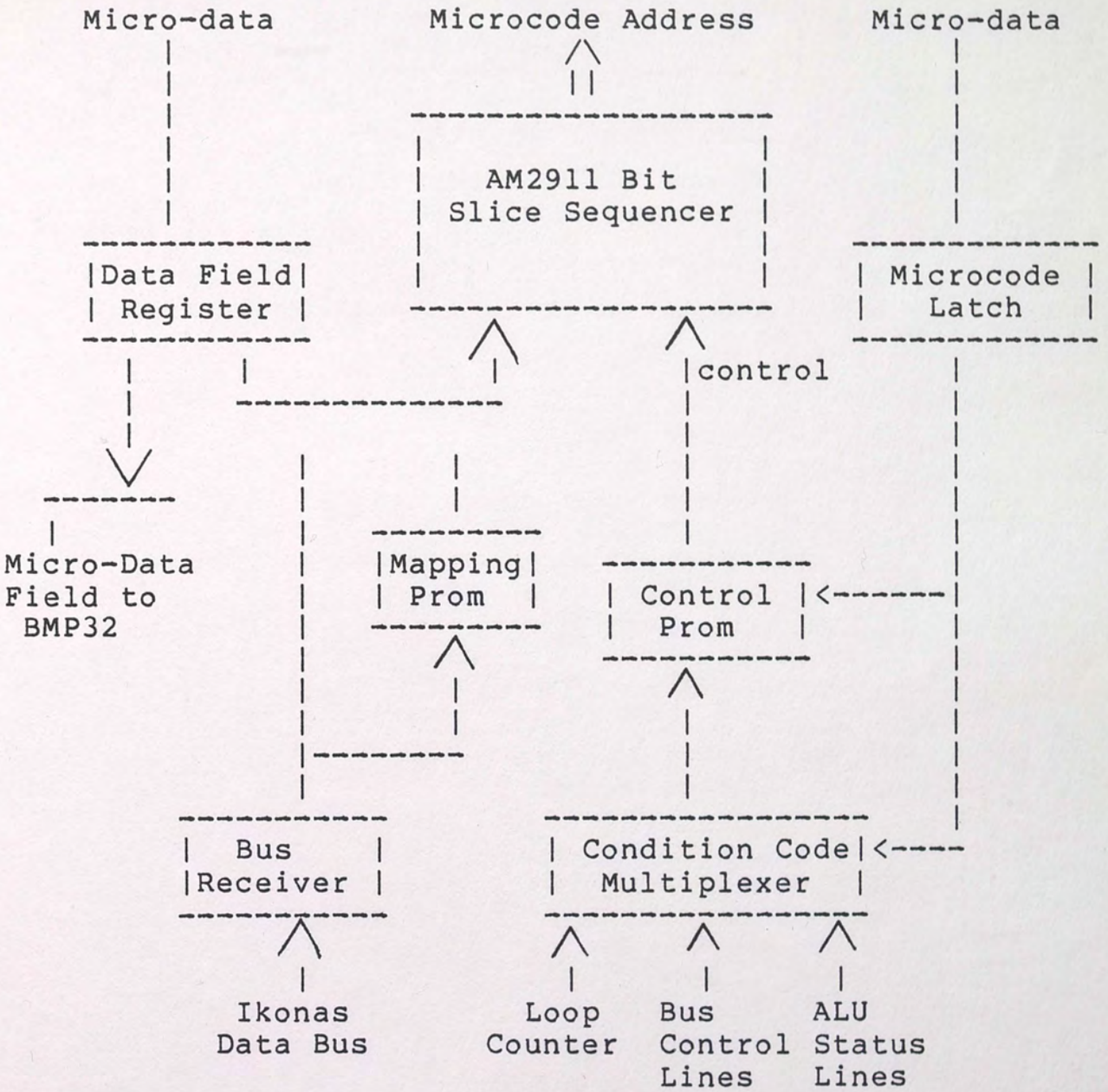


FIGURE 4: MPS16 Microsequencer

contains the 16 bit address of the next microcode instruction to be fetched and executed. Using the Am2911 allows loops and subroutines to be executed with no overhead, and the Control Prom has the flexibility of being programmable. The other 32 bits of microcode data is latched to the control portions of the BMP32, and 16 bits of the microsequencer data can be selected to pass as data to the microprogram Data Field Register on the processor.

DATA FIELD: This operation directs one of the three paths onto the data field, depending of the OP CODE and LOAD CONTROL bits selected. It can be sent as immediate data to the BMP32 processor, used as a loop counter for the sequencer, or as a pipeline address for the next microcode instruction.

CONDITION CODE SELECT: The condition codes allow the programmer to execute various OP Codes depending on conditional executions. Options presently implemented are: Host Request High, Matrix Multiply Busy, Image Memory Busy, ALU Overflow, ALU Negative, ALU Zero, and ALU Carryout. The condition code multiplexer is designed such that others can be introduced with very little difficulty. Condition parity can also be selected as to whether an operation takes place on a condition TRUE or FALSE.

LOAD CONTROL: The Load Control determines if the 16 bits of microsequencer data is latched to the processor data register or used as a loop counter for the MPS 16.

IKONAS BUS FUNCTION CODE: This code determines the type of transfer and destination of data to or from Static (32 bit parallel) or Dynamic (1Ø24X1Ø24) RAM, the Matrix Multiplier, or other special purpose devices.

1.1.1.2 BMP32 PROCESSOR

The BMP32 processor consists of sixteen 32 bit general purpose registers, two selectors and shifters as part of the Am 29Ø3 processor, an R bus selector, the Y bus, MDR, MAR, Data Register, and connection circuitry to the Ikonas Bus Data and Address lines. A block diagram of the BMP32 processor and its control circuitry is shown in Figure 5. The Data Registers consist of dual port RAM, such that addresses can be selected from two ports simultaneously and data transmitted thru two address ports A and B independent of each other. Data is written to the registers only through address port B.

The 32 bit microcode instruction that are input to the processor, are used to determine several fields.

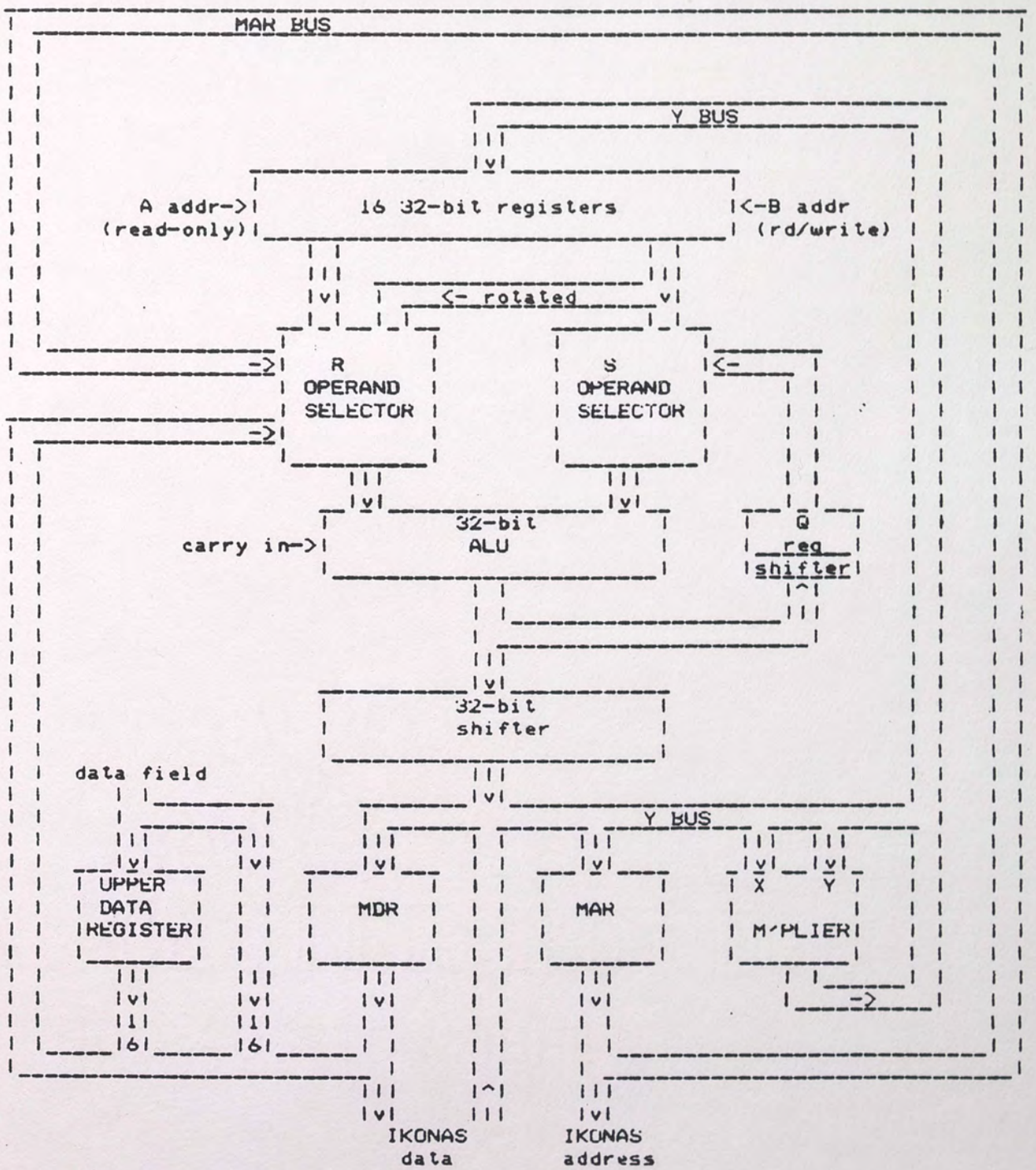


FIGURE 5: BMP32 Processor

1. R and S Operand Select
2. ALU Instructions
3. Carry and Shift Control
4. Y Bus Source/Destination
5. Data Register Bus Source
6. Ikonas Bus Enable/Address Source

All of these fields are selected using different parts of the 32 bits of microcode instruction. The format and execution tables for these fields will be further discussed in Chapter III.

S OPERAND SELECT: The S Operand can be selected as a Y Bus output destination or any of the 16 data registers. The field can also select the Q register as the S Operand.

R OPERAND SELECT: This field can select any of the 16 data registers or the R Bus as the R operand. Using the R Bus there are a few paths which data can take, to the R operand. There is the Data Register, transferring both high and low addresses. The R Operand can also be loaded from MAR, and data from the B register in several forms. These are selected by the ALU FUNCTION field.

ALU FUNCTION: The Arithmetic Logic Unit on the Am 29Ø3 is capable of performing nine logical and seven arithmetic

operations on both 32 bit operands. The selection of the Q shifter and Q register are also a part of the ALU Function. There are special functions which the ALU can select as well, such as multiplication, division and normalization. Selection of shifts, either arithmetic or logical are among some of the other functions. These can be determined to be left or right shifts, as well as determining the type of shift required. The write control for the B and Q registers are also selected from this field.

ALU SHIFT AND CARRY CONTROL: Though the direction and type of ALU shift is controlled by the ALU Function code, the selection of either a shift or rotate is made in the Shift Control Field. A shift can be selected as a long shift, 64 bits of both the ALU and Q shifter combined to form one, or short shift, each being used as separate 32 bit shifters. The Carry Control Field is used to select whether operands are to be treated as two 16 bit words, or one 32 bit word. In the 16 bit form two carry bits can be set or cleared, bit Ø and 16. In the 32 bit case only one carry bit is available but has the flexibility of Ø, 1, bit 31, or bit 31 inverted.

YBUS SOURCE/DESTINATION: The Y bus is used as a communication link between the Am 29Ø3 and the other parts of the BMP32 processor. Inputs to the Y bus can come from

Figura and McDuffie's Model

This speciation scheme was developed in 1980 and divided soluble heavy metals in environmental water samples into four categories. Trace metal species are classified as "very labile" which includes free and hydrated metal ions, "moderately labile," "slowly labile," and "inert." The non-labile fraction includes metal bound in complexes or absorbed on colloidal material with a slow rate of dissociation to the free metal. Examples of "moderately labile" fractions might be Cd-NTA (Cd-nitrilotriacetate) or Cu-humate. Also, some examples of "slowly labile" fractions might be Cd-, Pb-, and Zn-EDTA (Zn-ethylenediaminetetraacetic acid) complexes.

Figura and McDuffie (1980) applied their scheme to St. Lawrence and Susquehanna River waters and Hudson River Estuary samples. The estuary samples were distinctly higher in Cd, Cu, Pb, and Zn content than samples from the St. Lawrence and Susquehanna rivers. The soluble lead represented only 11 to 46 percent of the total metal. Most of the Cd and Zn in all the samples existed as "very labile" or "moderately labile," with a small percentage of "slowly labile" and no "inert" fractions. In contrast, both Cu and Pb existed in forms which are less labile. Copper was found almost entirely in the "moderately

The Am 29Ø3 also has special functions which are utilized by the processors ALU. These functions are provided by Ikonas in the form of special ALU operations. These special functions are as follows.

1. UMPY-UNSIGNED MULTIPLY: Using B register for result, CARØ and SSØ are used for multiplication of unsigned numbers.

2. TCMPY-TWO'S COMPLEMENT MULTIPLY: The result also in B register, using CARØ and LRØ for multiplication of two's compliment numbers.

3. INCRS-INCREMENT OPERANDS: the S operand plus 1 plus the Carry is placed in the B register.

4. SMTC-SIGN/MAGNITUDE TO TWO'S COMPLEMENT: Converts S operand from sign/magnitude to two's complement using CARZ and written to the B register.

5. TCMPL-TWO'S COMPLEMENT MULTIPLY LAST CYCLE: Two's complement multiply of last step using RSØ and CARZ, is written to the B register.

6. SLNML-SINGLE LENGTH NORMALIZE: Used to normalize the Q register with CAR1, SSØ and CCOVR bits. The S operand is incremented and sent to the B register.

7. DLNML-DOUBLE LENGTH NORMALIZE: The Double length normalize uses CARØ, LSØ and CCOVR to normalize the Ø register and S operand. Also used for first step in division with CARØ and LR.

8. TCDIV-TWO'S COMPLEMENT DIVIDE: Uses CARZ and LR for two's complement divide operation.

9. TCDIVC-TWO'S COMPLEMENT DIVIDE CORRECTION: The division correction takes place on the final step and uses CARZ and SS1 bits.

These functions as well as the other operation codes for the BPS32 microcode are presented as tables in Appendix A. The name of the function and the bit numbers in

microcode it represents are given. This allows the programmer to construct microprograms with a one-to-one correlation between instruction and microcode.

1.1.2 MA1Ø24 MULTIPLIER

The MA1Ø24 Matrix Multiplier is a 64 bit microprogramable, high speed multiplier, capable of processing 3ØØ,ØØØ multiply operations per second. It is a 2ØØ nsec, 16X16 bit multiply and 35 bit Add/Subtract accumulator. The 1Ø24X16 on-card memory buffer is used to store transformation arrays. The multiplier has direct access to the dual port 1ØØ nsec 32 bit static RAM module (SR8). Three dimensional transformations can be done with less than a microsecond execution time. The multiplier/accumulator is microprogrammable and has a control module to control access of the Ikonas Bus lines and static RAM modules.

There are several locations to which information can be written to or from within the MA1Ø24 matrix multiplier. Using the Ikonas Bus, information can be written to the MA1Ø24 at several locations: the 1Ø24X16 Coefficient Memory (CM), 1Ø24X32 Microprogram Memory (MPM), or one of three Control Words. Control Word Ø is made up of the microprogram memory starting address which is eight bits, the microprogram memory page of two bits, coefficient memory

starting address with eight bits, and coefficient memory page two bits. Control Word 1 consists of the static RAM input and output address both fifteen bits in length. Control Word 2 is for loop counter data and is twelve bits. To read from the MA1Ø24 the following locations can be accessed. Coefficient Memory (CM), the thirty two bit XY and ZS Product Latches, and sixteen bit W. The Auxillary Bus Interface allows access of two to the fifteenth memory locations in static RAM by the MA1Ø24. RAM addresses are loaded by the microcode control from the Ikonas Bus. An Auxillary Bus READ or WRITE is controlled by the static RAM Read/Write Line (SRW-L). A write will send data to the XY or ZS input latches, and a read will send it through the XY or ZS product latches. All of these latches are thirty two bits, and receive the address information from counters which get incremented by microcode control. This enables the multiplier to operate with full parallel processing, and does not require access to the Ikonas Bus to run.

The 4X4 coefficient matrix controls the rotation of a graphics image of a three-dimensional object. The coefficient matrix can be loaded from the Ikonas Bus into the coefficient memory of the MA1Ø24. The basic form in which each point is multiplied with the coefficient matrix and represented as the rotated output points, is shown in Figure 6 (Rogers 1976).

$$[\ Xo\ Yo\ Zo\ 1\ ] = [\ Xi\ Yi\ Zi\ 1\ ] \begin{bmatrix} Cxx & Cxy & Cxz & 0 \\ Cyx & Cyy & Cyz & 0 \\ Czx & Czy & Czz & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$

FIGURE 6: Coefficient Matrix Equation

Equations for each coefficient were determined using a method for transformation of three-dimentional objects (Newman 1979). The first column of the matrix is devoted to calculating the YAW diffraction, the second calculates the PITCH, and the third is for the ROLL.

Figure 7 is a diagram of the path of flow control in which the MA1Ø24 multiplier follows. The correct procedure for loading control information, data and execution of the MA1Ø24 processor is as follows.

1. Load the correct microcode routine into the MA1Ø24 program memory. The standard rotation microprogram supplied by Ikonas is MMEOL.MOB. This program is used for this application, though the user can write other microcode programs for the MA1Ø24 if desired. The microcode can be loaded from the PDP-11/34 HOST COMPUTER using the system task called the Online Debugging Tool (ODT).

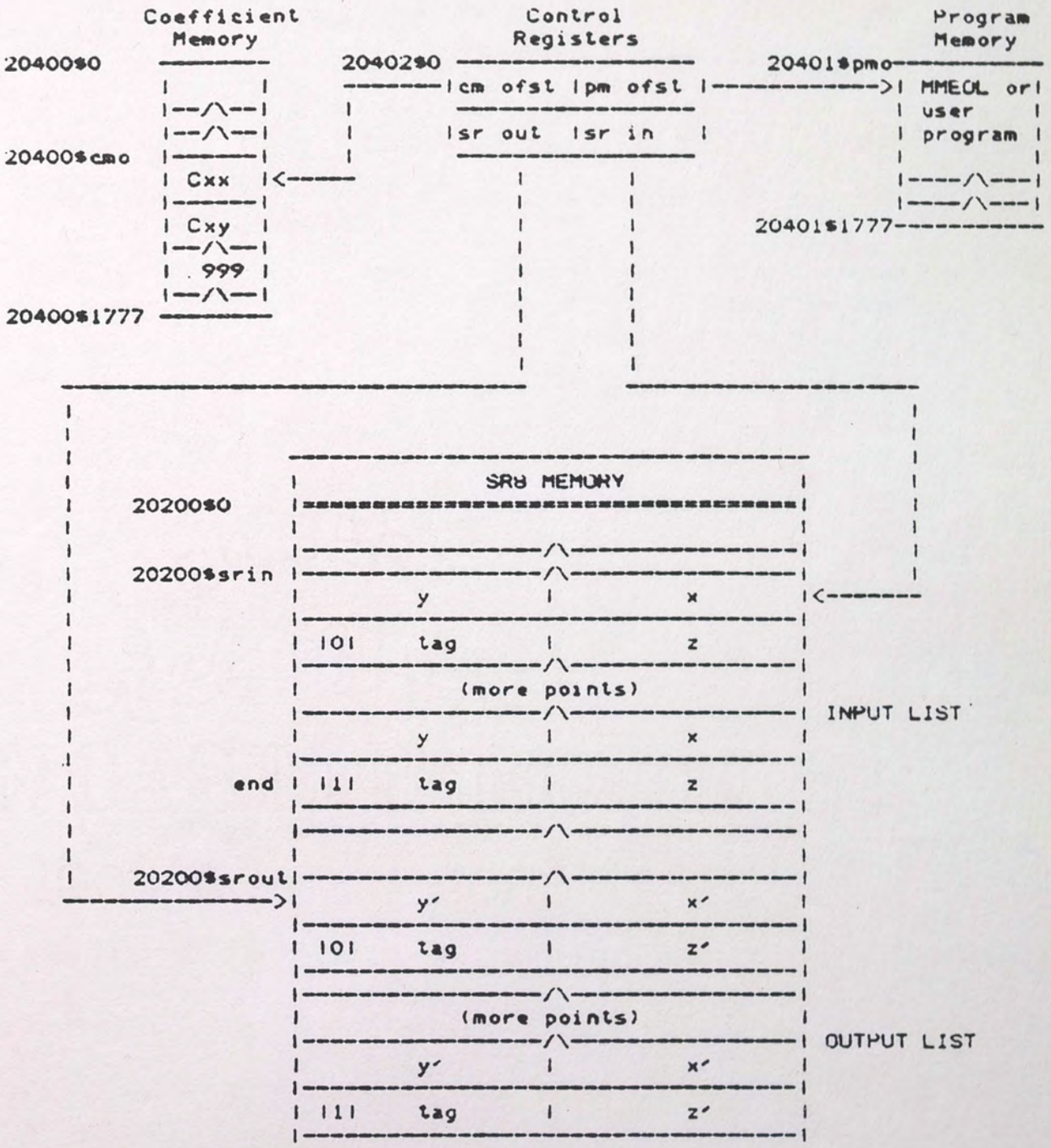


FIGURE 7: MA1024 Control Flow

2. Load data to be transformed into the static RAM memory (SR8), terminating the sequence with bit 30 set on the last word.

3. Load the coefficient matrix into coefficient memory.

4. Send control data to the MA1024 at address 20402$0. The following data to be stored at these locations.
   1. MA1024 microprogram starting address
   2. coefficient matrix address (offset from 0)
   3. Data input address
   4. Data output address

5. Start the Multiplier by storing 0 at address 20402$3 using the function sending code 25, if starting from the host computer.

Figure 8 shows the format for the control data sent to the matrix multiplier. This procedure is only for the standard case. If the programmer wishes to use other features of the MA1024 matrix multiplier microcode then this may change (Ikonas 1980).

20402$0	Coefficient Address	Microprogram Address
20402$1	Data Output Address	Data Input Address
20402$2	--	Loop Counter (optional)
20402$3	Clear both locations	to start processor

Figure 8: MA1024 Control Address Format

1.1.3 CGM4 CHARACTER GENERATOR

The Ikonas CGM4 is a high speed character generator which is capable of writing one million pixels per second. These can be written directly to the DR64 Image memory framebuffer for output to the display. The character size can be selected by the programmer to a 32X32 pixel size. A user programmed character set or font can be used by the character generator, and is accessed through the Ikonas SR8 memory. The CGM4 comes equipped with an on-board font which gives access to a standard 7X9 pixel character set without any additional font programming. From this the character font may be magnified, or written in variable directions either up, down, left or right.

Control of the character generator is done through microcode programs and information sent from the PDP-11 Host Computer. The CGM4 requires a few things other than the font program information.

1. Must receive the address in which character strings of certain length are to be displayed.
2. The address in the font of a specified character.
3. The character string itself should be placed in SR8 memory to be looked up by the character generator.
4. The CGM4 Control Blocks, used for Mode settings, and flags, must be sent to 20600$0 and CGMCB.

The microprogram developed for the dynamic character update process on the HUD is called CGMS.MOB. This is used to control the output character strings of up to eight characters to specified pixel addresses. The character and pixel address information are updated and written to Ikonas memory by the Host Control Program. The programming techniques for the character generator are discussed further in Chapters III and IV.

There are two control blocks which are used by the CGM4, the CGM4 Base Control Block located at 20600$0, and the Auxilary Control Block (CGMCB), which is addressed by the base control block anywhere in SR8 memory. The format for CGMCB is shown in Figure 9. The character height and width are done in pixels. The Font Table Offset is the address in SR8 memory where the font table is stored. If

the on-board standard font table is to be used then this value would be zero. The character shade is the color in which the pixels representing the character would be. The background shade is for those pixels which are not written as characters, and is only recognized if the Non-Transparent mode is selected by the CGM4 base control block. The character spacing selection gives the numbers of pixel spaces to allow between characters. The starting output address is used only when all characters to be displayed are from one starting address. The microcode CGMS.MOB allows multiple strings of eight characters or less to be displayed from varying starting addresses. The character string address is the address in memory where the ASCII character strings are stored. The CGM4 base control address consists of 2 Ikonas words (32 bits), located at 20600$0. Figure 10 shows the format to the base control block.

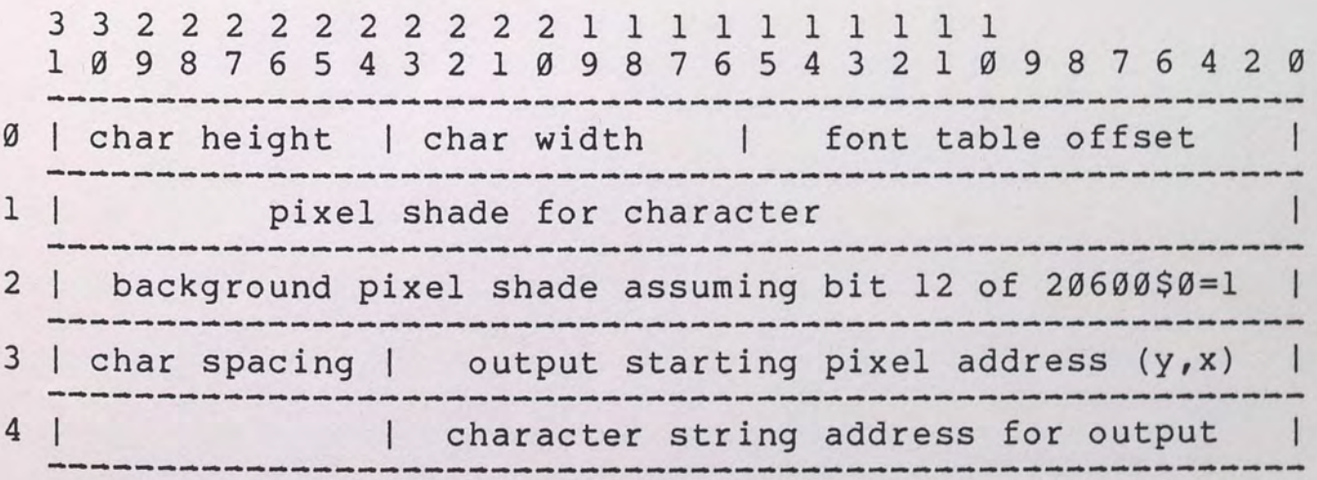


Figure 9: Format For CGMCB Control Block

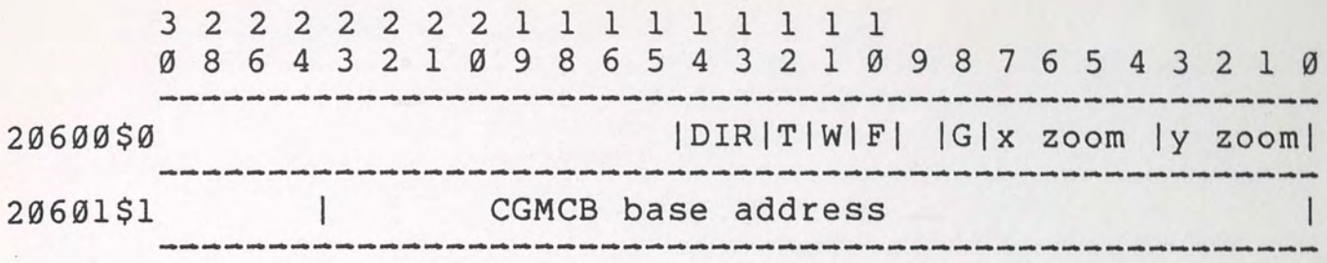


Figure 1Ø: Format For CGM4 Base Control Block

The direction can be selected as ØØ-left to right (normal), Ø1-top to bottom, 1Ø-bottom to top, and 11-right to left. The T bit is used for selecting transparent or non-transparent background shade. If set, the background shade color is written. If not the background pixels are left unchanged. The W is used to select word mode writes or pixel mode, the former is the normal case (W=Ø). The F determines whether the use of the on-board font or if a font located in SR8 memory is to be used. The G bit is the Go bit, this is set after all other control words are in place and is ready to start the character generator. The G bit will stay high as long as the CGM4 is busy. When low it becomes ready for another character to be written. The X and Y zoom are used as magnification factors in each respective direction.

1.1.4 IF/IK HOST INTERFACE

The IF/IK Host Interface has a direct DMA connection between the host PDP-11/34 computer and the Ikonas Bus. It can transfer two million bytes per second when DMA is selected. The interface also has the ability for single word transfers for smaller I/O demands. This reduces the time overhead for transferring smaller blocks of information. There are several modes of transfer which can be selected 32, 16, or 8 bit modes allows flexibility and better efficiency for smaller items. Figure 11 shows examples of the three modes of transfer that can take place either to or from the Ikonas processor. The 32 bit WORD transfer allows easy access to 32 bit data. One 32 bit Ikonas word corresponds to two 16 bit host words. The low order half of the Ikonas word represents the even host word and the odd is the high order half of the Ikonas word. The 16 bit transfer otherwise called HALFWORD mode is used when one 16 bit host word is to represent one 32 bit Ikonas word. The high order portion of the Ikonas word is left zero. 8 bit transfers become desirable in accessing single image components. Any one of the four bytes of an Ikonas word can be accessed or written to. This type transfer is referred to as BYTE mode. The Ikonas interface also supports vectored interrupts from the Ikonas to the host. The host

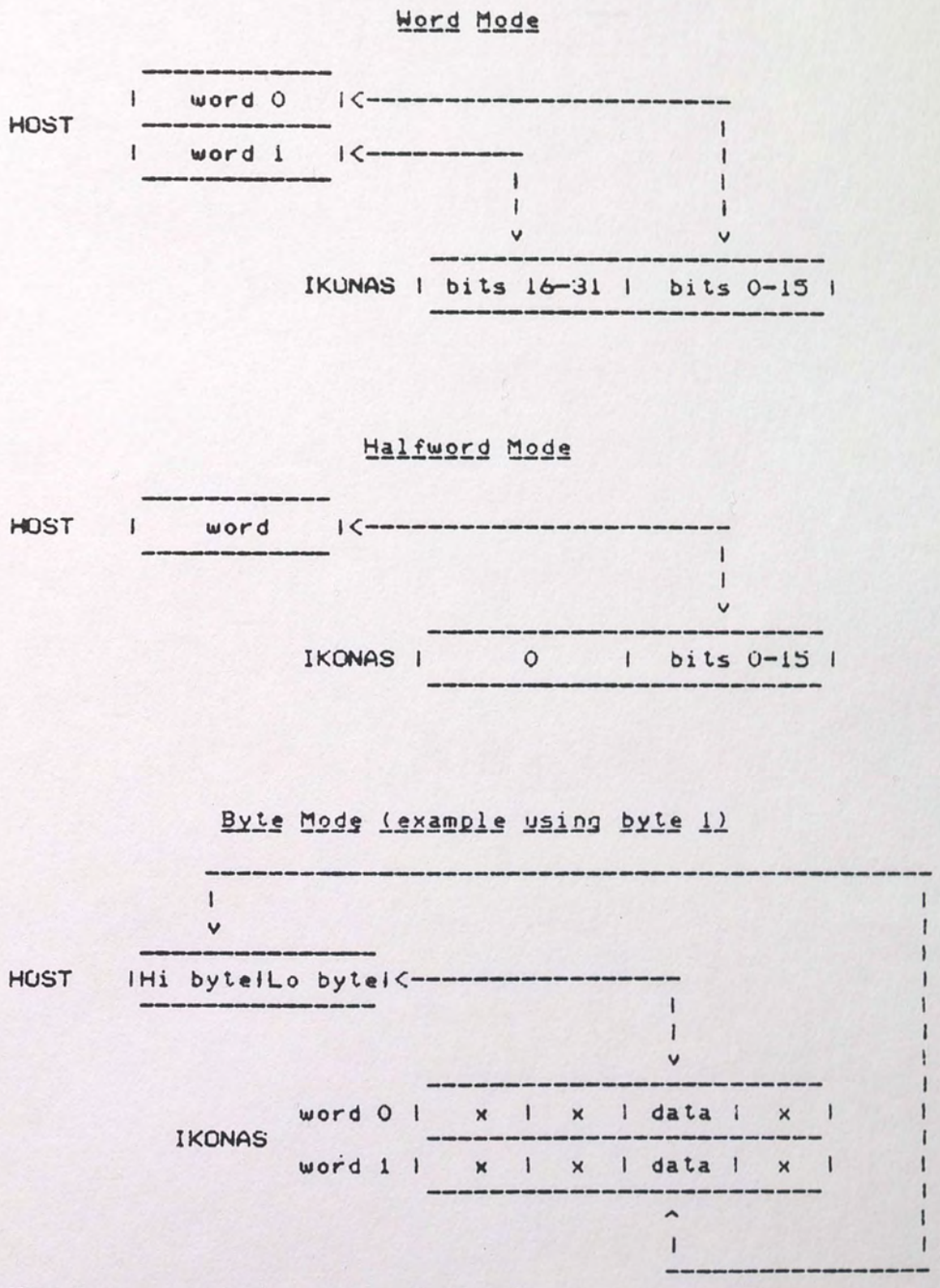


FIGURE 11: Transfer Mode Examples

can be programmed to control activities on the Ikonas Bus through the interface. An Ikonas system-reset can be accomplished from the host computer.

The PDP-11/34 is tied to the IF/IK Host Interface by a DR-11B Parallel Interface card. The DR-11B can be assigned virtual address locations which correspond to transfer and control lines at the DR-11B. In this manner control programs at the host computer can have access to the lines of the Ikonas interface. This allows transfers to be made using Fortran, and transfer control can be initiated from the Host computer.

The Ikonas Interface Control Register is defined in the host software as IKSCSR and has the format shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYT	F	P	B	R	X	+	I	D	H	function					

The function code is used to determine the type of transfer to be made. Bit 4 is the read/write line and is set for a write. Therefore the most common function code for a write is $20 and for a read $00. There are special cases when reading or writing to particular modules. For example a write to the MA1024 matrix multiplier module would be $25. The DR64 Image memory has several special function codes to distinguish resolutions and access addresses. The H bit is set for HALFWORD mode and off for WORD or BYTE mode. The D

bit is set when DMA transfer is to take place. The I bit is used for invisible I/O. The + bit is set to increment Ikonas address between words, this is usually always set. The X bit is set when the BPS32 is to execute, when this bit is cleared the processor will stop and setting it again starts the BPS32 processor where it left off. The R bit forces an Ikonas system reset. The B bit is set for BYTE mode, F is read-only and specifies the vertical blanking interval. The P bit is also read-only, when on the Ikonas is requesting service from the host. The BYT is a byte number to be selected for the byte mode transfer, and is used only when bit 11, the B bit is set.

1.1.5 VIDEO OUTPUT CHAIN

The Video Output Chain is a module in the Ikonas system which is responsible for the display of the images produced. This module can be better explained as three units. The DR 64B Image memory, the Frame Buffer Controller (FB/HC), and the Video Output. Figure 12 is a block diagram of the Video Output Chain.

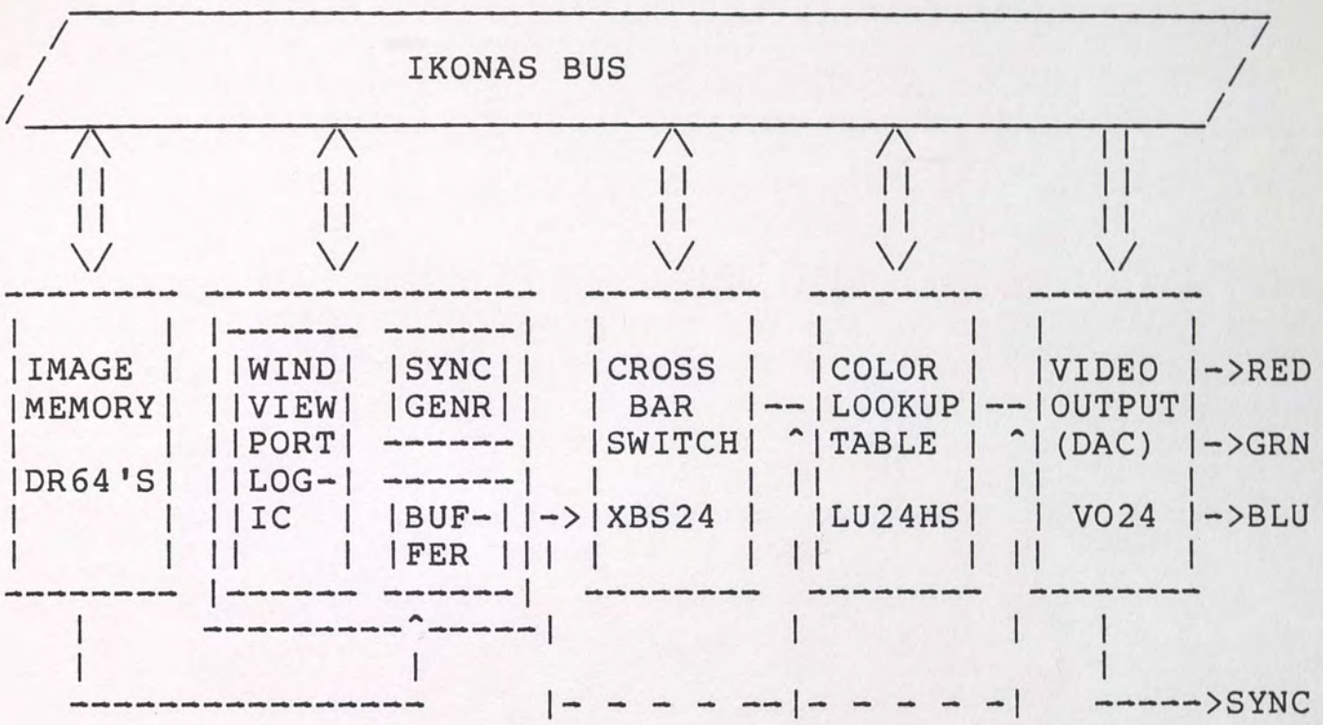


FIGURE 12: BLOCK DIAGRAM OF VIDEO OUTPUT CHAIN

1.1.5.1 DR 64B IMAGE MEMORY

The DR64 is organized in bit plane modules of 512X512 of 8 bits or 1Ø24X1Ø24 of 2 bit pixels. This allows the configuration of images which are software selectable between a low resolution (512X512) or high resolution (1Ø24X1Ø24) display. The Image memories can be configured to display up to 512X512X32 or 1Ø24X1Ø24X8 bit resolutions. The DR 64B is a dual port 128K byte dynamic RAM, with a memory access time of 3ØØ nsec. One port is used for communication with the Ikonas Bus, and the other is for video rate I/O. When a pixel is addressed to the DR64's

several of these memory devices are selected at one time. When using the low resolution called LORES, a 512X512 display image is created. Three DR64's are used to create a 24-bit color image, consisting of eight bits of red, green, and blue pixel bits. Using high resolution called HIRES a 1Ø24X1Ø24 image of four 2-bit color codes is produced. LORES and HIRES can only be selected by using the function codes as discussed in the Ikonas Interface sections. The Ikonas Bus function codes for selecting LORES is $Ø2 for a read, and $22 for a write. For HIRES the function codes are $Ø3 and $33 for read and write modes. In both cases pixels are addressed as (X,Y) coordinates. The LORES pixels are addressed in multiples of two, thus the pixel address values for X and Y, must be doubled to produce the correct output. The HIRES pixel addresses do not require a multiplication factor. The value of the X$Y coordinate is the actual pixel address.

1.1.5.2 FB/HC FRAME BUFFER CONTROLLER

The Frame Buffer Controller determines which data to display, fetches the data from the DR64, and sends it through the video output chain. This is done using the control of the following lines.

1. Aspect Ratio - The ratio of height to width of the picture display.

2. Automatic Erase - Erases the screen automatically succeeding each display.

3. Cursor - A programmable cursor which can be displayed in any shape. The cursor may be up to 32X32 bits and location must be specified.

4. Video Synchronization Rates - Gives the writing time for each line and the number of lines per field.

5. Viewport - Selects the size and position of the pixel data within the viewport.

6. Zoom - The amount of magnification which is applied to each pixel written to the display.

The data is transferred from the the DR 64 through the video output ports in parrallel up to four pixels at a time. The FB/HC then places it in a sequential format and sends it out the serial video bus to the Video Output Module.

The control block for the FBC consists of seven 32 bit registers starting at address 30000$0. Figure 13 is a diagram of the FBC control block. Registers 0 and 1 are used to determine viewport starting locations and size. The location of the window inside the viewport is established in bit 30000$2. 30000$3 controls the zoom or magnification factor, each numerical integer increases or decreases the size of the coordinate by a factor of two.

	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Y viewport start														X viewport start															
1	Y viewport size														X viewport size															
2	Y window location														X window location															
3	Y zoom														X zoom															
4	number of lines/frame														time per line															
5	pixel clock														R	3	PG		X	E			H	C						
6	Y cursor position														X cursor position															

Figure 13: FBC control Registers

The Frame Buffer Controller is responsible for setting synchronization information for the video signal. This is accomplished by controlling the number of lines per field, and the amount of time required for each line. The information is stored at address location 30000$4.

The FB/HC Flag bits are used for setting modes of operation for video output. Register 30000$5 is used for this purpose. This register is a write-only register to set flag and mode conditions. The C is set if a cursor is to be displayed. Bit H is OFF for a 512X512 LORES pixel display and ON for a 1024X1024 HIRES display. The E bit allows the DR 64's to be cleared after each field. The X bit is used if an external synchronization is to be used. If the video

sync is to be produced by the FBC this bit would be zero. PG determines the color map page. These two bit tags are placed at the end of the pixel data to be used in the video output module. The (3) function at bit 9, is set if RS-343 synchronization is required. The R is a flag to determine whether a repeat field (ON) or interlacing is preferred (OFF). The pixel clock field which is seven bits in length, determines the number of nsec/pixel in LORES and nsec/half pixel, for HIRES diplays.

1.1.5.3 VIDEO OUTPUT

The Video Output module consists of the XBS34 Crossbar Switch, Color Lookup Table, and Output to Digital to Analog Converters. Figure 14 shows the path in which data will take through the video output. In this case the crossbar switch is set up to produce a mono-colored output, thus it bypasses the color lookup table to the output. This figure also shows an example of pseudo-color operations in which the green channel is selected for video output.

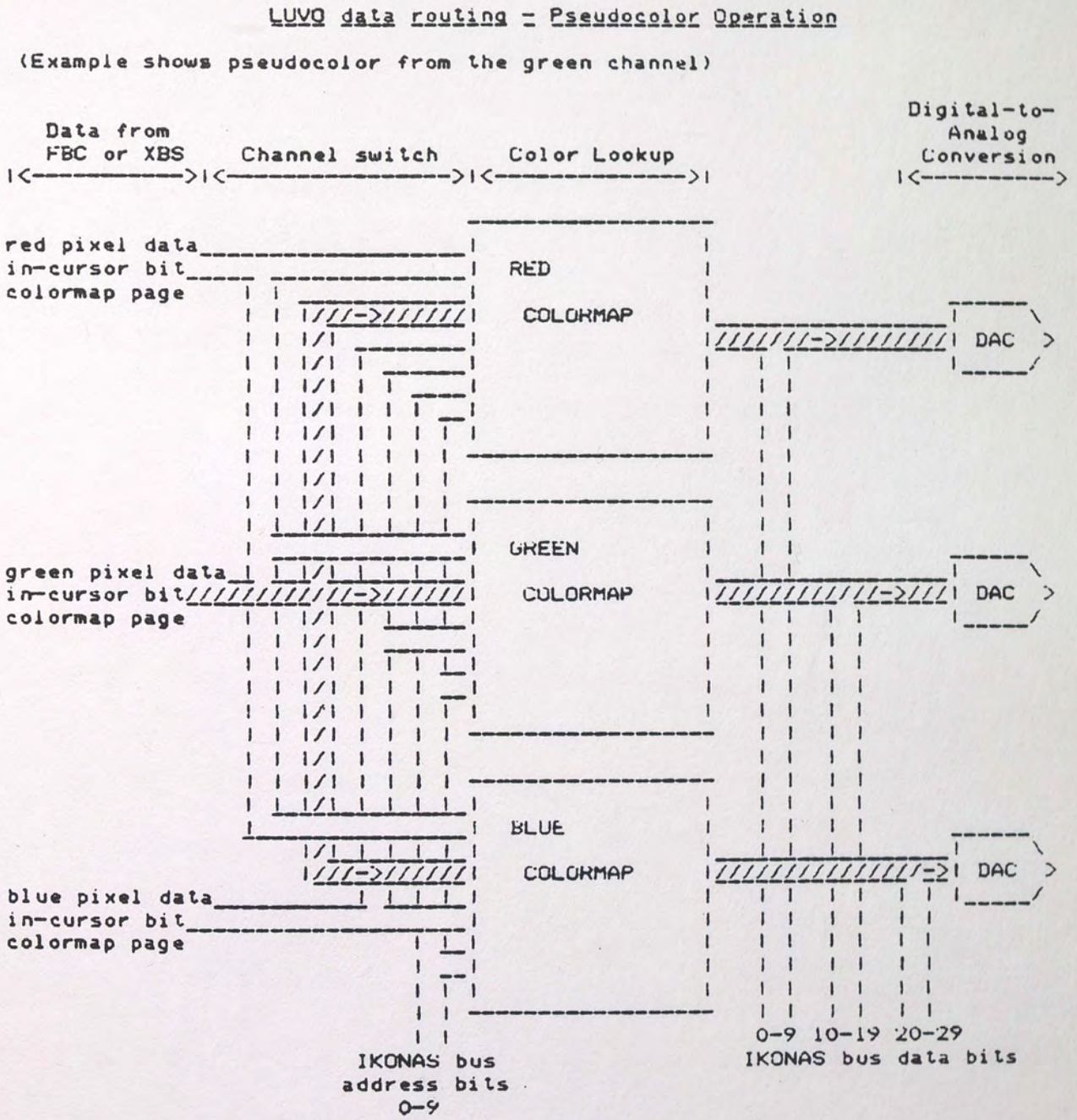


FIGURE 14: Video Output

1.2 PDP-11/34 HOST COMPUTER

The Programmable Data Processor-11/34 is a minicomputer product of Digital Equipment Corporation. Figure 15 shows the general architecture of the machine, in the form of a block structure. The processor is equipped with a unique data bus called the UNIBUS. The UNIBUS is the key to the PDP-11 families flexibility. It was the first minicomputer data bus to allow information to be sent, recieved, or exchanged without processor intervention or buffering in memory (Digital 1978). This allows interfacing of other devices through the UNIBUS to be accomplished with less difficulty.

The PDP-11 uses the RSX-11M version 4.Ø operating software system. This is a multi-tasking, realtime system which supports both development and task execution concurrently. The RSX-11M operating system has a number of user utilities which make program development and operation much easier. Tasks can be written in MACRO-11 Assembly Language or Fortran IV-PLUS (F4P).

The Host computer is interfaced using two DR-11B parrallel interface boards to both the SEL 32/75 simulation computer and the Ikonas Graphics processor. The SEL

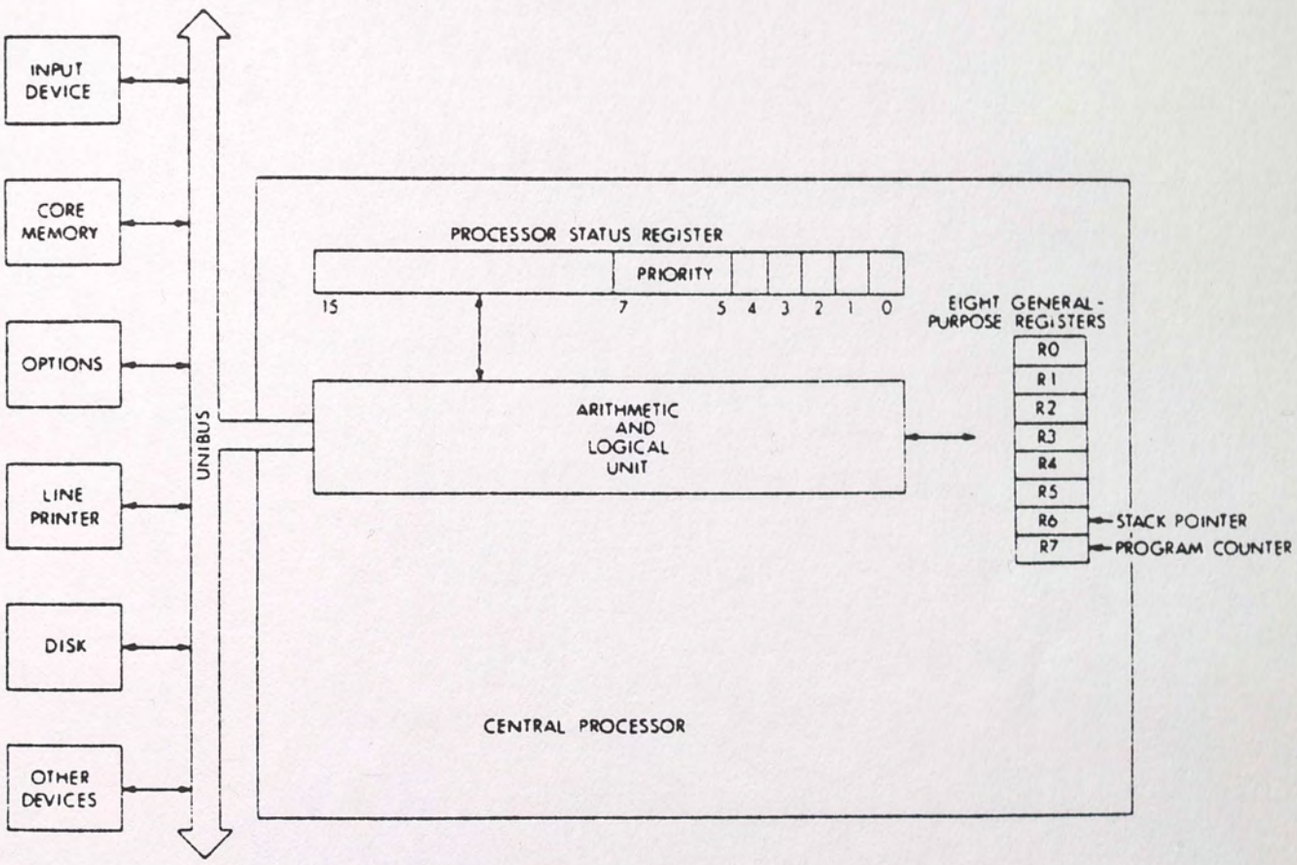


FIGURE 15: PDP-11/34 CPU

interface is an HSD line that requires a special hardware communication interface to the DR11B, which was designed and implemented by (Sampayo 1982). The interface was designed so that flight data could be transferred from the simulator to the Ikonas Host computer. The interface from the second DR11B to the Ikonas Processor is accomplished through the Ikonas Interface as described previously. This allows transfers to be made as blocks of data (DMA), or as single data types. With these configurations and control capabilities from Host software, all transfers and communications can be syncrhonized.

CHAPTER II

HEAD-UP DISPLAY

The HUD is used in Air-to-Air and Air-to-Ground fighter aircraft and is implemented in almost all of the modern aircraft of the worlds armed forces. As technology advances the levels of sophistication and detail of the flight HUD will increase, presumably until the need for conventional instrumentation becomes secondary.

In a true HUD the display is projected on a combining glass structure mounted on top of the main instrument panel as shown in Figure 16 (Navy 1982). The model simulation HUD implementation will project the HUD image on the simulator dome directly in front of the pilot. The size of the image is calibrated to be approximately the same as that which would be viewed through the combining glass.

Figure 2 was presented in the introduction as the basic flight HUD. This is the portion of the HUD which processes flight information and allows the pilot to fly effectively without looking down at the instruments in the cockpit. The three modules which are responsible for this process are:

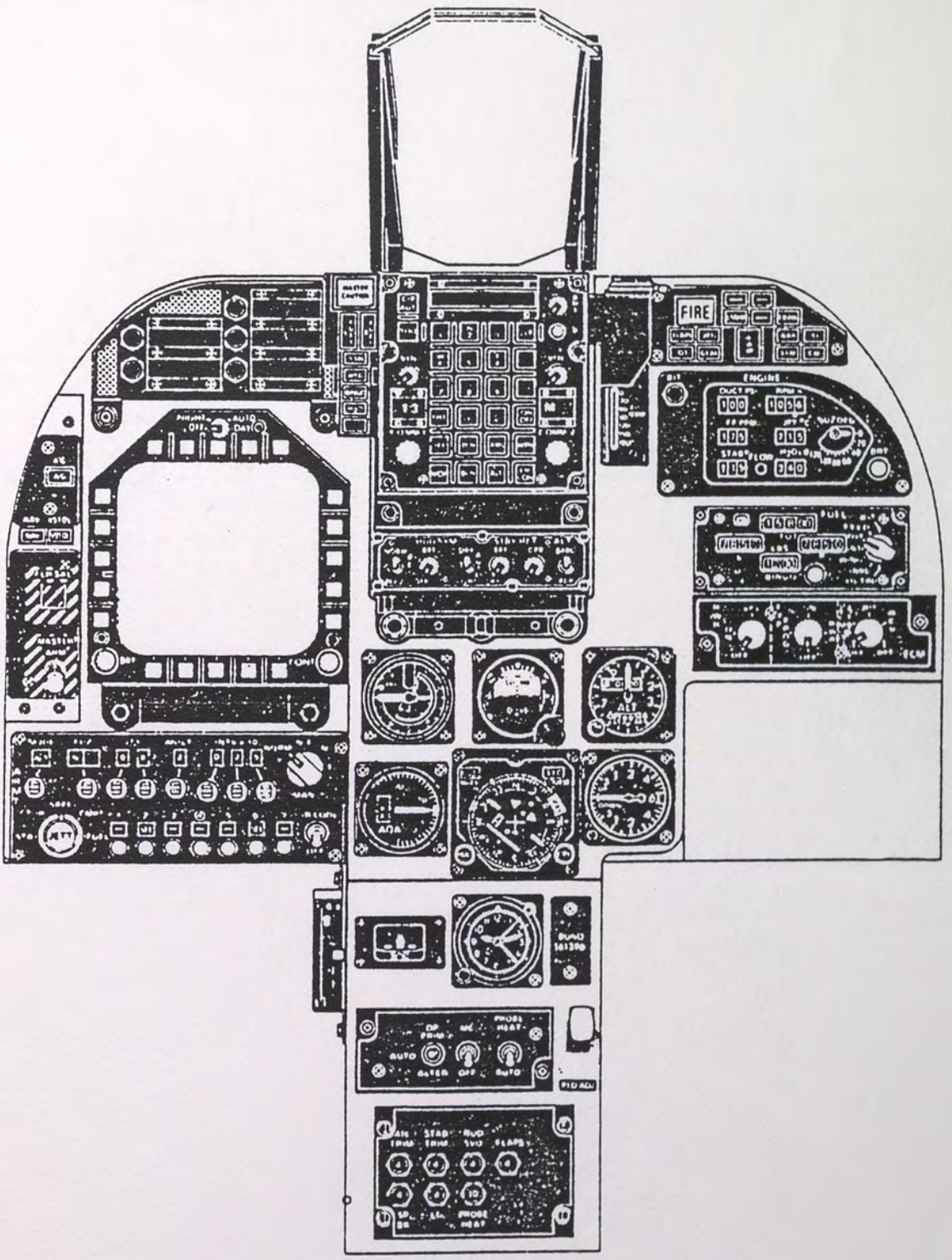


FIGURE 16: Basic Flight HUD

1. Flight data information
2. Roll/Pitch Ladder indicator
3. Flight Path Marker or Velocity Vector model

The combination of these processes are the basis for the flight HUD, from this many different variation can be developed, depending on the environment or purpose of the aircraft (McDonnell Douglas 198Ø). Each of these functions will be discussed separately in terms of definition and implementation factors pertaining to the HUD system at VTRS.

2.1 FLIGHT CONTROL INFORMATION

The format for flight control data is the alpha numeric representation of the navigation and steering control information.

HEADING: The Heading indicator is located in the top center box of the HUD display. This is displayed in degrees with zero degrees being due north. The indicator is represented in a clockwise fashion, for example 9Ø degrees is east and 275 degrees is west.

AIRSPEED: The Airspeed indicator is the box at the left and below the heading indicator. This is represented as an integer number in nautical knots.

ALTITUDE: The altitude indicator is represented in the box located at the right and below the heading on the HUD display. The alphanumeric representation in the box is the radar altitude which indicates the value derived from the aircrafts radar system. A backup system can be implemented which would automatically display the barometric altitude in the event that the radar altitude should fail. This would be distinguished from the radar altitude by placing a B on the right-hand side of the box, causing the number to flash on and off.

AIRCRAFT G: The G is located in the bottom left-hand corner of the Head-Up Display. This is the normal acceleration of the aircraft and is displayed in feet per second. The G is represented as a real number with one significant digit.

ANGLE OF ATTACK: The angle of attack is represented on the HUD as an alpha symbol, and is located just below the aircraft G. This is displayed in degrees as the True Angle of Attack. The alpha shown gives the angle at which the aircraft is in relation to the target.

MACH NUMBER: The large M displayed just beneath the Angle of Attack symbol represents the Mach number. This is displayed as a two significant digit real number and calculates the fractional airspeed in speed of sound units.

RATE OF CLIMB/DESCENT: The FPM indicator located on the bottom right hand side of the Head-Up Display, gives the rate of climb or descent in Feet Per Minute. The negative sign indicates the aircraft is descending.

2.2 VELOCITY VECTOR

The Velocity Vector is represented as the small stick figure aircraft, usually located around the center of the HUD. This gives the pilot the actual flight path of the aircraft. The Velocity Vector is placed on the HUD at a point which describes the path which the aircraft is taking. On the HUD the Velocity Vector is caged to the vertical center line, this allows reference to the Roll/Pitch Ladder and steering information to be used with the Velocity Vector. The equation used by the HUD simulator for calculation of the Velocity Vector is shown below.

VV=P*alpha

Where P is the actual Pitch angle of the aircraft and alpha represents the Angle of Attack.

The Velocity Vector is used by the Roll/Pitch Ladder as the center point for the Roll angle. This gives the true roll point, but makes it more difficult to determine the actual pitch angle of the aircraft.

2.3 ROLL/PITCH LADDER

Artifical Horizon or Attitude Indicator is the conventional term for the Roll/Pitch Ladder. The Roll/Pitch Ladder or Flight Path/Pitch Ladder is used by the pilot to determine aircraft position with respect to the Flight Path Marker or Velocity Vector. This is determined with a display of flight path pitch lines representing five degree increments. These lines are each numbered at their respective angle and have a range of plus or minus 90 degrees. Positive pitch lines appear above the horizon, and are represented by solid lines. The negative pitch lines are those below the horizon line and are signified by dashed lines. The Roll/Pitch Ladder software is designed for normal operation between -90 and +90 degree pitch values. When a pitch angle is detected which exceeds these limits the angle is treated as the inverted or 180 degree roll

angle and the difference between the actual pitch angle from plus or minus 90 degrees is subtracted from the 90 degree pitch line. This works in the same manner as the glide slope indicator instrument functions in the aircraft cockpit. Thus, for example, an actual pitch of 180 degrees and roll of 0 degrees would appear on the Roll/Pitch Ladder as flying at 0 degree pitch upside down or with 180 degrees roll. The outer segments of the pitch lines adjacent to the character representation of the angle designate the direction toward the zero pitch line or the horizon line. This serves as an aid in determining flight path angle when dynamic elements are rapidly changing.

The Roll/Pitch Ladder database was constructed using a database development program written in FORTRAN specifically for this application. The program is entitled MAKDAT and was designed to allow maximum flexibility in Ladder in database construction. This allows the Roll/Pitch Ladder database to be modified easily and allows size calibrations to be accomplished quickly. Listings of the database modelling program (MAKDAT), and the actual data base (LADDER.DAT), are not presented for purposes of conserving document length. Listings and user instructions are available through the author for those interested.

The size of the Roll/Pitch Ladder as a whole can be adjusted by the user, as an option in the Host Control Program (HCP). This is a scaling factor which is multiplied

into each point representing the Roll/Pitch Ladder database. This feature allows for interactive changes in the size of the entire database. If only a part of the Roll/Pitch Ladder needs adjustment, such as the character size being enlarged, the database modelling program would have to be used.

The Roll of the aircraft is displayed as the rotational portion of the Roll/Pitch Ladder, and rotates about the Velocity Vector.

CHAPTER III

MICROCODE DEVELOPMENT

The Ikonas Graphics Processor's modular architecture is designed to allow multiple processing units to share a common bus. Therefore the individual modules within the Ikonas unit, use microcode instructions to process information. There are several different configurations which could be applied to the HUD system, because of the nature of the Ikonas system. Control of the Ikonas Bus can come from either the Host Computer or from within the Ikonas Processing Unit. Microcode instructions are executed independently from each respective Ikonas module, and only needs to communicate with others when requesting the use of the Ikonas Bus. Thus a program in the host computer can be inputting data and updating, while the Ikonas Graphics processor is displaying the previous information. Using this method the processing time would be the time it takes to transfer the necessary information from the PDP-11/34 Host Computer to the Ikonas working memory, plus the Ikonas image processing and display output time. With this

configuration the update rate of the HUD system was accomplished at 3Ø Hz.

There are two areas within the Ikonas Graphics Processor in which microcode programs for this application are stored. One is in the BMP32 Ikonas processor microcode memory and the other is the MA1Ø24 Matrix Multiplier microcode memory. From these memories each of the respective microcode sequencers can fetch microcode intructions quickly and effectively (Ikonas 198Ø).

The BMP32 processor is used as the main module and controls access to the bus. A memory map of the BMP32 microcode memory is shown in Figure 17. At location address Ø in microcode memory is HMCP.MOB. This is the HUD Master Control Program (Appendix A). All of the other microprograms are controlled from this sequence of microcode. Location 1ØØ is the start of the line display routine called BRESL1.MOB. This program plots lines from designated origination and termination points. The microcode located at address 5ØØ is HWIND.MOB. This creates a window in the Roll/Pitch Ladder database to be extracted, rotated by the roll angle, and displayed to the HUD projection system. The CGMS.MOB microcode starts at location 6ØØ and allows up to eight characters to be generated as a string, by the Character Generator.

The MA1Ø24 has one microcode program located at address Ø in the matrix multiplier microcode memory. This program,

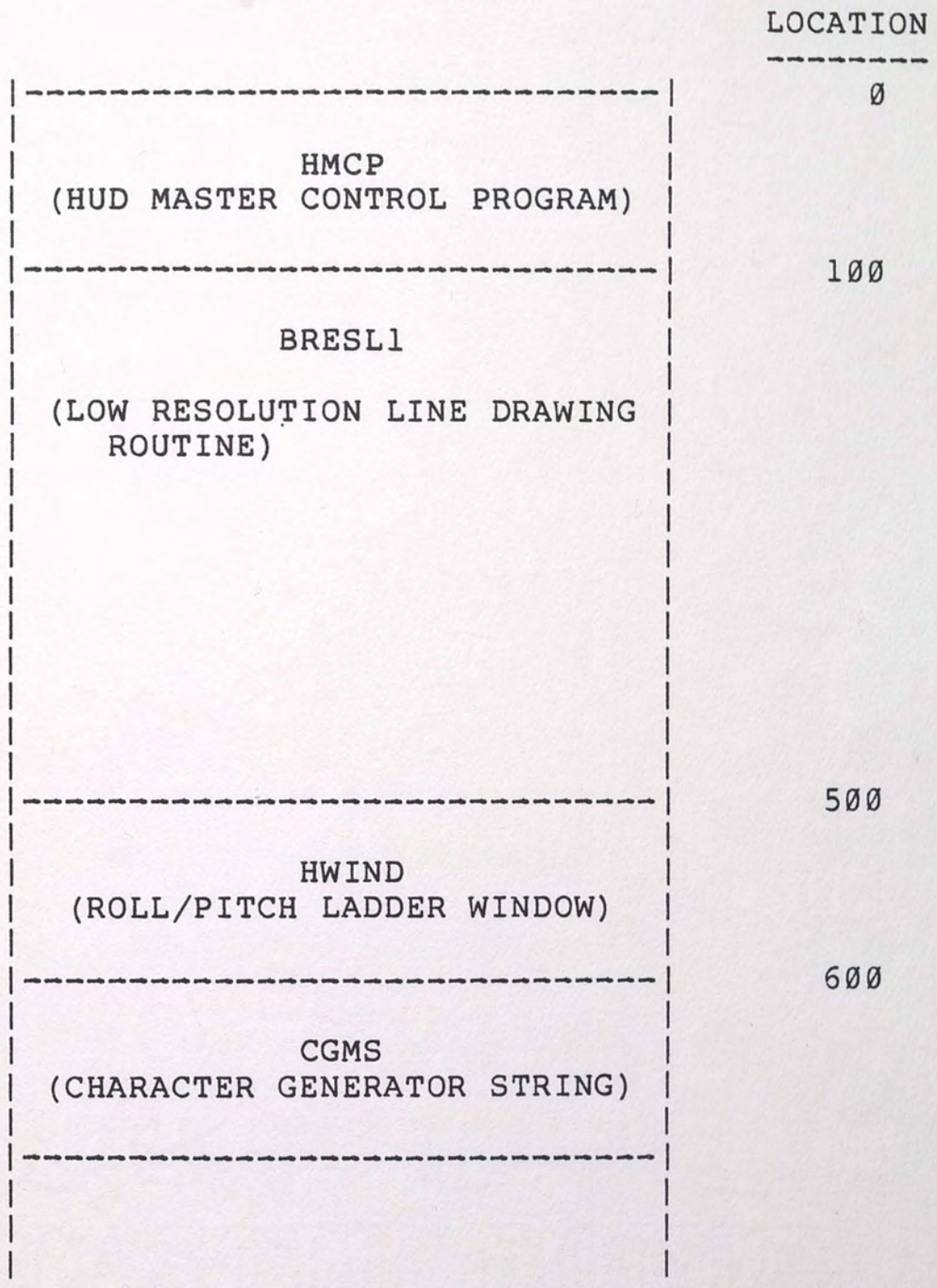


FIGURE 17: BMP32 Microcode Memory Map

given a coefficient matrix and data address, will rotate a given three-dimensional object.

These microprograms as well as control and status information from the Host Control Program (HCP), display the flight information transferred to it in the form of the Head-Up Display.

3.1 MASTER CONTROL PROGRAM (HMCP)

The HMCP microcode was developed as the main module in the Ikonas Processor, which calls the other microcode modules and controls status and bus lines. Figure 18 is a flow diagram of the HMCP program. The processor is started by the Host Control Program when the processor is syncrhonized with the display field (Chapter IV). A listing of the microcode intructions for this program is presented in Appendix A-1. The flow diagram shows each subprogram as they are initiated starting with the Roll/Pitch Ladder window program HWIND. After HWIND has returned completed, the window which was just extracted must have the terminator bits placed in the last word of the buffer. The termination byte of "177777 is written in the last location. This is used by MMEOL to determine where the end of the data is detected. Next the control block address for the MA1Ø24 is loaded and sent. This starts the MA1Ø24 processor by loading the first microcode instruction located at address Ø

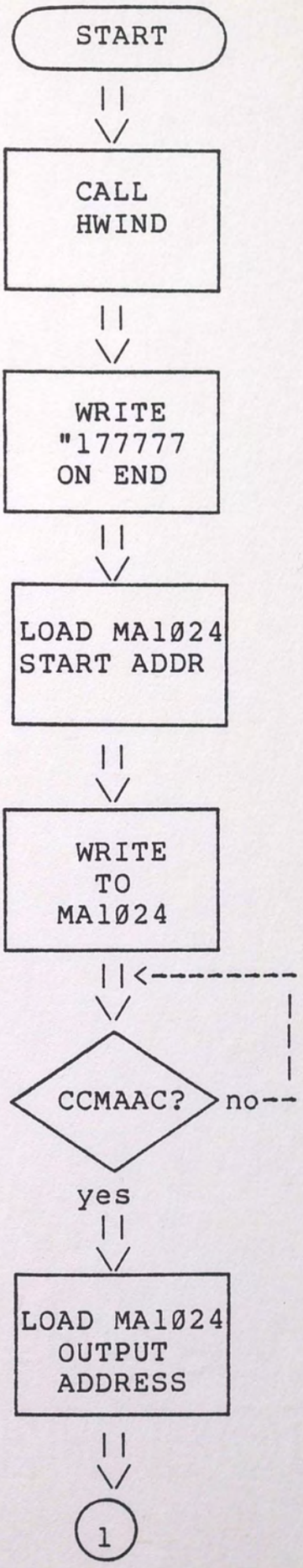
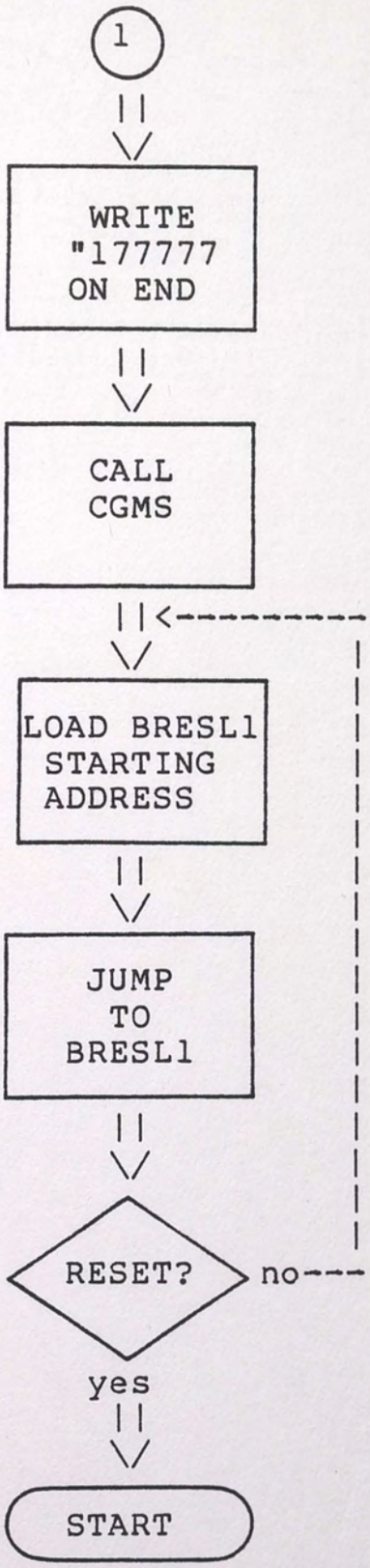


FIGURE 18: HMCP Flow Diagram


1
WRITE
"177777
ON END
CALL
CGMS
LOAD BRESL1
STARTING
ADDRESS
JUMP
TO
BRESL1
RESET?
no
yes
START

in MA1Ø24 microcode memory (MMEOL). HMCP then waits for the condition code CCMAAC to go high which indicates the rotation is completed. Once the rotation has finished and the new points are placed in the Output Buffer, the terminating word must again be placed at the end of the buffer. BRESL1 also detects the end of data by the same terminator bits. The next step is to output the character strings required for the HUD flight information displays. CGMS is the microcode which sets up and executes the Character Generator Module and displays character strings of up to eight characters in length. When this is completed HMCP calls BRESL1, which takes the points from the Ikonas Output Buffer and draws the appropriate lines to the display. This process is repeated until the Ikonas processor is reset to zero and the microcode begins again on the next field.

3.2 ROLL/PITCH LADDER WINDOW (HWIND)

The HWIND microcode routine is used to extract points from the Roll/Pitch Ladder database, which are to be displayed on the HUD. Figure 2Ø shows a flow diagram of HWIND and the microcode is presented as Appendix A-2. This routine starts by inputting the window control block and loading it into the BPS32 processor's 32 bit working registers. Starting with RØ the following information is

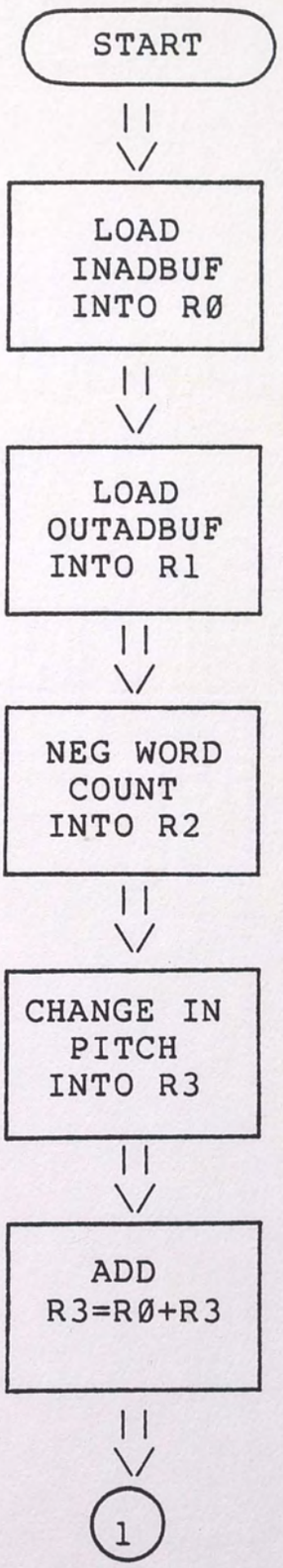
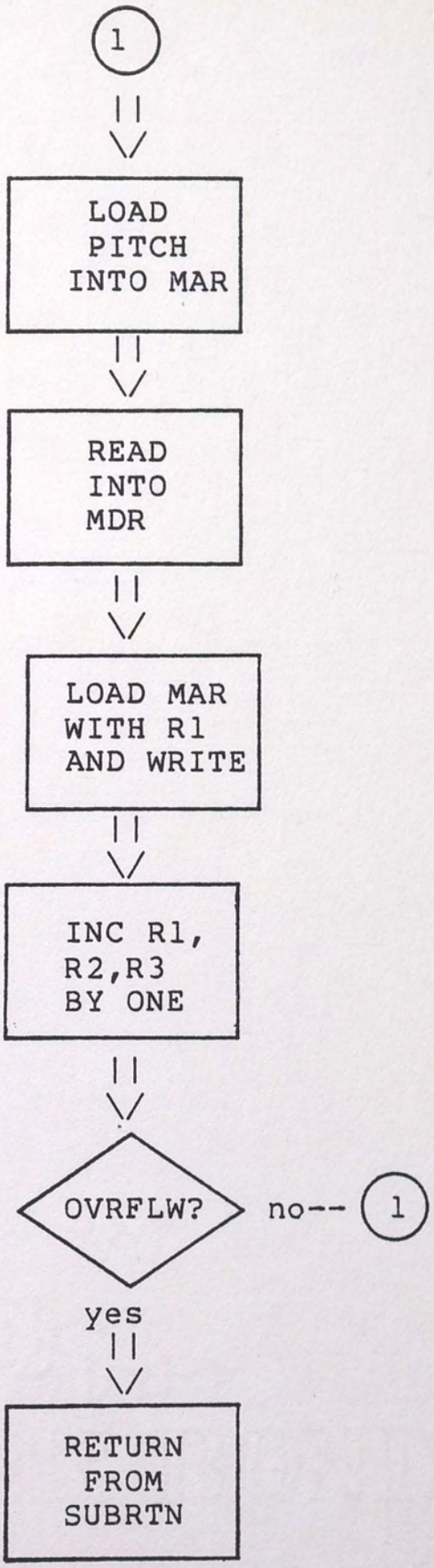


FIGURE 19: HWIND Flow Diagram


1
LOAD PITCH INTO MAR
READ INTO MDR
LOAD MAR WITH R1 AND WRITE
INC R1, R2,R3 BY ONE
OVRFLW?
no--
1
yes
RETURN FROM SUBRTN

read from the HWIND control block located in SR8 scratch memory.

1. RØ - Address of input buffer
2. Rl - Address of output buffer
3. R2 - Negative word count of window points
4. R3 - Pitch offset within database

From this control data, which is determined and sent by the Host Control Program, the correct pitch attitude is extracted from the remainder of the database. This smaller subset is now ready to receive the roll rotation, before being output. The beginning of the window is determined by adding the input buffer address (RØ), with the calculated pitch attitude (R3). This gives the address of the first point to be extracted and placed in the Memory Address Register (MAR) and the data at that address to be written to Register 1, and the contents of MDR is output to that location. Each of the input and ouput addresses are then incremented to prepare for the transfer of another data point. The negative word count (R2), gives the size which the window is to be. This register is incremented after each cycle until it overflows. This indicates that the entire window has been transferred and contol returns to HMCP.

3.3 CHARACTER GENERATOR MODULE (CGMS)

The Character Generator can be intialized and controlled in several manners. Control from the Ikonas

processor was determined to be the most straightforward and economical method. The standard on board font table is used by the character generator to produce characters which are displayed with 5X7 pixel dimensions. CGMS is a microprogram loaded in the BMP32 processor, which starts and controls the characters being produced. A listing of this microprogram is given in Appendix A-3. This microcode is a modified version of the original character generator control program CGMMCR.MOB (Sampayo 1982). Figure 20 is a flow diagram of the CGMS microcode routine. The program starts by loading the CGMS Control Block from the SR8 working memory, into the BPS32 processor's 32 bit registers. The descriptions of each control word and their locations in memory are given in the flow diagram. A working buffer in SR8 memory of three Ikonas 32 bit words is set aside and cleared for use by the character generator. The program then fetches the first pixel address in which the character string is to be displayed. Then the appropriate character string is placed into the working buffer.

The Character Generator is started by first sending the address of the Character Generator control block to location 20600$1, and then sending the correct startup and control bits to location 20600$0. While the character string is being output, the processor increments the pixel address and character string pointers and checks for an ALU overflow, which indicates the last string is presently being output.

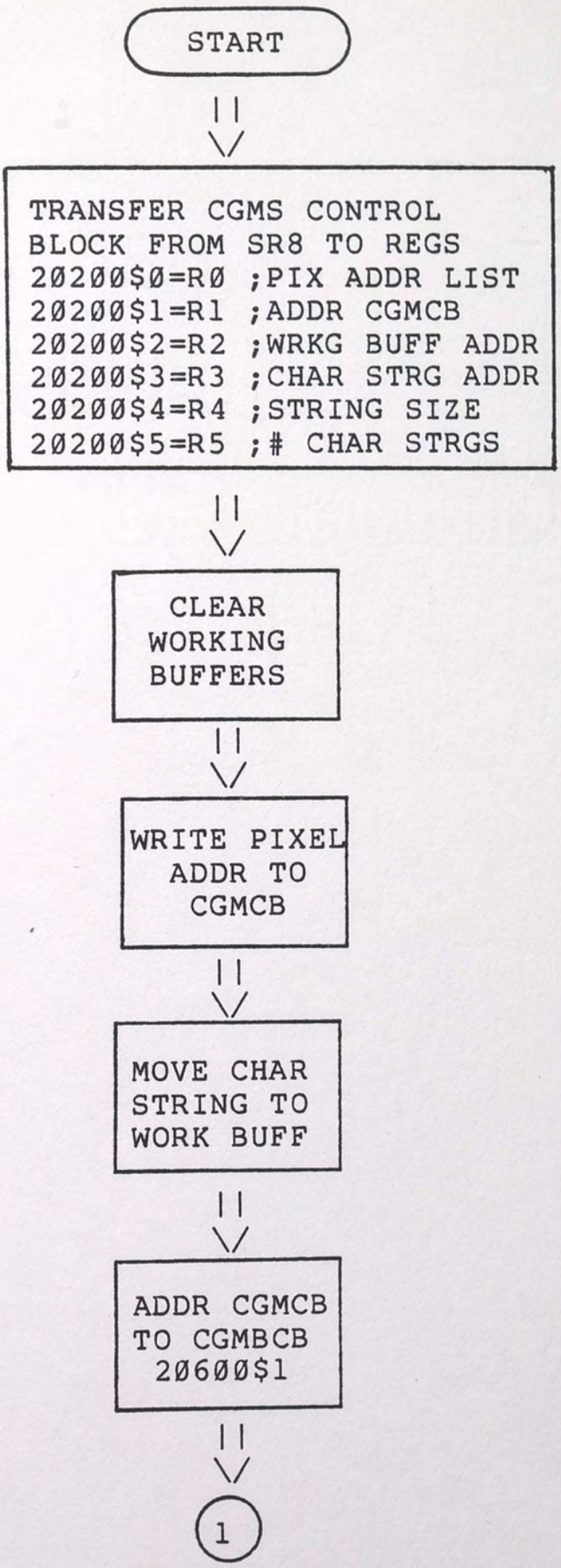
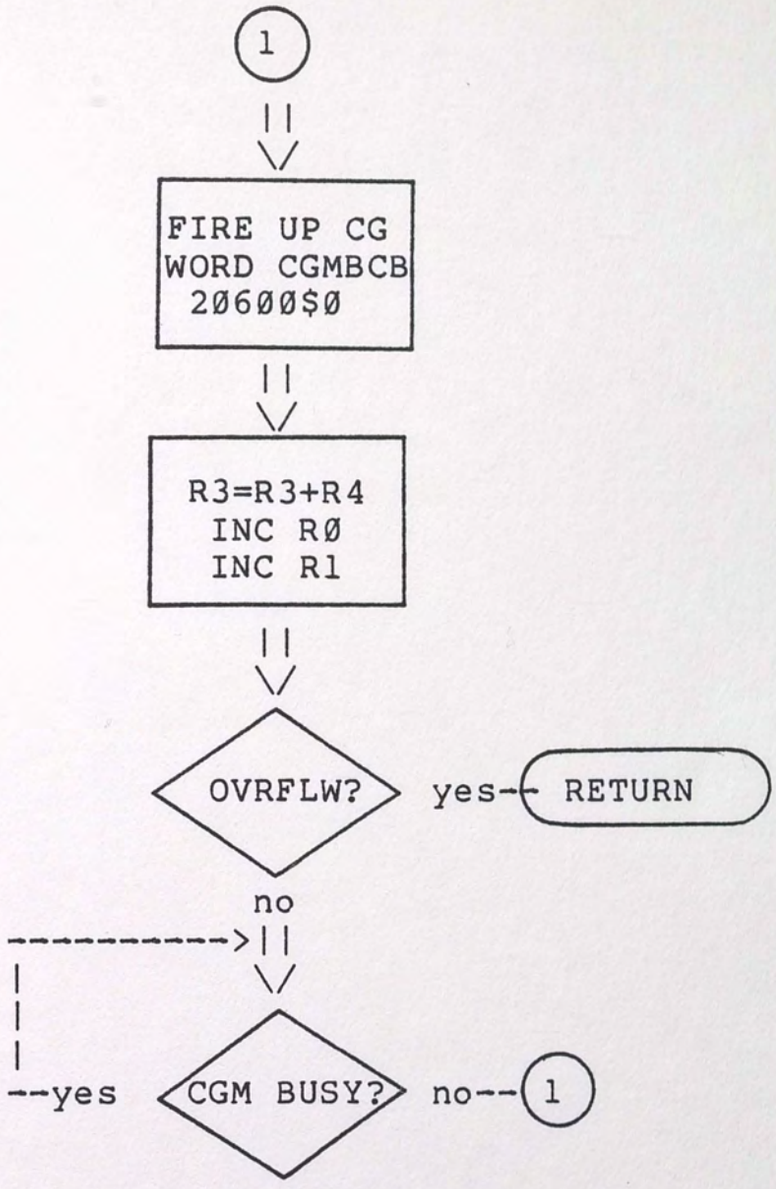


FIGURE 20: CGMS Flow Diagram


1
FIRE UP CG
WORD CGMBCB
20600$0
R3=R3+R4
INC R0
INC R1
OVRFLW?
yes
RETURN
no
CGM BUSY?
yes
no
1

If an overflow does not occur the program begins processing the next character string. Once complete, control returns to the Hud Master Control Program (HMCP).

3.4 MATRIX MULTIPLY (MMEOL)

The Matrix Multiply routine is a system subroutine which is supplied by Ikonas Graphics Inc (1982). MMEOL is stored in the MA1Ø24 microcode memory and starts up when the MA1Ø24 processor is activated. This microcode routine will utilize MA1Ø24 coefficient memory for multiplication of the data points with the coefficient matrix. The coefficient matrix is calculated and sent to the MA1Ø24 by the Host Computer after each frame occurs. The Roll/Pitch Ladder points database is located in SR8 memory. After the HWIND microcode program extracts the active database segment, MMEOL will input the first point to be rotated. Once the matrix multiplication is complete, MMEOL will output the transformed point to an output buffer in SR8 memory, and increments the address pointer for the next point to be rotated. The end of the data base segment is detected by MMEOL using flag bits. The last word of the each point indicates the type of action to be done. In the case of MMEOL, if bits 3Ø and 31 are high, it indicates the end of points to be rotated.

3.5 LINE DRAWING (BRESL1)

The Line Drawing Routine is also a microcode subroutine supplied by Ikonas, and it resides in the BPS32 microcode memory. This microcode is called when it is required to draw lines between points on the diplay. BRESL1 inputs points from SR8 memory, calculates the slope between two points, and generates a line segment on the display. This program also uses flag bits to determine the type of point which is input. Bits 30 and 31 of the last word representing a point, are used as indicators. Figure 21 shows a truth table of the different flag bit settings.

BITS 31	BITS 30	ACTION
0	0	MOVE TO POINT
1	0	DRAW LINE TO POINT
1	1	LAST POINT, DRAW TO
0	1	XXXXXXXXXXXXXXXXXX

FIGURE 21: BRESL1 Flag Bits

If a 00 is sent, this represents a starting point for BRESL1 to move to before drawing a line. A 10 indicates a line is to be drawn to that point from the previous point. And a 11 represents the last point to be drawn to and the end of the input buffer.

CHAPTER IV

HOST PROGRAMMING

The Host Control Program is written using PDP-11 Fortran IV-plus and executes the initialization, startup and update in a realtime environment. Figure 22 is a flow diagram of the Host Control Program (HCP.FTN), and a listing of this program appears in Appendix B. The first step in HCP is to obtain the physical addresses of the input and output data arrays. A detailed description of this calculation is given in the documentation of the HCP program listings. This procedure is the step which allows the host programming to be accomplished using Fortran. By obtaining the virtual address of an array, it can be accessed and transferred by the DR11B interface, using Fortran statements. There are two DR11B interfaces used in the HUD/Ikonas applications. The control variables are linked to the Host Control Program as common memory locations. The Option function is used during task building, to allow common virtual memory locations, between the program and the DR11B interfaces. The command lines are used to initialize

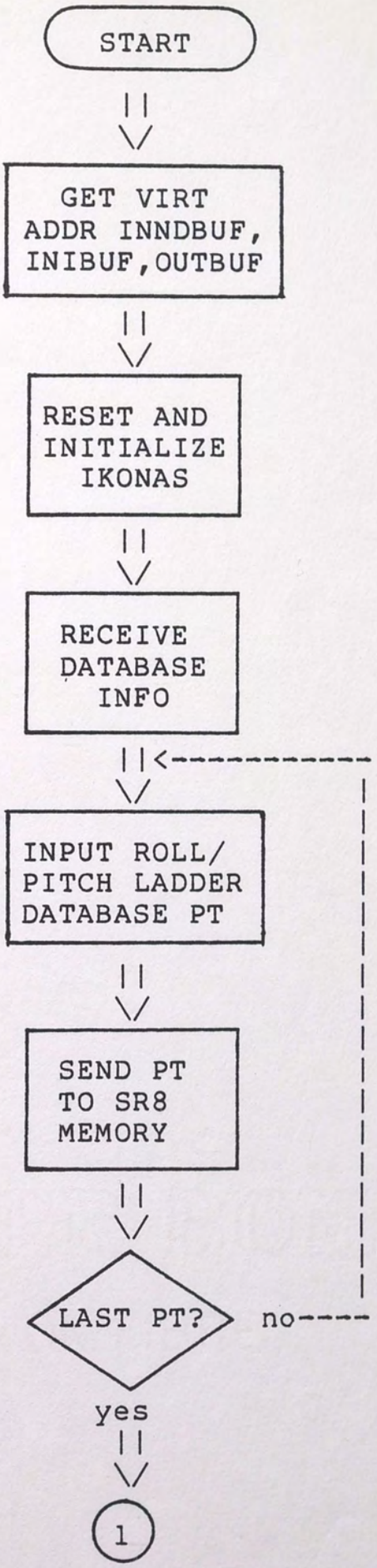
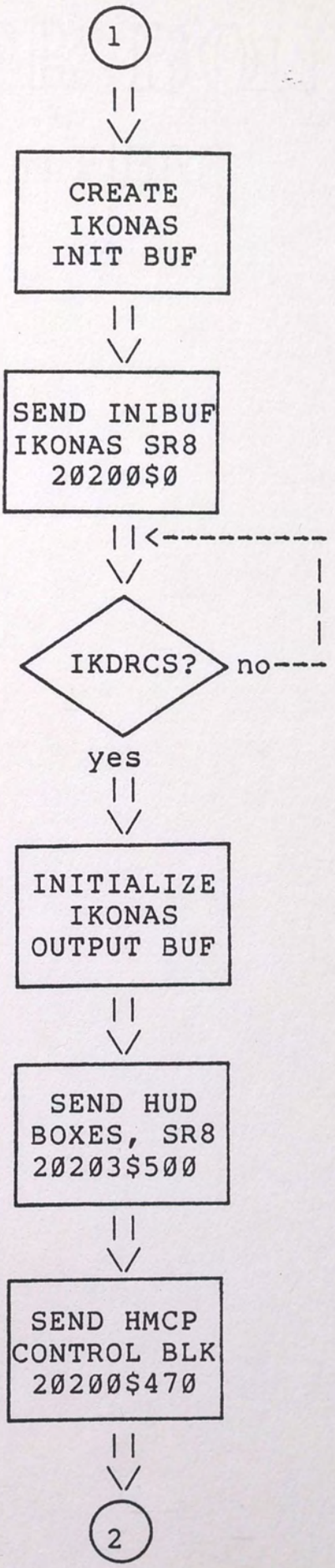
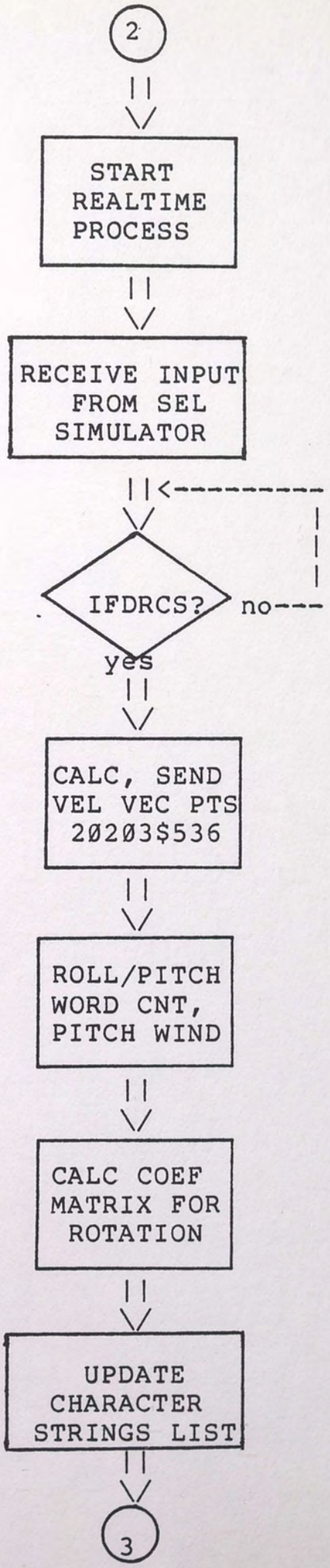
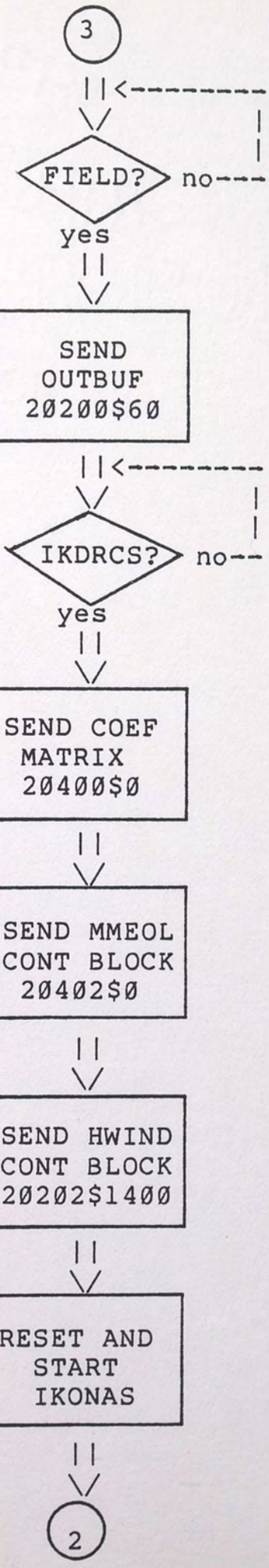


FIGURE 22: Host Control Program (HCP) Flow Diagram


1
CREATE IKONAS INIT BUF
SEND INIBUF IKONAS SR8 20200$0
IKDRCS?
no
yes
INITIALIZE IKONAS OUTPUT BUF
SEND HUD BOXES, SR8 20203$500
SEND HMCP CONTROL BLK 20200$470
2


2
START REALTIME PROCESS
RECEIVE INPUT FROM SEL SIMULATOR
IFDRCS?
no
yes
CALC, SEND VEL VEC PTS 2Ø2Ø3$536
ROLL/PITCH WORD CNT, PITCH WIND
CALC COEF MATRIX FOR ROTATION
UPDATE CHARACTER STRINGS LIST
3


3
FIELD?
no
yes
SEND
OUTBUF
20200$60
IKDRCS?
no
yes
SEND COEF
MATRIX
20400$0
SEND MMEOL
CONT BLOCK
20402$0
SEND HWIND
CONT BLOCK
20202$1400
RESET AND
START
IKONAS
2

the virtual memory address which are used in the Host Control Program.

```
>LINK/CODE:FPP/PRIVILEGED:4/OPTIONS HCP

 OPTION?  VSECT=PAGADF:172350:2:1
 OPTION?  VSECT=IFDR11:172450:10:1
 OPTION?  VSECT=IKDR11:172460:16:1
```

The COMMON statement is used in the program to identify the interface control locations with program variable names.

```
COMMON/IFDR11/IFDRWC,IFDRBA,IFDRDR

COMMON/IKDR11/IKDRWC,IKDRBA,IKDRCS,IKDRDR,
*IKSLAD,IKSHAD,IKSCSR
```

The IFDR11 us used as a label for the SEL-PDP-11 DR11B interface, and the IKDR11 represents the PDP-11-IKONAS DR11B interface.

To determine the actual physical address of the input and output buffers, a link between the program and the Page Address Field (PAF) in the PAR register, must be made.

```
COMMON/PAGADF/PAF
```

A system subroutine called GETADR is used to obtain the virtual address of each buffer and place them in an array called VIRTAD.

```
CALL  GETADR(VIRTAD,INIBUF(1),INNBUF(1),OUTBUF(1))
```

Once this is accomplished the physical address can be found using VIRTAD and the PAF.

The Ikonas reset and initialization steps utilize the buffers set up for the DR11B interface. The following is a list of the Ikonas DR11B interface variables and their function. Using these variables, information can be transferred to the Ikonas system through the DR11B interface.

1. IKSLAD - LOW 16 BITS OF IKONAS MEMORY ADDRESS
2. IKSHAD - HIGH 16 BITS OF IKONAS MEMORY ADDRESS
3. IKSCSR - IKONAS FUNCTION CODE REGISTER
4. IKDRDR - SINGLE TRANSFER REGISTER
5. IKDRWC - IKONAS BLOCK TRANSFER NEG WORD COUNT
6. IKDRBA - BLOCK TRANSFER INPUT ADDRESS
7. IKDRCS - DR11B STATUS CONTROL REGISTER

The Ikonas interface is reset by first setting the Ikonas function code line and then clearing it. The processor is set up to allow automatic erase, by sending the proper status bits to Ikonas location 30000$5. With this capability, the screen will automatically erase at each field to prepare to the drawing of the next field. This requires that every object on the display be generated at each 30Hz field interval.

The Host Control Program allows for user input from the system terminal, to determine the position of display features, select input database, and choose a scaling factor to increase or decrease the size of the display. Detailed descriptions of these options and their uses are given in the HUD Users Guide. The database is read into a scratch array and immediately transferred to Ikonas SR8 memory. A Memory Map of the Ikonas SR8 memory for the HUD application is shown in Figure 23. The database starts at Ikonas memory location 20200$500, and can consume up to "2700 Ikonas 32 bit words in SR8 memory. Each data point of the database is represented by 2 Ikonas words, therefore approximately 750 points can be stored. Appendix C gives detailed descriptions of data types and formats necessary.

Once the database is input, HCP begins the inialization phase of the program. The Inialization Buffer INIBUF is created, containing control block and address information. The first block of data is the CGMS control block, which is responsible for the addresses of all the other control blocks used by the Character Generator module to initialize and execute. The character string pixel address list is placed at the end of the intialization buffer to be used by the generator when outputting character string data. INIBUF is then sent to the Ikonas SR8 memory, using a block transfer, to location 20200$0. Setting up the character output buffer is done for error-debugging purposes. If an

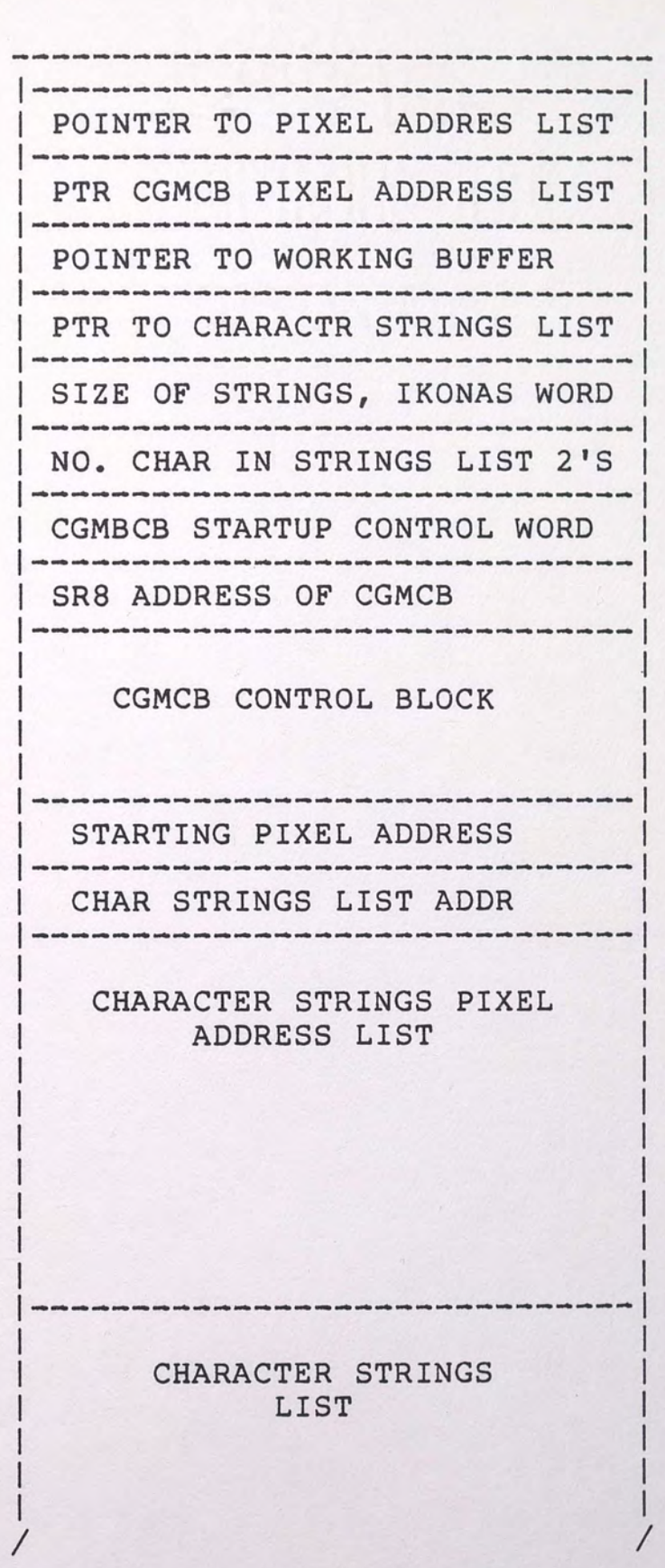
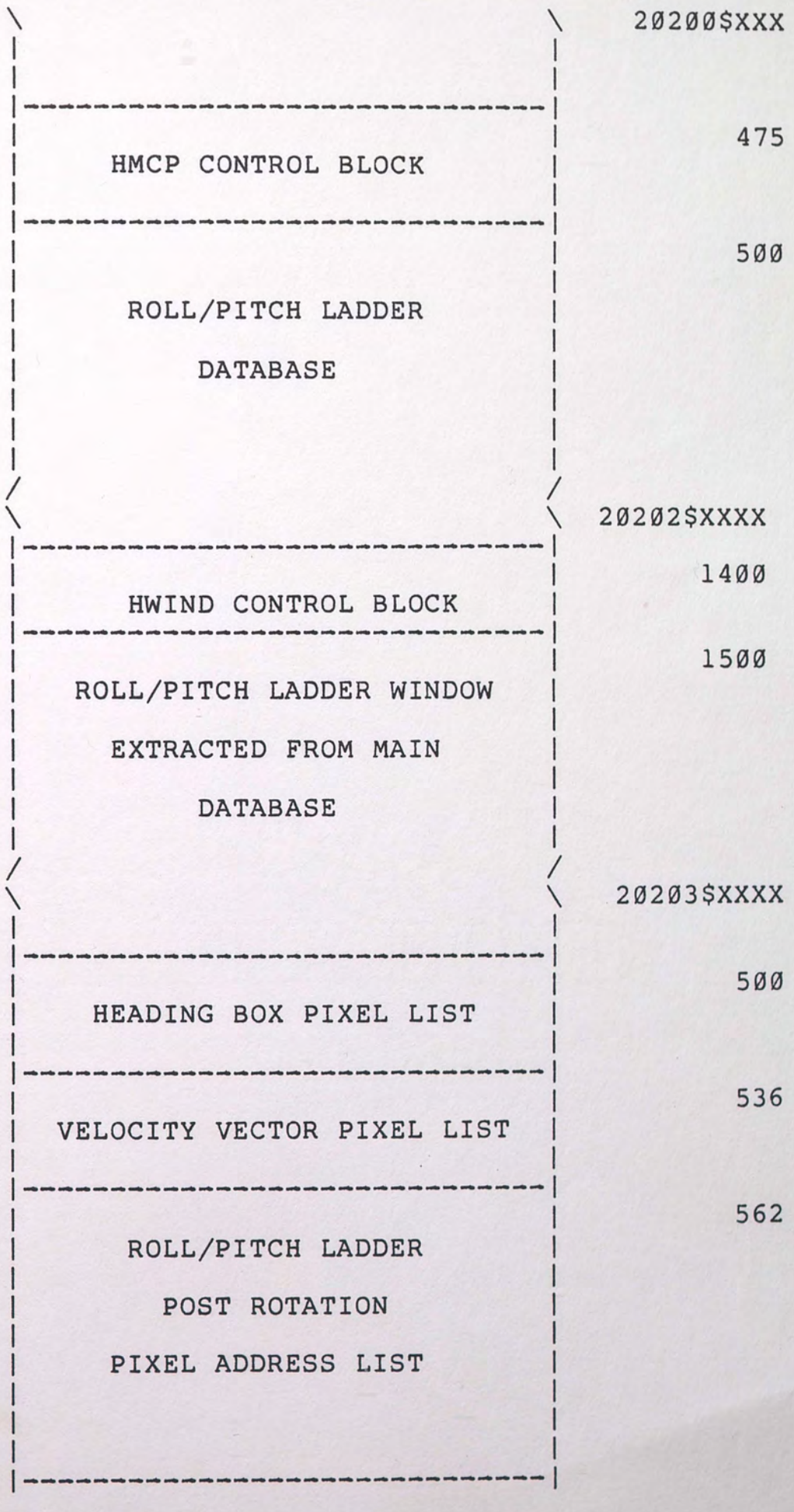


FIGURE 23: IKONAS SR8 MEMORY MAP


20200$XXX
475
HMCP CONTROL BLOCK
500
ROLL/PITCH LADDER
DATABASE
20202$XXXX
1400
HWIND CONTROL BLOCK
1500
ROLL/PITCH LADDER WINDOW
EXTRACTED FROM MAIN
DATABASE
20203$XXXX
500
HEADING BOX PIXEL LIST
536
VELOCITY VECTOR PIXEL LIST
562
ROLL/PITCH LADDER
POST ROTATION
PIXEL ADDRESS LIST

error occurs the character output can be traced, without requiring communication with the simulator computers. The SIZE buffer is used to set character string lengths for each output.

Pixel addresses for the three indicator boxes displayed on the HUD are generated and output to Ikonas SR8 memory, starting at location 20203$500. This is the beginning address of the Ikonas display data output buffer, which is used by the line drawing microcode routine BRESL1 and output to the HUD display.

The next step is preparing to enter the realtime module, by setting up control data and sending all information to the Ikonas which does not have to be updated every field. Here the control block for HMCP microcode is sent to SR8 memory starting at location 20200$470.

The first step in the realtime module is to receive the dynamic data input buffer from the Systems Engineering Laboratories simulator computers. This is accomplished in the same manner as the Ikonas Interface transfers, with the exception that only read attributes are required. No handshaking communications are necessary for a High Speed Data (HSD) transfer, thus only inputs are required. The IFDR11 is LINKED to the Host Program and the following control variables may be used for transfer of the 256 byte input buffer INNBUF from the SEL to the PDP-11 Host computer.

1. IFDRWC - SEL BLOCK TRANSFER NEG WORD COUNT
2. IFDRBA - SEL BLOCK TRANSFER BUFFER ADDRESS
3. IFDRCS - SEL/DR11B FUNCTION CODE REGISTER
4. IFDRDR - SINGLE WORD TRANSFER REGISTER

The input buffer is initialized as INNBUF and the format for this 256 byte array is shown in Figure 24. The Velocity Vector calculations are done by the simulator computer and the numerical coordinates are sent over in INNBUF, from these values the position of the Velocity Vector is updated. The pixel addresses are then sent to the Ikonas data output memory and placed after the indicator boxes pixel address list, location 2Ø2Ø3$536. The Roll/Pitch ladder update are done using viewpoint pitch and roll values sent by the SEL computers. These are used to determine the Ikonas address offset for extracting the Roll/Pitch Ladder active window display. Calculation of the Matrix Multiply Cofficient Matrix uses the roll value to determine the rotation of the Roll/Pitch Ladder window to be displayed. The coefficient matrix is a 4X4 matrix which multiplies each three dimensional point to obtain the new Roll position of the ladder. The Coefficient Matrix calculation is done prior to transfer to the Ikonas. The equations for three-dimensional rotation are shown below (Rogers 1976).

BYTE	FUNCTION
1	Display control: False=off; True=on
2	Detail control for Hud and Optical Sight:F=Ø;T=1
3	Master Switch state: False=off; True=on
4	Ordinate selected B state: False=off; True=on
5	" " G " " "
6	" " R " " "
7	Armament Safe state: False=unsafe: True=safe
8	Hud select: Ø=Hud; 1=Optical Sight
9	Mil selection (Ø to 255 Mils)
1Ø-16	not used
17-24	ASCII representation of the mil depression
25-32	ASCII representation of heading
33-4Ø	" " air speed
41-48	" " altitude
49-56	" " g's
57-64	" " angle of attack
65-72	" " mach
73-8Ø	" " vertical speed
81-88	" " aircraft pitch angle
89-128	not used

FIGURE 24: Computer Interface Buffer Definition

129-132 X position of View Point

133-136 Y " " "

137-140 Z " " "

141-144 YAW attitude of View Point

145-148 PITCH " " "

149-152 ROLL " " "

153-156

157-160

161-164

165-168

169-172

173-176

177-178 X position of Target

181-184 Y " "

185-188 Z " "

189-192 YAW attitude of Target

193-196 PITCH " "

197-200 ROLL " "

201-256 not used

```
ICOEF(1)=COS(Y)*COS(R)
ICOEF(2)=(SIN(P)*SIN(Y)*COS(R)+COS(P)*SIN(R))
ICOEF(3)=(SIN(P)*SIN(R)-COS(P)*SIN(Y)*COS(R))
ICOEF(4)=512+IDIFF1
ICOEF(5)=-COS(Y)*SIN(R)
ICOEF(6)=(COS(P)*COS(R)-SIN(P)*SIN(Y)*SIN(R))
ICOEF(7)=(COS(P)*SIN(Y)*SIN(R)+SIN(P)*COS(R))
ICOEF(8)=512+IDFR+IDIFF
ICOEF(9)=SIN(Y)
ICOEF(1Ø)=-SIN(P)*COS(Y)
ICOEF(11)=COS(P)*COS(Y)
ICOEF(12)=512
```

The character strings update is done by transferring the input character values to the output buffer, OUTBUF. The size buffer is used to determine how many characters are to be transferred for each string. A terminating byte of all zeros is placed at the end of each string and is used by the Character Generator, to find the end of the string.

At this point all updates and realtime calculations are complete and ready to be sent to the proper locations in Ikonas memory. First the Host Control Program will pause until the next Ikonas field takes place. At that time OUTBUF, the Coefficient Matrix, HWIND control block, and the Matrix Multiply control block for MMEOL, are sent to the

Ikonas. The output buffer is sent using a block transfer to Ikonas SR8 memory location 20200$60. The HWIND control block is five Ikonas words in length and starts at location 20202$1400. The Coefficient Matrix and MMEOL control block are sent to the MA1024 module at locations 20400$0 and 20402$0 respectively.

Once the update transfers are complete the Ikonas processor is reset and restarted, which begins the microcode processing. The microcode routine HMCP.MOB is addressed and the BMP32 processor pbtains control of the Ikonas Bus. The Host Control Program then returns back to the start of the realtime processing module and begins updating for the next field.

CHAPTER V

DISCUSSION OF WEAPONS DELIVERY SYSTEMS

There are several weapons systems which can be simulated with the Basic Flight HUD developed during this project. These systems are used by the pilot to determine accurate weapons release and target status information. In actual flight as in a simulation, the aircraft flight computers determine the position of the target, and depending on the type of weapon, calculate the estimated trajectory. This can be displayed to the HUD in the form of cross symbols, and aides the pilot in the release of a weapon. There are three types of weapons used in most air-to-ground or air-to-air applications. Those are rockets, guns and bombs (Navy 1982).

The Continuously Computed Impact Point (CCIP), is computed for all three types of weapons, in Air-to-Ground attack mode. The A/G mode also allows for AUTO release computations. An example of the symbology used for target designations is shown in Figure 25. There are three important elements used in A/G bombing operations. The CCIP

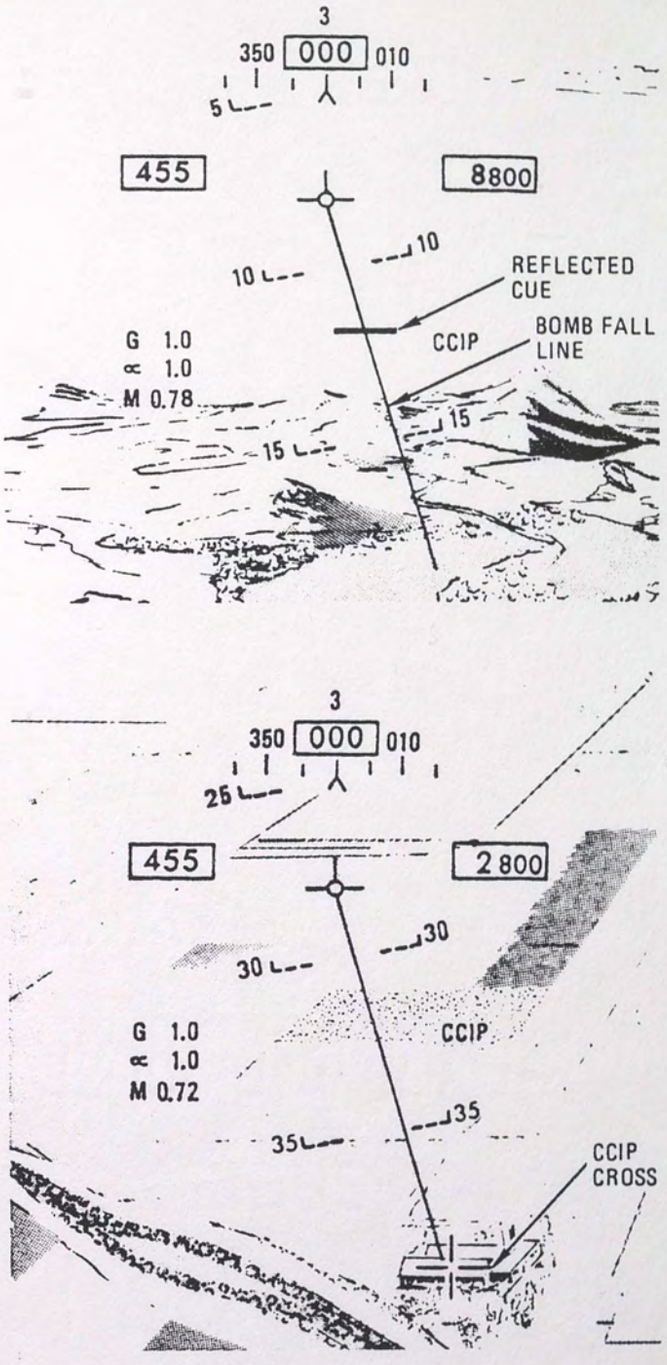


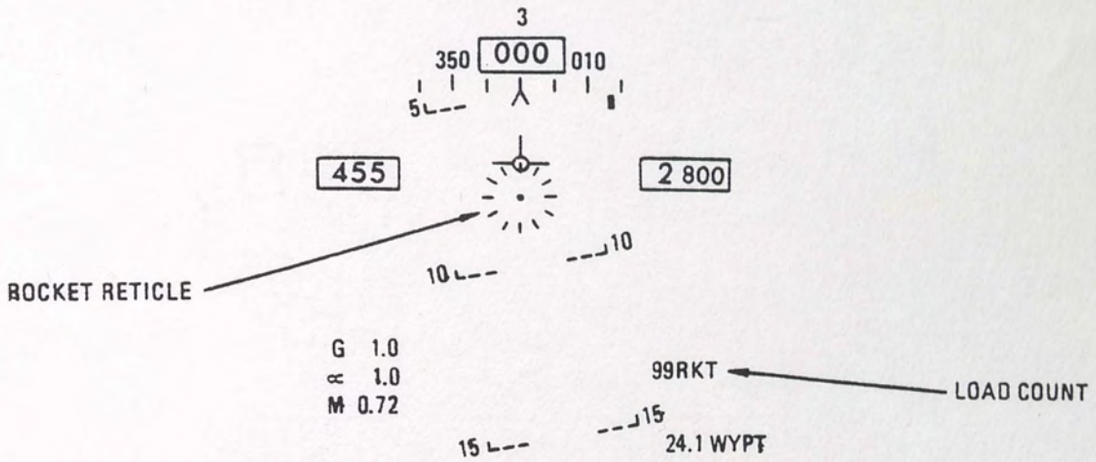
FIGURE 25: Target Designation Example

Cross gives visual reference to the target when within range. The Reflecting Cue gives an indication to the pilot of the optimal weapon release trajectory. And the Bomb Fall Line allows the pilot to maneuver over the designated target. Using these tools in conjunction with the Velocity Vector, the pilot can release a weapon when the Reflecting Cue, CCIP Cross, and Velocity Vector are aligned.

The rocket and gun reticle are usually circular sights, and indicate the release of the weapon when lined up with the Velocity Vector. This is illustrated in Figure 26. The rocket and gun modes, like the bombs, use the CCIP Cross for reference to the target. When the reticle, Velocity Vector and CCIP Cross are aligned the pilot can release the weapon. Each weapon has a calculated Milliradians of Depression, which is factored into the placement of the reticle. The Mil Depression is an amount of resistence on a weapon in flight, which causes it to change trajectory.

Air-to-Air mode settings only gun and rocket reticles, which are circular as shown in the A/G examples. The gun reticles usually have a layered pattern with more than one circle inside another. The A/A rocket reticle, on the other hand, consists of just one circle, which when at close range becomes dashed instead of solid.

There are many other applications of weapon status enhancement for Head-Up Displays. These are only a few of the more standard applications being used. The intent of



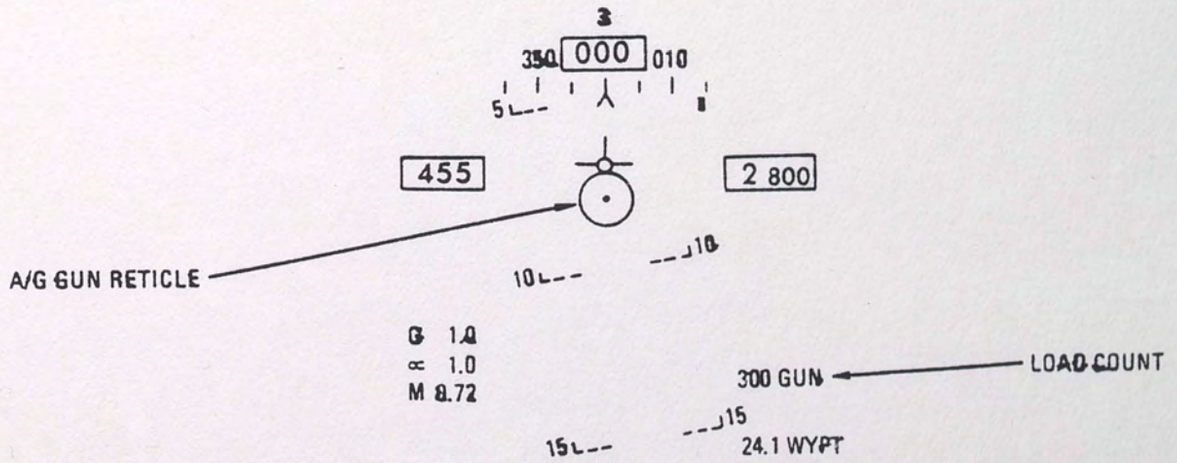


FIGURE 26: Rocket and Gun Reticle Examples

this chapter was to discuss possible features which could be simulated using the configured flight HUD developed. The HUD system was designed to be flexible so that several configurations could be implemented.

CHAPTER VI

CONCLUSIONS

The RDS 2000 Ikonas Graphics Processor is a useful tool for displaying limited sized graphics applications at a relatively fast update rate, approximately 400 vectors at a 30 Hz rate. The Ikonas BMP32 Microsequencer module has a high speed microcode fetch cycle, which allows for fast execution and transfer of instructions. The DR64 Image memory modules are a bit slice architecture, and provide a bit mapped display. This enables the Head-Up Display hardware to be simulated using Line Drawing, Matrix Multiply, and Character Generator modules supplied by Ikonas.

Microcode routines are used for Ikonas control, memory module manipulations, and display output. The configuration is such that more than one processor can be executing instructions simultaneously. These processor modules need only communicate when requesting the use of the Ikonas Bus. Therefore it was possible to overlay processing units such that the HUD simulation cycle is equivalent to the transfer

time of update buffers from the Host PDP-11/34 computers to the Ikonas Processor, plus the microcode execution and display time.

HUD cycle=Host Transfer + Ikonas Microcode/Display

All data transfers and update calculations on the Host computer are done while the Ikonas is processing and displaying the previous field.

The Host Control Program (HCP) was able to be developed in Fortran IV-Plus programming language. I/O buffers were linked to physical memory using virtual array addresses. This was accomplished by linking the program arrays to the Page Address Field (PAF), in PAR. Therefore the control program has direct access to hardware devices of the UNIBUS. The HCP is responsible for receiving the dynamic data from the SEL 32/75 simulator computer, updating pixel addresses, generating microcode control words, and sending data buffers to the Ikonas SR8 fast RAM memory module. The Host Control Program also resets and starts the Ikonas Processor at each 30 Hz field time. This allows synchronization between the Host and Ikonas system. Then as the Ikonas microcode is executed the HCP returns to update the next set of dynamic data input from the SEL and prepares it for the next field time. Using this overlapping technique, processing volume was increased by 50 percent.

The simulated HUD system was observed to have some problems with resolution at high activity periods. When the aircraft maneuvers quickly, either in a tight roll or a rapid change in pitch, the Roll/Pich Ladder becomes blurred and difficult to read. This was noted as a problem in actual HUD's as well. There several enhancements which are added in the actual HUD to help this problem, which were not investigated in this project (Navy 1982).

APPENDIX A

IKONAS MICROCODE LISTINGS

APPENDIX A-1

```
;
;     HMCP.ASM
;
;     AUTHOR: DARYL R. MAIR
;
;     THIS MICROPROGRAM IS THE MASTER CONTROL PROGRAM
;     FOR HUD PROCESSING MICROCODE.  IT CALLS ALL
;     OTHER MICROCODE PROGRAMS AND SYNCHRONIZES WITH
;     THE HOST CONTROL PROGRAM.
;
     ORG Ø
;
;
DEFAULT  NANOP CCNOP LDNOP SB ALUZ YD CARØ SSØ ALUBR MDRIKD
     MARIKA
;
;
START: JSBDF    5ØØ              ;JUMP TO SUBROUTINE HWIND
;
     NANOP
;
     LDUDR     177777                    ;LOAD INTO R OPERAND
     RIMM      ALUMDR    PR     Ø        ;LAST WORD 177777
;
     RA1        ALUMAR    PR     CAR1     ;BACK TO ADDRESS
     IKWR                                ;DATA OUT
;
     NANOP
;
     LDUDR      2Ø4
     RIMM      ALUMAR    PR    4ØØ3     ;LOAD ADDRS FOR MA1Ø24
;                                        ;START
     LDUDR     Ø
     RIMM      ALUMDR    PR     Ø        ;DATA EQUAL ZERO
;
     CCMAAC    WRSTRTMA                  ;WRITE TO IKONAS BUS
;
BUSY: CCMAAC     JMPDF     BUSY          ;WAIT FOR ROTATION
;                                        ;COMPLETE
     RLIMM     ALUMAR    RPS    B1     1Ø62    ;LOAD OUT ADDRS -1
     IKRD      IKBR      B4     BD             ;PLACE IN REG 4
;
     LDUDR     14ØØØØ
     RIMM      ALUMDR    RORS   B4     Ø    ;PLACE 177777 AT END
     IKWR                                    ; WRITE TO OUTPUT ADDR
;
     NANOP
;
     JSBDF     6ØØ              ;JUMP TO CGMMCR MICROPROGRAM
;                               ;TO DISPLAY CHARACTER INFO
```

```
    ;
    ;
    NANOP
    ;
LINE:  LDUDR    2Ø2                ;JUMP SUBROUTINE BRESL
    ;
    RIMM     PR      BD     BØ    475
    ;
NULPROC:
    ;
NXTCOM:  BØ    PS   ALUMAR        ;READ WHATS ADDR RØ
    ;
GO:      IKRD
    ;
    JMPIB                        ;JUMP TO SUBR AT ADDR RØ
    ;
    END
```

APPENDIX A-2

```
;
;
;     HWIND.ASM
;
;
;     AUTHOR: Daryl R. Mair
;
;     DATE:   SEPT 5, 1984
;
;
;This microprogram provides a window for the Dynamic Hud

;Roll/Pitch Ladder data base, which enables only the
;portion of the data base that is to be displayed to be
;updated and rotated.
;
;
;     CONTROL ADDRESSES
;
;2025400=Address of Input Buffer
;2025401=Address of Output Buffer
;2025402=Negative Word Count
;2025403=Adjusted Pitch value
;2025404=Pixel address adjustment
;
DEFAULT NANOP CCNOP LDNOP SB ALUZ YD CARØ SSØ ALUBR MDRIKD
    MARIKA
;
ORG    500
;
;
;     TRANSFER CONTROL WORDS TO WORKING REGISTERS
;
;
BEGIN:   LDUDR   202
    RIMM   ALUMAR   PR   5400          ;MAR CONTAINS 202/5400
    IKRD   IKBR     B0   BD             ;READ CONTENTS INTO R0
    ;
    LDUDR  202
    RIMM   ALUMAR   PR   5401          ;MAR CONTAINS 202/5401
    IKRD   IKB  R   B1   BD            ;READ CONTENTS INTO R1
    ;
    LDUDR  202
    RIMM   ALUMAR   PR   5402          ;MAR CONTAINS 202/5402
    IKRD   IKBR     B2   BD            ;READ CONTENTS INTO R2
    ;
    LDUDR  202
    RIMM   ALUMAR   PR   5403          ;MAR CONTAINS 202/5403
    IKRD   IKBR     B3   BD            ;READ CONTENTS INTO R3
    ;
```

```
;
LDUDR  2Ø2
RIMM   ALUMAR   PR   54Ø4          ;MAR CONTAINS 2Ø2/54Ø4
IKRD   IKBR     B4   BD            ;READ CONTENTS INTO R4
;
;    ADD PITCH TO BASE ADDRESS
;
RAØ    RPS      B3   BD            ;RØ + R3, RESULT IN R3
;
;    READ DATA INTO MDR
;
RETDW: RA3   ALUMAR   PR           ;ADDRESS IN R3 TO MAR
IKRD   IKMDR                       ;MDR HAS DATA IN ADDR R3
;
;    ADD PIXEL ADJUST TO Y
;
RMDR   ALUMDR    RPS   B4
;
;    WRITE TO OUTPUT ADDRESS
;
RA1    ALUMAR    PR               ;ADDRESS IN R1 TO MAR
IKWR                              ;WRITE TO ADDR R1
;
;    ADD ONE TO NEGATIVE WORD COUNT
;
RA2    PR    CAR1    B2    BD
;
;
;    INCRAMENT ADDRESS POINTERS
;
RA1    PR    CAR1    B1    BD     ;INC OUTPUT ADDR R1
;
RA3    PR    CAR1    B3    BD     ;INC INPUT ADDR R3
;
;    READ DATA INTO MDR
;
RA3   ALUMAR   PR                 ;ADDRESS IN R3 TO MAR
IKRD  IKMDR                       ;MDR HAS DATA IN ADDR R3
;
;    WRITE TO OUTPUT ADDRESS
;
RA1   ALUMAR   PR                 ;ADDRESS IN R1 TO MAR
IKWR                              ;WRITE TO ADDR R1
;
;    ADD ONE TO NEGATIVE WORD COUNT
;
CCOVR   RA2   PR   CAR1   B2   BD
;
;    CHECK FOR OVERFLOW
;
CCOVR   NARETN          ;IF OVERFLOW RETURN SUBROUTINE
;
;
```

```
     ;
     ;     INCRAMENT ADDRESS POINTERS
     ;
     RA1    PR    CAR1    B1    BD        ;INC OUTPUT ADDR R1
     ;
     RA3    PR    CAR1    B3    BD        ;INC INPUT ADDR R3
     ;
     ;     CHECK FOR BUS STATUS
     ;
DELAY:  CCBUSAC    JMPDF    DELAY         ;WAIT FOR IK BUS READY?
     NANOP
     JMPDF    RETDW                       ;GO TO TRANSFER ANOTHER.
     ;
     ;
     END
```

APPENDIX A-3

```
;
;
; CGMS.ASM
;
;
;
;     PROGRAM TO CONTROL THE CGM FROM THE BPS32
;
;
;     DATE:    2 NOV 1984
;
;
; THIS IS A MODIFICATION OF CGMMCR WRITTEN BY C.A.
; SAMPAYO TO BE INCORPORATED AS A SUBROUTINE OF HMCP
; AND THE DYNAMIC HUD ROUTINES.
;
;
;2020=POINTER TO <PIXEL ADDRESS> LIST
;2021=POINTER TO CGMCB <PIXEL ADDRESS> WORD
;2022=POINTER TO WORKING BUFFER
;2023=POINTER TO <CHARACTER STRINGS> LIST
;2024=SIZE OF CHARACTER STRINGS IN IKONAS WORDS
;2025=ONE'S COMPLEMENT OF THE NUMBER OF STRINGS
;2026=CGMBCB <CONTROL> WORD
;2027=CGMBCB <ADDRESS OF CGMCB> WORD
;
;
ORG     600
;
;
DEFAULT  NANOP CCNOP LDNOP SB ALUZ YD  CAR0 SS0 ALUBR MDRIKD
    MARIKA
    ;
START:   LDUDR   202
    ;
    RIMM    ALUMAR    PR    0     ;MAR CONTAINS 2020
    IKRD    IKBR      B0    BD    ;READ CONTENTS OF 2020 INTO R0
    ;
    LDUDR   202
    RIMM    ALUMAR    PR    1     ;MAR CONTAINS 2021
    IKRD    IKBR      B1    BD    ;READ CONTENTS OF 2021 INTO R1
    ;
    LDUDR   202
    RIMM    ALUMAR    PR    2     ;MAR CONTAINS 2022
    IKRD    IKBR      B2    BD    ;READ CONTENTS OF 2022 INTO R2
    ;
    LDUDR   202
    RIMM    ALUMAR    PR    3     ;MAR CONTAINS 2023
    IKRD    IKBR      B3    BD    ;READ CONTENTS OF 2023 INTO R3
    ;
```

```
;
;
LDUDR  2Ø2
RIMM   ALUMAR   PR   4     ;MAR CONTAINS 2Ø24
IKRD   IKBR     B4   BD    ;READ CONTENTS OF 2Ø24 INTO R4
;
LDUDR  2Ø2
RIMM   ALUMAR   PR   5     ;MAR CONTAINS 2Ø25
IKRD   IKBR     B5   BD    ;READ CONTENTS OF 2Ø25 INTO R5
;
RA2    ALUMAR   PR        ;MAR CONTAINS ADDRESS IN R2
;
LDUDR  Ø
RIMM   ALUMDR   PR   Ø     ;MDR CONTAINS ØØ
IKWR                       ;WRITE TO IKBUS
;
RA2    ALUMAR   PR   CAR1 ;MAR CONTAINS ADDR R2 PLUS 1
IKWR                       ;WRITE MDR TO IKBUS
;
RA2    ALUMAR   PR   CAR1 ;MAR CONTAINS ADDR R2 PLUS 2
IKWR                       ;WRITE MDR TO IKBUS
;
GOCGM: RAØ   ALUMAR   PR   ;MAR CONTAINS ADDRS IN RØ
    IKRD   IKMDR    ;MDR HAS DATA IN ADDRS POINTED BY RØ
    ;
    RA1    ALUMAR   PR        ;MAR CONTAIN ADDRS IN R1
    IKWR                      ;WRITE TO IKBUS
    ;
    RA3    ALUMAR   PR        ;MAR CONTAINS ADDRS IN R3
    IKRD   IKMDR    ;MDR HAS DATA IN ADDRS POINTED BY R3
    ;
    RA2    ALUMAR   PR        ;MAR CONTAINS ADDRS IN R2
    IKWR                      ;WRITE TO IKBUS
    ;
    RA3    ALUMAR   PR   CAR1 ;MAR CONTAINS ADDRS R3 PLUS 1
    IKRD   IKMDR   ;MDR CONTAINS DATA IN ADDRS POINTED BY R3
    ;
    RA2    ALUMAR   PR   CAR1 ;MAR CONTAINS ADDRS R2 PLUS 1
    IKWR                      ;WRITE TO IKBUS
    ;
    LDUDR  2Ø2
    RIMM   ALUMAR   PR   7     ;MAR CONTAINS 2Ø27
    IKRD   IKMDR               ;MDR CONTAINS DATA IN 2Ø27
    ;
    ;
    ;
    ;
    LDUDR  2Ø6
    RIMM   ALUMAR   PR   1     ;MAR CONTAINS 2Ø61
    IKWR                       ;WRITE TO IKBUS
    ;
    LDUDR  2Ø2
    RIMM   ALUMAR   PR   6     ;MAR CONTAINS 2Ø26
```

```
    IKRD   IKMDR                ;MDR CONTAINS DATA IN 2026
    ;
    LDUDR  206
    RIMM   ALUMAR   PR   Ø      ;MAR CONTAINS 2060
    IKWR                        ;WRITE TO IKBUS
    ;
    NANOP
    ;
    RA4    RPS      B3   BD     ;ADD R4 TO R3 , RESULT IN R3
    ;
    RAØ    PR   CAR1   BØ   BD
    ;
    CCOVR  RA5   PR    CAR1 B5   BD
    ;
    CCOVR  JMPDF   IDLE2
    ;
WAIT:  CCBUSAC   JMPDF   WAIT ;WAIT FOR CGM NOT BUSY
    ;
    CCBUSAC   JMPDF   WAIT
    ;
    NANOP
    ;
    JMPDF   GOCGM
    ;
    NANOP
    ;
IDLE2:  NARETN      ;ALU OVERFLOWED RETURN FROM SUBROUTINE
    ;
    END
```

APPENDIX B

HOST CONTROL PROGRAM (HCP.FTN)

LISTINGS

```
C
C HOST CONTROL PROGRAM (HCP.FTN)
C
C AUTHOR: Daryl R. Mair
C
C DATE: JULY 3, 1984
C
C THIS PROGRAM WILL UPDATE AND DISPLAY THE DYNAMIC FEATURES
C OF A HEAD UP DISPLAY.
C
C This program generates and updates a Head Up Display
C (using the Ikonas frame buffer, microprocessor,and
C character generator module) for air to ground
C applications in the VTRS CTOL dome.
C The program generates Flight Control Data and updates the
C alphanumeric parameters based on data received from
C the Simulator's visual computer (SEL 32/75).
C
C
C To compile:
C
C FOR/F4P/NOTRACEBACK/NOWARNING/CONTINUATIONS:1 DCHUD
C
C To build the task:
C
C LINK/CODE:FPP/PR:4/OPT DCHUD,[1,1]SYSLIB/LIBRARY
C /INCLUDE:$SHORT
C
C OPTION? ACTFIL=2
C OPTION? UNITS=5
C OPTION? STACK=64
C    Page Address Field in PAR Reg.
C OPTION? VSECT=PAGADF:172350:2:1
C    Sel-Dec Interface and Control
C OPTION? VSECT=IFDR11:172450:10:1
C    Ikonas Interface and Control
C OPTION? VSECT=IKDR11:172460:16:1
C OPTION?

C
C
      DIMENSION   INIBUF(54),IDATA(4,17),ICOEF(12),VWPBUF(64)
      DIMENSION   CORBUF(14,2)
C
      BYTE        INNBUF(256),OUTBUF(112),INAME(25)
C
      DOUBLE PRECISION     CGDAT(14)
C
      INTEGER      VIRTAD(3),PAF,REG(2),PHYADL(3),PHYADH(3)
C
      INTEGER      SIZE(15)
C
      INTEGER*4    REG21,IINC,TEMBUF(15)
```

```
C
     EQUIVALENCE  (OUTBUF(1),CGDAT(1)),(INIBUF(27),TEMBUF(1))
C
     EQUIVALENCE  (REG21,REG(1)),(VWPBUF(1),INNBUF(1))
C
C
     COMMON /PAGADF/PAF   !Page address field in PAR register
C
     COMMON /IFDR11/IFDRWC,IFDRBA,IFDRCS,IFDRDR
C
     COMMON /IKDR11/IKDRWC,IKDRBA,IKDRCS,IKDRDR,
    *IKSLAD,IKSHAD,IKSCSR
C
C
C  Calculate physical addresses for virtual array VIRTAD
C  store low 16-bits in array PHYADL, and extension bits
C  shifted for use by DR11B's in array PHYADH
C  the following data definitions are required:
C
C  INTEGER        VIRTAD(2),PAF,REG(2),PHYADL(2),PHYADH(2)
C  INTEGER*4      REG21
C  EQUIVALENCE    (REG21,REG(1))
C
C  COMMON    /PAGADF/PAF !Page address field in PAR register
C
C  The system subroutine GETADR is used VIRTAD is the
C  output array of virtual addresses,DATA1(1),DATA2(1),
C  etc are the variables whose addresses are desired.
C
C         CALL    GETADR(VIRTAD,DATA1(1),DATA2(1))
C
C
C         START .MAIN.
C
          DO 10 I=1,30000
          THIS=WAIT
10        CONTINUE
C
          CALL   GETADR(VIRTAD,INIBUF(1),INNBUF(1),CGDAT(1))
C
 DO 15 I=1,3                !NUMBER OF ADDRESSES
 REG21=0                    !CLEAR WORKING REGISTER
       REG(1)=VIRTAD(I) !PUT VIRTUAL ADDR IN LOWER WORKING
 REG
 REG21=8*REG21              !SHIFT APF TO UPPER WORKING REG
 REG(2)=0                   !APF NOT NEEDED, APR DEFINED
          REG21=128*REG21   !BLOCK NUMBER INTO UPPER WORKING REG
          REG(2)=REG(2)+PAF !PAGE ADDRESS FIELD + BLOCK NUMBER
          REG21=REG21/1024          !PLACE BACK TO POSITION
 PHYADH(I)=16*REG(2)        !UPPER 2 BITS OF ADDRESS <17:16>
 PHYADL(I)=REG(1)           !LOWER 16 BITS OF ADDRESS <15:0>
15        CONTINUE
C
```

```
 TYPE *,'PHYSICAL ADDRESSES FOR INIBUF,INNBUF,OUTBUF'
 TYPE *,' '
 TYPE 1000,PHYADH(1),PHYADL(1)
 TYPE 1000,PHYADH(2),PHYADL(2)
 TYPE 1000,PHYADH(3),PHYADL(3)
C
 INITWC=-54
 INWC=-128                !NEG WORD COUNT FOR INNBUF FROM SEL.
 IOUTWC=-56
C
 INCS=PHYADH(2)+3         !GO AND TRANSFER MODE BITS.
 INITCS=PHYADH(1)+1
 IOUTCS=PHYADH(3)+1
C
C
 IKSCSR="177777           !RESET IKONAS
 IKSCSR="0
C
 IKSLAD="000005
 IKSHAD="030000
 IKSCSR="0020
 IKDRDR="000041           !ALLOW AUTO ERASE
 IKDRDR="000060
C
C
C
 W=32767
 RAD=57.29578
C
 TYPE *,'ENTER VERTICAL DISPLACEMENT FOR CHARACTER DATA'
 ACCEPT *,IDPL
C
 TYPE *,'ENTER VERTICAL DISPLACEMENT FOR ROLL/PITCH LADDER'
 ACCEPT *,IDFR
C
 TYPE *,' DATA BASE NAME?'
 ACCEPT 500,INAME
C
 TYPE *,' ENTER SCALING FACTOR'
 ACCEPT *,SCALE
C
C
 OPEN (UNIT=1,NAME=INAME,TYPE='OLD')
C
 READ (1,*) K
C
C
C        TYPE *,K
C
 IKSLAD="500
 IKSHAD="20200
 IKSCSR="420
C
```

```
 DO 70 J=1,K
C
C
 READ (1,*) FACE,IEDGE
 READ (1,*) IDUMMY
C
 DO 40 I=1,IEDGE
C
 READ (1,*) DATA1,DATA2,DATA3
C
 IDATA(1,I)=DATA1*SCALE+.5
 IDATA(2,I)=DATA2*SCALE+.5
 IDATA(3,I)=DATA3*SCALE+.5
C
40       CONTINUE
C
C
 READ (1,*) DUMMY
C
 L=IEDGE
 IDATA(4,1)="037777
 DO 50 N=2,L
 IDATA(4,N)="137777
50       CONTINUE
 IF (J .EQ. K) IDATA(4,L)="177777
C
C
 DO 60 N=1,L
 IKDRDR=IDATA(1,N)
 IKDRDR=IDATA(2,N)
 IKDRDR=IDATA(3,N)
 IKDRDR=IDATA(4,N)
60       CONTINUE
C
70       CONTINUE
C
 CLOSE (UNIT=1)
C
C        CGMS CONTROL BLOCK
C
 INIBUF(1)="15        ! Pointer to pixel address list
 INIBUF(2)="202
 INIBUF(3)="13        ! Pointer to pixel address word in CGMCB
 INIBUF(4)="202
 INIBUF(5)="200       ! Pointer to active character string
 INIBUF(6)="202
 INIBUF(7)="60        ! Pointer to character strings list
 INIBUF(8)="202
 INIBUF(9)="2 ! Size of one character string in Ikonas words
 INIBUF(10)="0
 INIBUF(11)="177762 !2's complement of the number of strings
 INIBUF(12)="77777  !sign bit must be zero
C
```

```
C
C        CHARACTER GENERATOR CONTROL BLOCK
C
 INIBUF(13)="421          ! Control word
 INIBUF(14)="0
 INIBUF(15)="10           ! Address of CGMCB
 INIBUF(16)="202
C
C
C        CHARACTER GENERATOR CHARACTER CONTROL (CGMCB)
C
 INIBUF(17)="0            ! Character string control
 INIBUF(18)="4010
 INIBUF(19)="177400       ! Character string color
 INIBUF(20)="0
 INIBUF(21)="0            ! Background color
 INIBUF(22)="0
 INIBUF(23)="0            ! Starting pixel address
 INIBUF(24)="0
 INIBUF(25)="200          ! Character string address
 INIBUF(26)="202
C
C
C        PIXEL ADDRESS LIST
C
C
72      CORBUF(1,1)=168*2          ! `G' low res screen pixel
 CORBUF(1,2)=(292+IDPL)*2   ! coordinates: X and Y
 CORBUF(2,1)=160*2          ! alpha
 CORBUF(2,2)=(308+IDPL)*2
 CORBUF(3,1)=168*2          !`M'
 CORBUF(3,2)=(324+IDPL)*2
 CORBUF(4,1)=356*2          ! `FPM'
 CORBUF(4,2)=(292+IDPL)*2
 CORBUF(5,1)=312*2          ! `RKTS'
 CORBUF(5,2)=(308+IDPL)*2
 CORBUF(6,1)=264*2          ! `SAFE'
 CORBUF(6,2)=(252+IDPL)*2
 CORBUF(7,1)=260*2          ! mills
 CORBUF(7,2)=(200+IDPL)*2
 CORBUF(8,1)=245*2          !heading
 CORBUF(8,2)=(149+IDPL)*2
 CORBUF(9,1)=176*2          !speed
 CORBUF(9,2)=(216+IDPL)*2
 CORBUF(10,1)=308*2         !altitude
 CORBUF(10,2)=(216+IDPL)*2
 CORBUF(11,1)=184*2         ! g's
 CORBUF(11,2)=(292+IDPL)*2
 CORBUF(12,1)=184*2         !angle of attack
 CORBUF(12,2)=(308+IDPL)*2
 CORBUF(13,1)=184*2         ! mach
 CORBUF(13,2)=(324+IDPL)*2
 CORBUF(14,1)=296*2         ! vertical speed
```

```
 CORBUF(14,2)=(292+IDPL)*2
C
 DO 73 I=1,14
 TEMBUF(I)=CORBUF(I,1)+CORBUF(I,2)*1024
73       CONTINUE
C
C        SEND INITIALIZATION BUFFER TO IKONAS ADDRESS 20200$0
C
 IKSLAD="0
 IKSHAD="20200
 IKSCSR="520
 IKDRWC=INITWC
 IKDRBA=PHYADL(1)
 IKDRCS=INITCS
C
C        WAIT FOR COMPLETION HERE
C
C
74       M=IAND(IKDRCS,"200)
 IF (M .EQ. 0)GO TO 74
C
C
C        SET UP TEST CHARACTERS
C
C
 CGDAT(1)='G'
 OUTBUF(9)=5                ! Alpha ascii designations in
 OUTBUF(10)=6               ! the Ikonas resident font memory
 CGDAT(3)='M'
 CGDAT(4)='FPM'
 CGDAT(5)='RKTS'
 CGDAT(6)='SAFE'
 CGDAT(7)='140'
 CGDAT(8)='000'
 CGDAT(9)='720'
 CGDAT(10)='10533'
 CGDAT(11)='0.4'
 CGDAT(12)='1.0'
 CGDAT(13)='0.78'
 CGDAT(14)='-12570'
C
C        LIMIT STATIC FIELDS
C
 OUTBUF(2)=0      ! 'G'
 OUTBUF(11)=0     ! alpha
 OUTBUF(18)=0     ! 'M'
 OUTBUF(28)=0     ! 'FPM'
C
C       OUTPUT BUFFER SIZES
C
 SIZE(1)=4        !Mils = 3
 SIZE(2)=4        !Heading = 3
 SIZE(3)=4        !Air Speep=3
```

```
 SIZE(4)=2          !Altitude = 5
 SIZE(5)=4          !G's = 3
 SIZE(6)=3          !Angle of Attack = 4
 SIZE(7)=3          !M = 4
 SIZE(8)=2          !Vertical Speed = 5
C
C
C         HEADING BOX PIXEL ADDRESSES
C
 IDATA(1,1)=236*2
 IDATA(2,1)=2*(144+IDPL)
C
 IDATA(1,2)=277*2
 IDATA(2,2)=IDATA(2,1)
C
 IDATA(1,3)=IDATA(1,2)
 IDATA(2,3)=2*(160+IDPL)
C
 IDATA(1,4)=IDATA(1,1)
 IDATA(2,4)=IDATA(2,3)
C
 IDATA(1,5)=IDATA(1,1)
 IDATA(2,5)=IDATA(2,1)
C
C         AIRSPEED BOX PIXEL ADDRESSES
C
 IDATA(1,6)=2*168
 IDATA(2,6)=2*(212+IDPL)
C
 IDATA(1,7)=2*216
 IDATA(2,7)=IDATA(2,6)
C
 IDATA(1,8)=IDATA(1,7)
 IDATA(2,8)=2*(228+IDPL)
C
 IDATA(1,9)=IDATA(1,6)
 IDATA(2,9)=IDATA(2,8)
C
 IDATA(1,10)=IDATA(1,6)
 IDATA(2,10)=IDATA(2,6)
C
C         ALTITUDE BOX PIXEL ADDRESS LIST
C
 IDATA(1,11)=2*304
 IDATA(2,11)=2*(212+IDPL)
C
 IDATA(1,12)=2*357
 IDATA(2,12)=IDATA(2,11)
C
 IDATA(1,13)=IDATA(1,12)
 IDATA(2,13)=2*(228+IDPL)
C
 IDATA(1,14)=IDATA(1,11)
```

```
 IDATA(2,14)=IDATA(2,13)
C
 IDATA(1,15)=IDATA(1,11)
 IDATA(2,15)=IDATA(2,11)
C
 IDDAT1="1000
 IDDAT2="037777
 IDDAT3="137777
C
C        SEND HUD DISPLAY BOXES TO OUTPUT BUFFER
C
 IKSLAD="500
 IKSHAD="20203
 IKSCSR="420
C
 DO 80 J=1,11,5
C
 IKDRDR=IDATA(1,J)
 IKDRDR=IDATA(2,J)
 IKDRDR=IDDAT1
 IKDRDR=IDDAT2
C
 DO 75 I=J+1,J+4
C
 IKDRDR=IDATA(1,I)
 IKDRDR=IDATA(2,I)
 IKDRDR=IDDAT1
 IKDRDR=IDDAT3
C
75       CONTINUE
80       CONTINUE
C
C       INITIALIZE FOR START OF REALTIME LOOP
C
 IBOX=50         !Size of Display box buffer
 IKDAT1="0
 IKDAT2="100
 IKDAT3="6500
 IKDAT4="202     !Ikonas SR8 memory locations
 IKDAT5="5500
 IKDAT6="500
C
C       HMCP CONTROL BLOCK
C
 IKSLAD="470
 IKSHAD="20200
 IKSCSR="420
C
 IKDRDR=IKDAT2
 IKDRDR=IKDAT1
 IKDRDR=IKDAT3
 IKDRDR=IKDAT4
 IKDRDR=IKDAT1
```

```
 IKDRDR=IKDAT1
C
C        START OF REALTIME MODULE
C
C
C        INPUT BUFFER FROM SEL
C
100       IFDRWC=INWC
 IFDRBA=PHYADL(2)
 IFDRCS=INCS
C
C        WAIT FOR TRANSFER COMPLETE
C
120       M=IAND(IFDRCS,"200)
 IF(M .EQ. 0)GO TO 120
C
C
C         PROCESS VELOCITY VECTOR CALCULATION
C
 IXVEL=2*256
 IYVEL=(256+IDFR)*2
C
 IDATA(1,1)=(IXVEL-20)
 IDATA(2,1)=IYVEL
C
 IDATA(1,2)=(IXVEL-10)
 IDATA(2,2)=IYVEL
C
 IDATA(1,3)=(IXVEL)
 IDATA(2,3)=IYVEL+10
C
 IDATA(1,4)=IXVEL+10
 IDATA(2,4)=(IYVEL)
C
 IDATA(1,5)=(IXVEL+20)
 IDATA(2,5)=IYVEL
C
 IDATA(1,6)=IXVEL+10
 IDATA(2,6)=IYVEL
C
 IDATA(1,7)=IXVEL
 IDATA(2,7)=IYVEL-10
C
 IDATA(1,8)=IXVEL
 IDATA(2,8)=IYVEL-20
C
 IDATA(1,9)=IXVEL
 IDATA(2,9)=IYVEL-10
C
 IDATA(1,10)=IXVEL-10
 IDATA(2,10)=IYVEL
C
C
```

```
C
C
C
C         SEND POINTS TO IKONAS WORKING MEMORY
C
 IKSLAD="536
 IKSHAD="20203
 IKSCSR="1420
C
 IKDRDR=IDATA(1,1)
 IKDRDR=IDATA(2,1)
 IKDRDR=IDDAT1
 IKDRDR=IDDAT2
C
 DO 148 I=2,10
 IKDRDR=IDATA(1,I)
 IKDRDR=IDATA(2,I)
 IKDRDR=IDDAT1
 IKDRDR=IDDAT3
148       CONTINUE
C
C
C         PROCESS ROLL/PITCH LADDER DATA!
C
 IINC=VWPBUF(37)+.5       !BUFFER FOR VIEW POINT PITCH
 ROLL=VWPBUF(38)          !BUFFER FOR VIEW POINT ROLL
C
 IFRACT=IINC/5
 DIFF=(VWPBUF(37)-IFRACT*5)*SCALE*6.4
C
150       IF (IFRACT .LT. 0)GO TO 200
C
160       IKPTCH=956-(IFRACT*52)
 IWDCT=260
 IF (IFRACT .EQ. 0)IWDCT=212
 IF (IFRACT .EQ. 1)IWDCT=196
 IF (IFRACT .EQ. 2)IWDCT=200
 IF (IFRACT .EQ. 3)IWDCT=240
 IF (IFRACT .LT. 18)GO TO 300
C
 ROLL=ROLL+180
 IFRACT=IFRACT-18
 IFRACT=18-IFRACT
 IF (IFRACT .EQ. 18)GO TO 270
 GO TO 160
C
200       IKPTCH=(1100-(IFRACT+4)*68)
 IWDCT=340
C
 IF (IFRACT .EQ. -1)GO TO 210
 IF (IFRACT .EQ. -2)GO TO 220
 IF (IFRACT .EQ. -3)GO TO 230
 IF (IFRACT .GT. -18)GO TO 300
```

```
C
 ROLL=ROLL+180
 IFRACT=IFRACT+18
 IFRACT=-18-IFRACT
 IF (IFRACT .EQ. -18)IWDCT=360
 IF (IFRACT .EQ. -18)GO TO 300
 GO TO 200
C
210      IWDCT=228
 IKPTCH=1008
 GO TO 300
C
220      IWDCT=264
 IKPTCH=1040
 GO TO 300
C
230      IWDCT=320
 IKPTCH=1052
 GO TO 300
C
270      IWDCT=280
 IKPTCH=0
C
300  DR=0                                        !-PITCH
 DP=0                                            !-YAW
 DY=ROLL                                         !ROLL
C
 R=DR/RAD
 P=DP/RAD
 Y=DY/RAD
C
 CR=COS(R)
 SR=SIN(R)
 CP=COS(P)
 SP=SIN(P)
 CY=COS(Y)
 SY=SIN(Y)
C
 IDIFF=CY*DIFF+.5                                !Y DEFRACTION
 IDIFF1=SY*DIFF+.5                               !X DEFRACTION
C
 ICOEF(1)=(CP*CY)*W+.5
 ICOEF(2)=(SR*SP*CY+CR*SY)*W+.5
 ICOEF(3)=(SR*SY-CR*SP*CY)*W+.5
 ICOEF(4)=512+IDIFF1
 ICOEF(5)=(-CP*SY)*W+.5
 ICOEF(6)=(CR*CY-SR*SP*SY)*W+.5
 ICOEF(7)=(CR*SP*SY+SR*CY)*W+.5
 ICOEF(8)=512+IDFR+IDIFF
 ICOEF(9)=SP*W+.5
 ICOEF(10)=(-SR*CP)*W+.5
 ICOEF(11)=(CR*CP)*W+.5
 ICOEF(12)=512
```

```
C
 IF ((.NOT.INNBUF(3)).OR.(INNBUF(7)))  CGDAT(6)='SAFE'
C
 IF (INNBUF(3).AND.(.NOT.INNBUF(7))) CGDAT(6)='    '
C
 IF (INNBUF(4)) CGDAT(5)='BOMB'
C
 IF (INNBUF(5)) CGDAT(5)='GUNS'
C
 IF (INNBUF(6)) CGDAT(5)='RKTS'
C
C
C        MOVE DYNAMIC CHARACTERS FROM INPUT TO OUTPUT BUFFER

C       AND RIGHT JUSTIFYING
C
 DO 304 I=49,112
 OUTBUF(I)=0
304      CONTINUE
 INDCNT=0
 DO 312 I=1,65,8
 INDCNT=INDCNT+1
 INDOUT=48
 INDINN=16
 INDOUT=INDOUT+I
 INDINN=INDINN+I
 IOBYT=INDOUT
 OUTBUF(INDOUT)=INNBUF(INDINN)
 INBYT=INDINN+SIZE(INDCNT)
 DO 308 J=SIZE(INDCNT),6
 INBYT=INBYT+1
 IOBYT=IOBYT+1
 OUTBUF(IOBYT)=INNBUF(INBYT)
308      CONTINUE
312      CONTINUE
C
C
C
C        NORMAL FLIGHT CONTROL PROCESSING
C
C
C        LIMIT DYNAMIC FIELDS
C
 OUTBUF(37)=0     ! 'RKTS'
 OUTBUF(45)=0     ! 'SAFE'
 OUTBUF(53)=0     ! mils
 OUTBUF(61)=0     ! heading
 OUTBUF(69)=0     ! air speed
```

```
 OUTBUF(79)=Ø     ! altitude
 OUTBUF(85)=Ø     ! g's
 OUTBUF(95)=Ø     ! angle attack
 OUTBUF(1Ø2)=Ø    ! mach
 OUTBUF(112)=Ø    ! vertical speed
C
C         SEND DATA TO IKONAS ADDRESS 2Ø2ØØ$6Ø
C
 IKSLAD="6Ø
 IKSHAD="2Ø2ØØ
 IKSCSR="52Ø
 IKDRWC=IOUTWC
 IKDRBA=PHYADL(3)
 IKDRCS=IOUTCS
C
C         WAIT FOR COMPLETION HERE
C
32Ø       M=IAND(IKDRCS,"2ØØ)
 IF (M .EQ. Ø)GO TO 32Ø
C
C
C
C         WAIT FOR NEXT FIELD HERE
C
322       M=IAND(IKSCSR,"1ØØØØ)
 IF (M .EQ. Ø) GO TO 322
C
C        SEND COEF MATRIX TO MA1Ø24
C
 IKSLAD="Ø
 IKSHAD="2Ø4ØØ
 IKSCSR="46Ø
 DO 325 I=1,12
 IKDRDR=ICOEF(I)
325       CONTINUE
C
C        SEND MMEOL CONTROL BLOCK TO MA1Ø24
C
 IKSLAD="Ø
 IKSHAD="2Ø4Ø2
 IKSCSR="42Ø
 IKDRDR=IKDAT1
 IKDRDR=IKDAT1
 IKDRDR=IKDAT5
 IKDRDR=(IKDAT3+IBOX)
C
C        HWIND CONTROL BLOCK TRANSFER
C
 IKDAT9=IFRACT*32*SCALE
 IKNGWC=-IWDCT
 IKDATØ="77777
C
 IKSLAD="14ØØ
```

```
 IKSHAD="20202
 IKSCSR="420
C
 IKDRDR=IKDAT6
 IKDRDR=IKDAT4
 IKDRDR=IKDAT5
 IKDRDR=IKDAT4
 IKDRDR=IKNGWC
 IKDRDR=IKDAT0
 IKDRDR=IKPTCH
 IKDRDR=IKDAT1
 IKDRDR=IKDAT1
 IKDRDR=IKDAT9
C
 IKSCSR="3000
 THIS=IWAIT
 THIS=IWAIT
 IKSCSR="1000
C
 GO TO 100
C
500      FORMAT (25A1)
1000     FORMAT (' ',O6,' ',O6)
C
 END
```

REFERENCES

Advanced Micro Devices. Bipolar Microprossor Logic and Interface Data Book. Sunnyvale, CA: AMD, 1981. pp. 6-40 - 6-102.

Ikonas Inc. Hardware Configuration for Ikonas RDS-3000. Raleigh, NC: 1980.

Ikonas Inc. Ikonas Processor Programming Manual. Raleigh, NC: 1980.

McDonnell Douglas Corp. Procurement Specification for Head-Up Display Unit. St. Louis, MO: McDonnell Douglas, 1980.

Navy, U.S. Navtops Flight Manual, AV-8B. Washington, D.C.: 1982.

Newman, W. M.; Sproull R. F. Princples of Interactive Computer Graphics. New York: Mcgraw-Hill Co., 1979. pp. 333-352.

Rogers, D.F.; Adams, J.A. Mathmatical Elements for Computer Graphics. New York: McGraw-Hill Co., 1976. pp. 46-59, 206-211.

Sampayo, C.A. The Ikonas CGMMCR Microcode Program. Orlando FL: Naval Training Equipment Center. 1982.