

NASA Contractor Report 178210

USER GUIDE FOR THE DIGITAL CONTROL SYSTEM
OF THE NASA/LANGLEY RESEARCH CENTER'S 13-INCH
MAGNETIC SUSPENSION AND BALANCE SYSTEM

(NASA-CR-178210) USER GUIDE FOR THE DIGITAL
CONTROL SYSTEM OF THE NASA/LANGLEY RESEARCH
CENTER'S 13-INCH MAGNETIC SUSPENSION AND
BALANCE SYSTEM Final Report, 23 Oct. 1985 -
30 Sep. 1986 (Old Dominion Univ.) 81 p

N87-18574

Unclas
43613

G3/09

Colin P. Britcher

OLD DOMINION UNIVERSITY RESEARCH FOUNDATION
Norfolk, Virginia

Contract NAS1-17993
Task Authorization No. 24
March 1987



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

	Page
1. INTRODUCTION.....	1
2. CONTROLLER CONCEPTS.....	5
2.1 Control Algorithms.....	5
2.2 Software Block Structure.....	8
2.3 Data Array Format.....	11
2.4 I/O Filtering.....	13
3. SYSTEM CONFIGURATION.....	15
3.1 Position Sensors.....	15
3.2 Isolation Amplifiers and Current Monitoring.....	15
3.3 PDP 11/23+.....	22
4. CONTROLLER USERS GUIDE.....	23
4.1 Hardware.....	23
4.1.1 Startup.....	23
4.1.2 Closedown.....	23
4.2 Controller Software Operation.....	23
4.2.1 Activation of Control Program.....	23
4.2.2 Run-time Screen Display.....	24
4.2.3 Run-time Keyboard Commands.....	25
4.2.4 Launch and Landing Sequence.....	28
4.3 Limitations and Restrictions.....	29
APPENDIX A Compilation, Assembly and Linking of Executable Program.....	31
APPENDIX B Example Source Code Listings.....	34
APPENDIX C PDP 11/23 Configuration Reference.....	63
APPENDIX D Hardware Checkout and Test Procedures.....	65
REFERENCES.....	77

1. INTRODUCTION

Until relatively recently, all wind tunnel Magnetic Suspension and Balance Systems (MSBS) relied exclusively on analogue controllers, of rather standardized form, to stabilize and control the position and attitude of the suspended model (reference 1). In the mid 1970's, MIT began an investigation of the feasibility of converting the control system of their 6 inch MSBS to digital operation, though experimental work progressed only as far as a 1 degree of freedom benchtop demonstration (reference 2). Later, the University of Southampton MSBS controller was progressively digitized and can now only be operated in that form, with no provision for analogue reversion (references 3,4). Around 1980, the AEDC 13 inch MSBS was relocated to NASA Langley Research Center (LaRC) and was subsequently turned over to NASA ownership. Commencing in 1984, this system, hereafter referred to as the NASA LaRC 13 inch MSBS, has been digitized, following fairly closely the methods explored at Southampton.

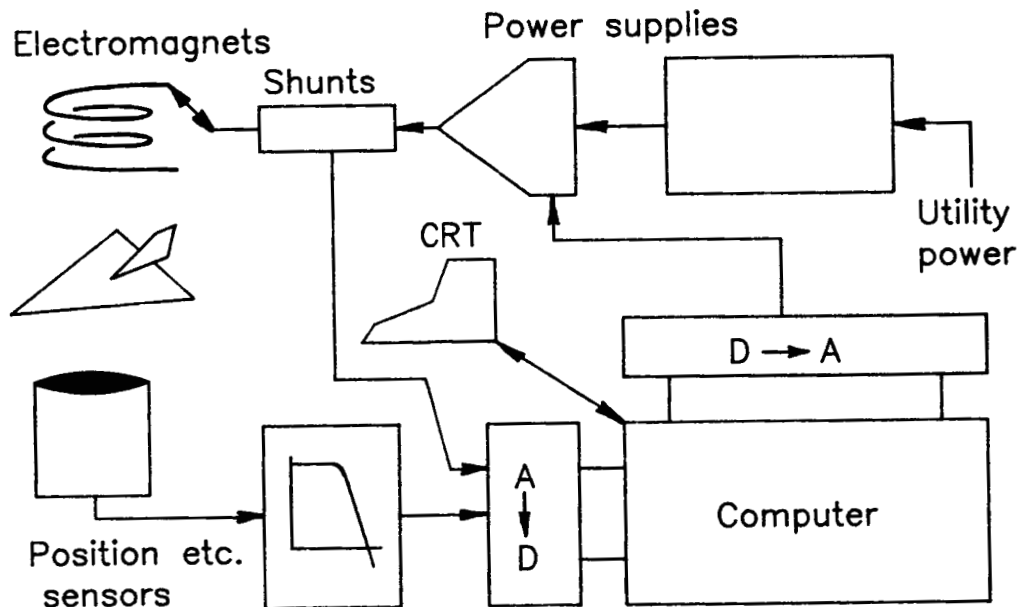


Fig.1.1 Typical block diagram of digitally controlled MSBS

The original motives for investigation of digital MSBS controllers included :

- 1) Incorporation of more advanced control algorithms. All MSBSs, including those operated with digital controllers, have

utilized broadly similar control strategies. There is no reason to believe that these strategies represent the best performance that can be achieved and it is strongly felt that design, implementation, testing and evaluation of new algorithms will be far easier within a digitized controller than otherwise. Advanced control strategies are likely to include some self-adaptive features.

2) Model changes. Controller coefficients (loop gains etc.) generally need to be reset or readjusted for each substantially different test model. This function is most easily implemented via accessing different software modules, rather than by mechanical or electrical adjustments.

3) Performance repeatability. Digitized controller sections do not suffer from drift, electrical interference etc., and retain accurate calibrations over unlimited time periods.

4) Versatility of model position and attitude selection. Automatic, pre-programmed or operator driven real-time control of model position, attitude and motion has already been achieved with an ease which analogue systems could never approach.

5) Data acquisition. Certain system data (model position, attitude and motion) is inherently available within the controller. Particularly when dynamic testing or rapid sequencing of model position or attitude is envisaged, the high level of synchronization required between the data acquisition and control functions recommends the complete integration of those functions. Further, it is anticipated that advanced algorithms would demand a tightly coupled flow of information between the controller and data processing functions (reference 5).

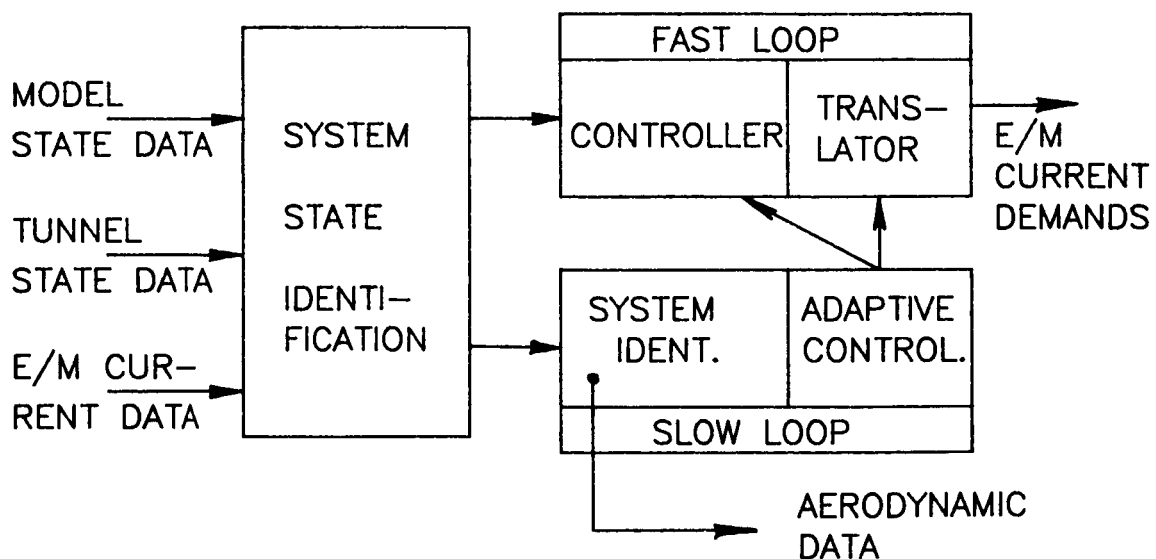


Fig.1.2 Conceptual block diagram of advanced controller

Development of a comprehensive suite of control software was initially undertaken by Bouchalis and Fortescue (reference 3), who achieved full 6 component control of winged models. Their software was later entirely re-written in a more modular form by this author. The software detailed in this report is essentially a second re-write, being modified to eliminate certain shortcomings of previous versions and to incorporate some more sophisticated features. A general comparison of the three suites is given in the following Table:

I (Bouchalis, Fortescue)	II	III
5/6 degree of freedom	5/6 degree of freedom	5/6 deg. of freedom
Traditional algorithms	Traditional alg.	Traditional alg.
Limited attitude, motion scheduling	Comprehensive att., motion scheduling	Comprehensive att., motion scheduling
No operator display	No operator display	Some display
No real-time data acquisition	Some real-time data acquisition	Some real-time data acquisition

All software so far written has been implemented using dedicated minicomputers, though on two slightly different configurations of same :

I,II (Southampton University)	III (NASA/AEDC 13")
PDP 11/34A CPU (see later note)	PDP 11/23-PLUS CPU
Extended Instruction Set	Extended Instruction Set
FP-11 Floating point processor	FPF-11 Floating point processor
28k words memory utilized	28k words memory utilized
KW-11 programmable clock	KW-11 programmable clocks
Custom built A/D and D/A subystem accessed via single DR-11 digital I/O	AAV-11 and ADV-11 A/D and D/A converters
RT-11 V4.0 operating system	DRV-11 digital I/O
MACRO V4.0 and FORTRAN V2.1 application software	RT-11 V5.0 operating system
	MACRO V5.0 and FORTRAN V2.6 application software

Due to the extensive commonality of the PDP-11 family, both in hardware and software, conversion of software from one configuration to another is relatively straightforward.

The current software is structured in a highly modular form in order to simplify the task of incorporating any amendments necessary to accommodate different test models, experimental set-ups or run-time features. Wherever possible, repetitious or common coding is accessed via simple calls to Subroutines or MACRO library modules.

Considerable efforts have been made to streamline program execution since CPU speed is the major limiting factor with presently available computer hardware. At the loop repetition

rates chosen at Southampton (typically 400 cycles per second), the hardware had been pressed close to its limit. Occasionally, with special features such as high angle-of-attack operation, computations extended beyond the available 1/400 secs, forcing reductions in loop rate. The NASA LaRC 13" MSBS can operate at lower loop repetition rates (typically 256 cycles per second) due to its larger size, hence lower system natural frequencies, but still has little spare CPU capacity since the PDP 11/23 is marginally slower than the 11/34. Certain desirable real-time features, particularly trapping of out-of-range or operator error conditions, are therefore not presently practical.

Note: Southampton University's PDP 11/34A CPU has now been upgraded to a PDP 11/84, other hardware remaining essentially unchanged. This represents a substantial increase in execution speed.

2. CONTROLLER CONCEPTS

Discussion in the remainder of this report will concentrate specifically on the controller hardware and software for the NASA LaRC 13" MSBS, though many details are common to the Southampton University system.

2.1 CONTROL ALGORITHMS

The block diagram of the controller, illustrated in Fig.2.1., corresponds to normal MSBS practice although various alternative orderings of the blocks are possible. The ordering shown is presently found to be the most convenient for digital operation.

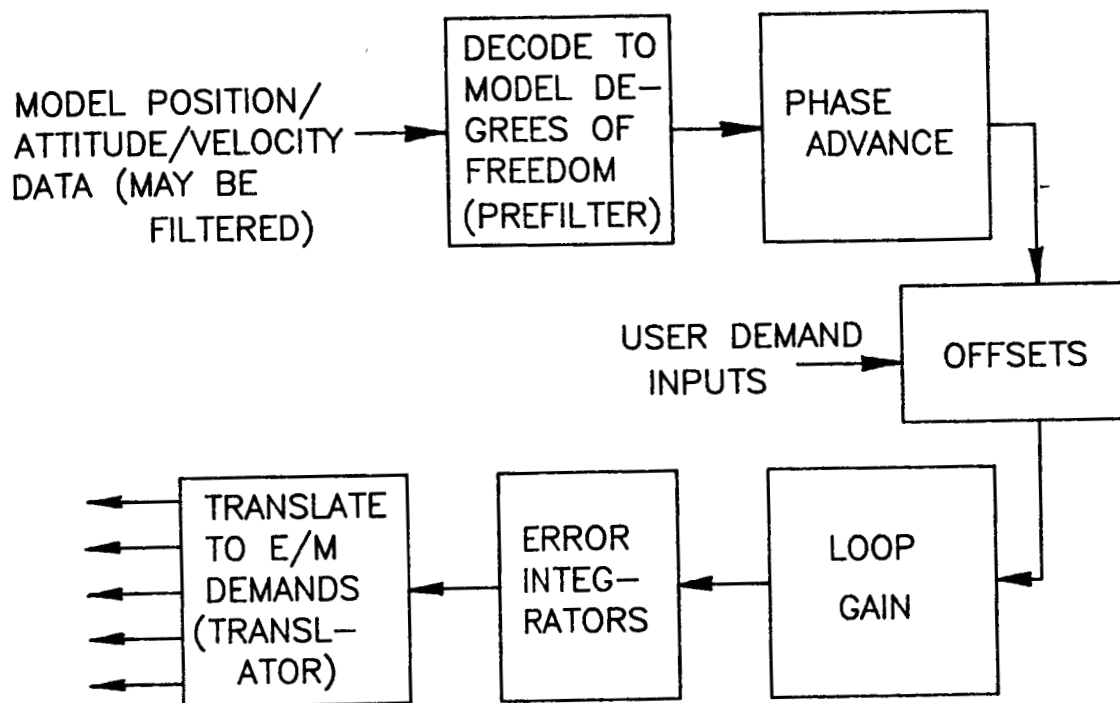


Fig.2.1 Functional block diagram of controller

The pre-filter and translator serve mainly to permit the control algorithms to operate in model degrees of freedom, this being thought to be the best approach. The levels of model suspension stiffness and stability that can be achieved are principally affected by the parameters and performance of the phase advance block. Dual series phase advance (lead-lag) stages have been widely used in this application (see reference 1) and are retained in the digital controller pending provision of any superior alternatives. The digital implementation of these stages is detailed in the following Section, although, due to the high stage gain at high frequencies, these algorithms are viewed as being rather unsuited to digital implementation. Nevertheless,

the qualitative performance, with regard to suspension stiffness and stability, of the two digitized controllers operated to date is virtually indistinguishable from the best that their analogue counterparts could achieve.

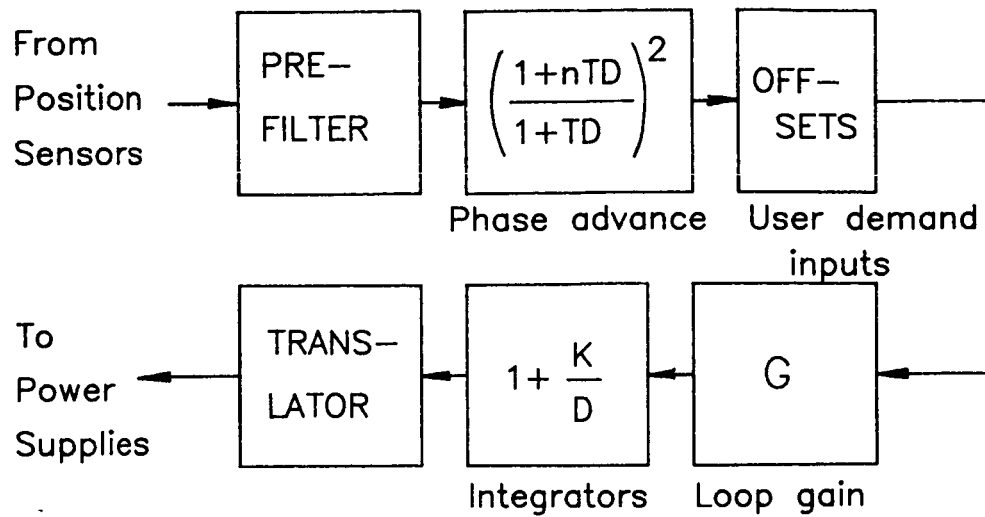


Fig.2.2 Idealized controller transfer functions

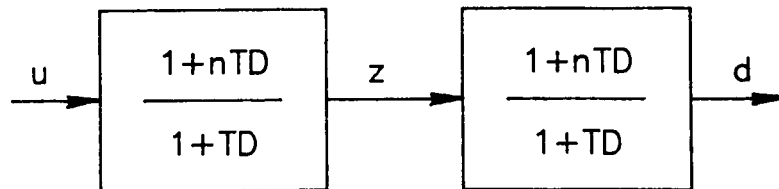
The controller transfer functions are shown in Fig.2.2. and may be implemented digitally as follows :

Prefilter

Algebraic combination of available position sensor data to provide measures of model position and attitude in model axes.

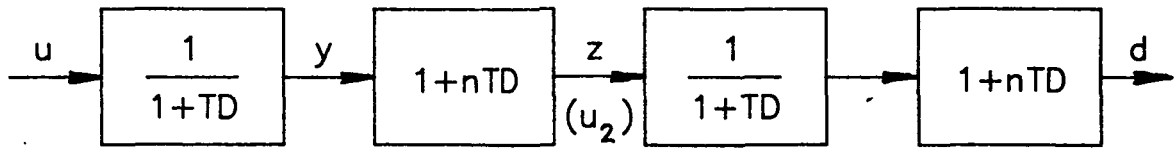
Phase advance

Configured in a dual series arrangement in the classical fashion :



- where n,T presently have the same value in each stage.

This is implemented digitally as follows :



Approximating for a discrete-time sampled-data system with sample interval Δt , we have :

$$T \frac{\Delta y}{\Delta t} = u - y \quad \text{with} \quad \Delta y = y(k) - y(k-1)$$

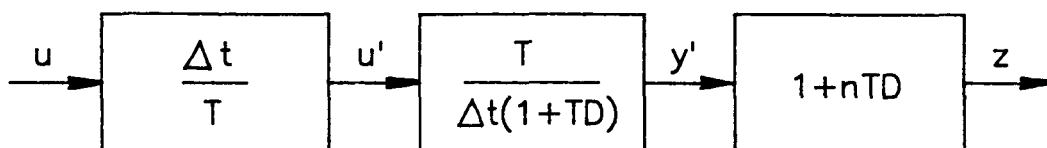
Thus :

$$y(k) \simeq \frac{\Delta t}{T + \Delta t} u(k) + \frac{T}{T + \Delta t} y(k-1)$$

This differs from the formulation given in reference 3 but is felt to be more appropriate at low sampling rates ($\Delta t \simeq T$).
Now following reference 3 :

$$z = y + nT \frac{\Delta y}{\Delta t} \quad \text{Giving :} \quad z(k) = y(k) + \frac{nT}{\Delta t} (y(k) - y(k-1))$$

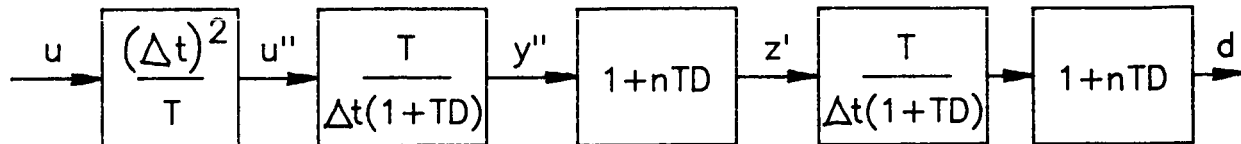
The above procedures are easily repeated by using $z(k)$ as the input ($u'(k)$) to the second controller block. It is seen that storage of one intermediate value ($y(k-1)$) is required for each block of the controller. It happens that slight savings in execution time can be made if the transfer function blocks are rearranged and a modified form of the first intermediate value stored :



$$u'(k) = \frac{\Delta t}{T} u(k)$$

$$y(k) = \frac{T}{T+\Delta t} (u'(k)+y(k-1))$$

The leading multiplier terms of the two blocks are now consolidated into a single operation :

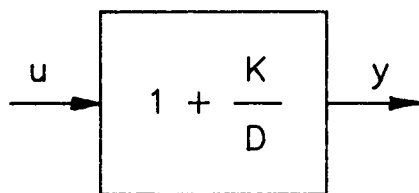


Additional gain terms can be agglomerated into the initial multiplier if desired.

Offsets and gains

Straightforward algebraic adjustments.

Integrator



$$y(k) \approx u(k) + K \sum_{n=-\infty}^k u(n) \Delta t$$

If the value of $K\Delta t$ is relatively small (typically $2 \cdot 10^{-12}$) it may be chosen somewhat arbitrarily, with no significant effect on controller stability. This has been the usual case.

2.2 SOFTWARE BLOCK STRUCTURE

A relatively complex structure for the executable program has evolved which seems to provide an appropriate framework for the required functions. The program is entered and exited through FORTRAN for convenience. In order to preserve sufficient memory capacity for storage of real-time data, the executable program uses overlays (handled by RT-11) so that surplus code areas can be written over by the real-time portion of the controller, as shown in Fig.2.3. This real-time portion is structured in four

layers, as shown in Fig.2.4.

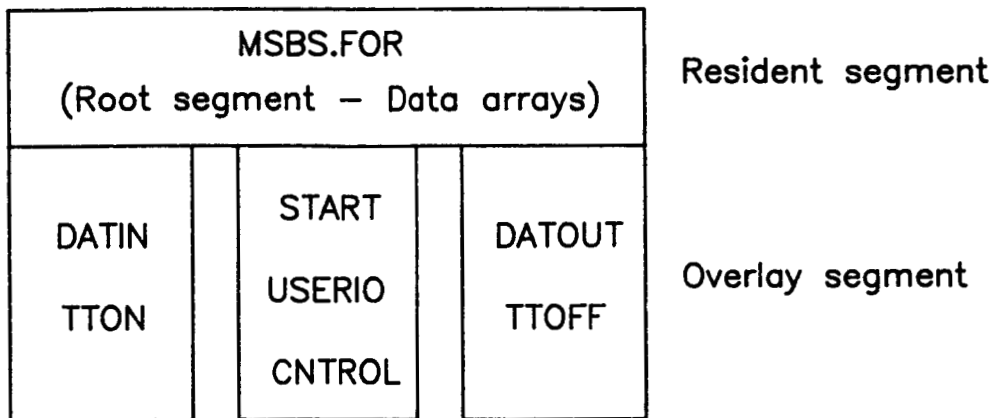


Fig.2.3 Control software overlay structure

"MSBS" performs the necessary system-level software initialization (automatically). Storage areas for oscillator, preprogrammed command, real-time (keyboard) command and acquired data arrays are reserved. All other program functions are handled by Subroutines which are called in sequence from MSBS.

"DATIN" reads preprogrammed routine data from a disc datafile and fills the oscillator array with one cycle of a sine wave.

"TTON" performs some system hardware initialization and sets up a screen display on the VT-102 console terminal.

"START" accesses data across the FORTRAN "CALL" interface and sets up certain hardware features, notably the programmable clock rate and interrupt status. Some hardware features are reset prior to exiting the real-time controller, including reset of all D/A channels to safe (zero current) values. These reset functions are also performed following certain run-time error conditions, prior to returning to the system monitor.

"USERIO" constitutes the only run-time interface between the console CRT/keyboard and the controller software. User commands are decoded, with some commanded functions performed directly and others routed via the user command array "CHARS".

This module is interrupted (by the programmable clock) whenever the module "CNTROL" is required to run. Execution of "USERIO" resumes after completion of each control loop, but only until interrupted again. Interrupts repeat at 256 cycles per second (presently), thus a rather small percentage of processor time is spent executing this module.

"CNTROL" preserves the system state following the hardware interrupt, permitting return to the previously executing code in "USERIO". All real-time controller, data acquisition and related I/O functions are performed with the previous system state being

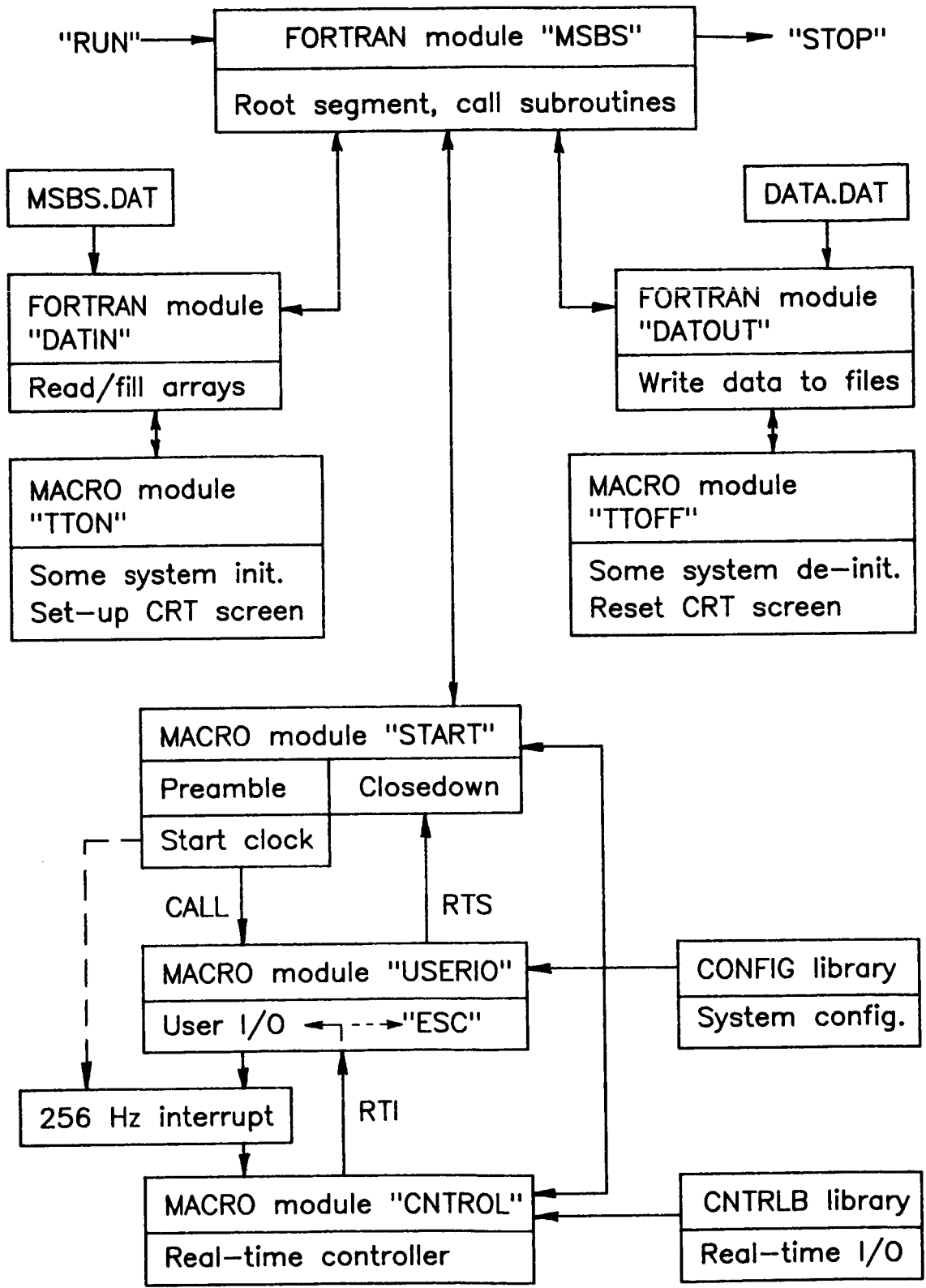


Fig.2.4 Control software block structure

restored prior to return to "USERIO".

"TTOFF" resets the VT-100 screen display.

"DATOUT" writes real-time data to a disc datafile if required.

Two library modules are employed to simplify program coding. "CONFIG" provides system hardware configuration information and the fixed global assignments for elements of the command array CHARS. "CNTRLB" provides MACRO's for all real-time I/O functions and for many controller functions repeated for each model degree of freedom.

2.3 DATA ARRAY FORMAT

The array CHARS is the medium by which run-time commands are transmitted to the controller and real-time data made available for inspection. The format of the array, summarized below, is crucial to the program structure and coding. Array addressing is based on use of ASCII numerical equivalents, with each array element effectively labelled with an ASCII character. It should be noted that the array is a subset of the ASCII code set.

	Element offset	ASCII char.	Element offset	ASCII char.	Element offset	ASCII char.
CHARS	0	SP	32	@	64	'
	1	!	33	A	65	a
	2	"	34	B	66	b
	3	#	35	C	67	c
	4	\$	36	D	68	d
	5	%	37	E	69	e
	6	&	38	F	70	f
	7	'	39	G	71	g
	8	(40	H	72	h
	9)	41	I	73	i
	10	*	42	J	74	j
	11	+	43	K	75	k
	12	,	44	L	76	l
	13	-	45	M	77	m
	14	.	46	N	78	n
	15	/	47	O	79	o
	16	0	48	P	80	p
	17	1	49	Q	81	q
	18	2	50	R	82	r
	19	3	51	S	83	s
	20	4	52	T	84	t
	21	5	53	U	85	u
	22	6	54	V	86	v
	23	7	55	W	87	w
24	8	56	X	88	x	

25	9	57	Y	89	Y
26	:	58	Z	90	Z
27	;	59	[91	{
28	<	60	\	92	!
29	=	61]	93	}
30	>	62	^	94	~
31	?	63	_	95	DEL

Element offsets are $(ASCII_8 - 40_8)$ or $(ASCII_{10} - 32_{10})$

Since CHARS is a word array, the byte offset is twice the element offset. It is often convenient, therefore, to code references to specific array elements in the following form :

$$CHARS + \langle 2 * 110 \rangle_8 = CHARS + \langle 2 * 72 \rangle_{10} = \text{Address element 72, ASCII "h"}$$

Each element of CHARS corresponds to a command value, real-time data value, program control switch or similar. Full details of the assigned functions of each element are given in Section 4, but as an example, the contents of element 33 (ASCII A) are interpreted as the loop gain of the axial degree of freedom. The contents of all elements may be inspected at run-time from the keyboard, but only elements 32 and above may be modified by keyboard commands. Any element may be modified by a preprogrammed routine, dealt with later in this Section.

The OSCILL array is loaded at run-time with a single cycle of a sine wave, amplitude 16,383 units, calculated at 1/1024th of a cycle increments. With the program loop rate as 256 Hz, the minimum usable frequency is 0.25Hz, with frequencies of any multiple of 0.25 Hz being available.

	Element	Contents	
OSCILL	1	16383*Sin (0)	(angles in degrees)
	2	16383*Sin (1*360/1024)	
	3	16383*Sin (2*360/1024)	
	4	etc.	
	1022		
	1023	16383*Sin (1022*360/1024)	
	1024	16383*Sin (1023*360/1024)	

A master counter (ASCII "~", CHARS element offset 94_{10}) is set up to tick once per program cycle (256 ticks/second), resetting to 0 on every 1024th tick, and provides a reference phase and frequency of oscillator. The run-time variable frequency and phase feature is implemented by the following addressing procedure :

OSCILL address = [(Master counter* Frequency) + Phase]
 (Truncated to first 10 bits)

The ICOMM array contains preprogrammed position, attitude and motion command data. Four independent routines, each of up to 80 individual commands, may be utilized, with options for fewer routines of greater length, up to the total command limit of 320. The format of the array is as follows :

	1st routine	2nd routine	3rd	4th
Element 1	{ Duration	240 { Duration	480 {	720 {
	{ -----	{ -----	{ ----	{ ----
(1st command)	2 { Value	241 { Value	{	{
	{ -----	{ -----	{ ----	{ ----
	3 { Code	242 { Code	{	
	-----	-----		
(2nd command)	4 { Duration			
	{ -----			
	5 { etc.			

(3n+1)	{ Terminator			
	{ -----			
	239 {			

A terminator (Duration=0) is usually present, otherwise execution proceeds to the next available instruction, such as the 1st command of the second routine.

The 1st routine may thus contain the 2nd, and so on.

There is no restriction on access to the CHARS array from a preprogrammed routine (contrasting with restricted access from the keyboard) but certain keyboard functions do not operate :

ESC The controller can only be aborted from the keyboard
 ? Real-time data cannot be displayed automatically
 DEL Not required

2.4 I/O Filtering

If analogue position sensors, or analogue data-links from the position sensors, are used, then some form of analogue filtering is necessary at the controller A/D inputs. This corresponds to classical anti-aliasing filtering, though it is not yet clear whether the anti-aliasing function per se is particularly necessary for MSBS controller operation. Typical controller sampling frequencies have been at least a factor of 10 above any natural frequency of the suspended model and the systems overall response falls very rapidly with increasing frequency. Rather, the filtering acts to limit high frequency noise at the controller inputs, since the conventional control algorithms exhibit high stage gain at high frequencies. The NASA LaRC 13" MSBS now has digital datalinks from inherently

digital Self-Scanning Photodiode Array position sensors to the controller and appears not to require any form of input filtering. The frequency content of the position sensor information is naturally limited, of course, by the sampling processes inherent in the position sensor operation.

Modest analogue filtering of the output demands to the electromagnet power supplies (via D/A convertors) appears to be advisable to avoid problems with conventional thyristor power amplifiers, notably the "beating" of the controller repetition frequency with the fundamental or harmonics of the thyristor firing frequency. This can be achieved by roll-off of the frequency response of the power supply preamplifiers.

3. SYSTEM CONFIGURATION

A schematic diagram of the 13" MSBS laboratory is shown as Fig.3.1, with a block diagram of the MSBS hardware as Fig.3.2. Some clarification is necessary.

The peripheral processor (KXT-11C) is installed but not presently utilized. The intended function is to serve as an intelligent interface to wind tunnel instrumentation in order to make available real-time tunnel data, such as Mach number.

Several different types of electromagnet power supply are in use, presently all on a temporary basis. Some power supplies have internal current feedback and their inputs can thus be regarded as current demand. The thyatron power supplies in use with the main electromagnets have no internal feedback and should thus be regarded as having a voltage demand input.

3.1. POSITION SENSORS

Model position data is acquired from five linear photodiode arrays, each of 1024 elements, illuminated by laser light sheets. The general principles of the system are shown in Fig.3.3. and Reference 7 gives more details of its operation. The output from the processing electronics is in the form of parallel 10-bit words, the value of each word corresponding to the number of photodiodes illuminated above some threshold level. Presently, the 1024th element is ignored. Synchronization of position sensor "scans" with the digital controller is important and is achieved in the following way. The (KWV-11C) clock signal which triggers the interrupt for initiation of the controller software loop also causes a "start scan" trigger to the position sensors. By placing the position sensor data input routines immediately after the interrupt (at the head of each controller loop), acceptably tight synchronization is achieved, with no possibility of erroneous data, for all usable position sensor scan times. The hardware arrangement is shown in Fig.3.4. and the system timing diagram as Fig.3.5. Port assignments for sensor data are shown as Fig.3.6.

3.2. ISOLATION AMPLIFIERS and ELECTROMAGNET CURRENT MONITORING

Electromagnet currents are monitored using conventional shunts, via a purpose-built isolation amplifier system. Each amplifier can be adjusted for gain, offset and first-order filter characteristics. Outputs from the current shunt amplifiers are monitored by 12-bit A/D converters (ADV-11C). The isolation amplifiers are also used as the interface between the PDP 11/23+ and the electromagnet power supplies. A complete separation of electrical grounds is thereby achieved. This is desirable for safety reasons and for the avoidance of earth-loop interference. A schematic diagram of the overall arrangement is given as Fig.3.7. and isolation amplifier channel assignments are shown as Fig.3.8.

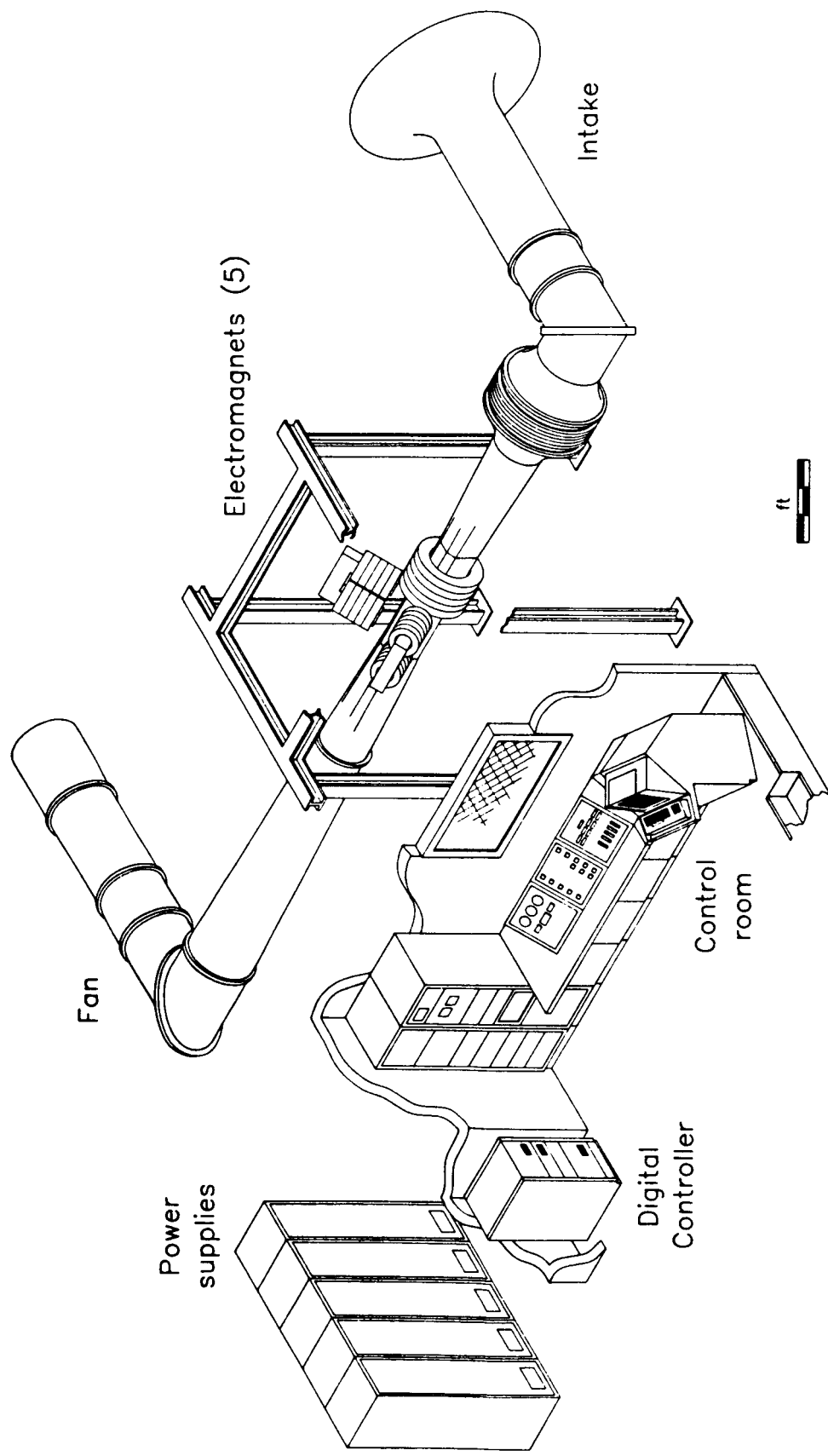


Fig.3.1.1. Schematic diagram of 13" MSBS laboratory

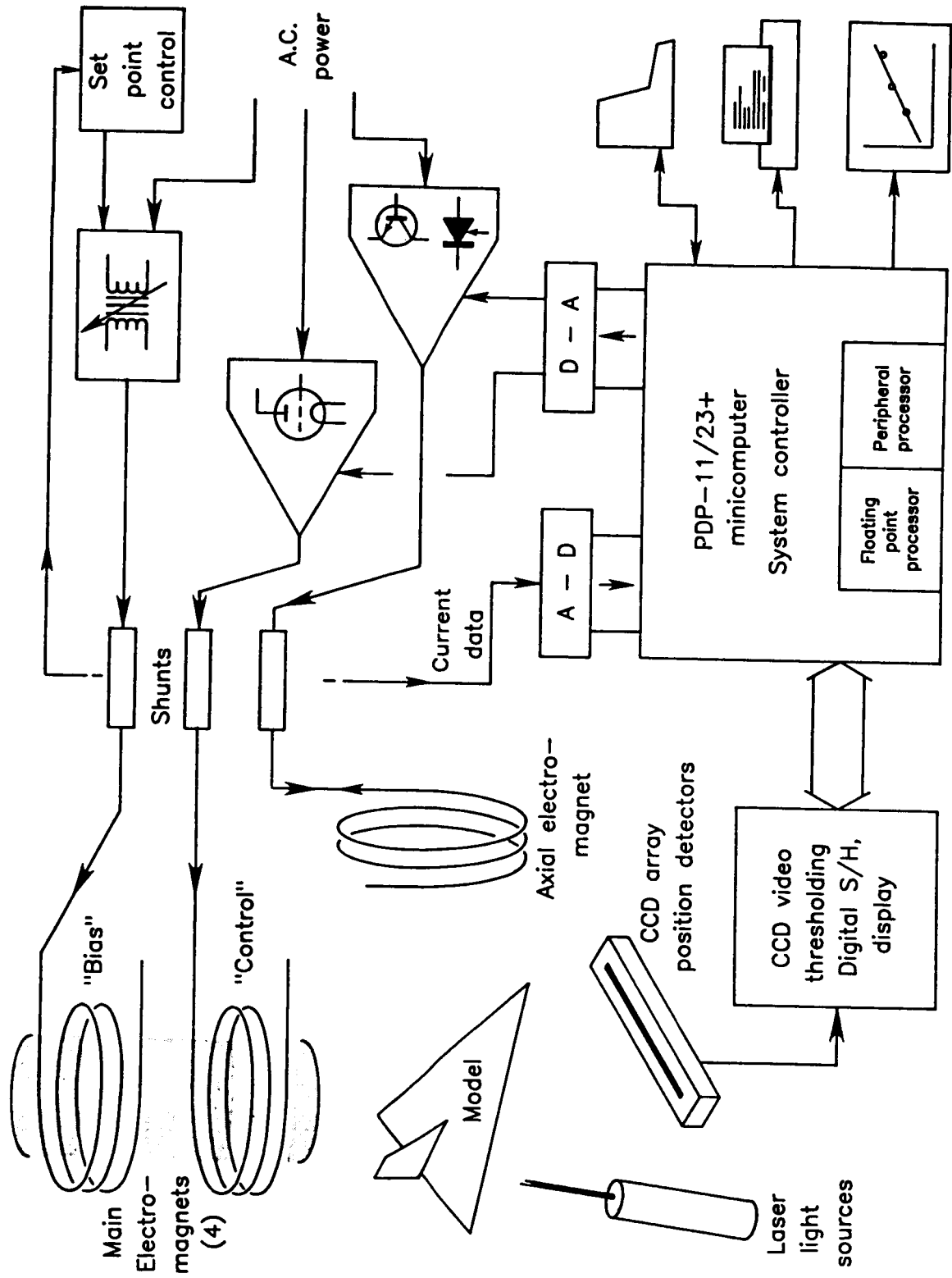


Fig.3.2. Block diagram of 13" MSBS hardware

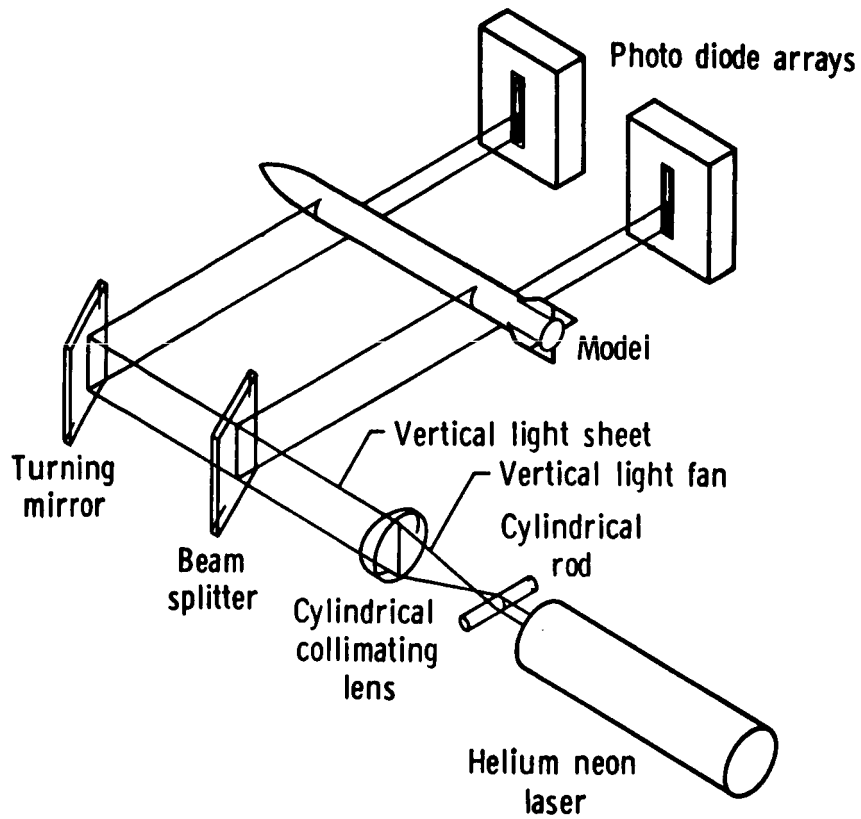


Fig.3.3. Schematic diagram of position sensor system (2 channels)

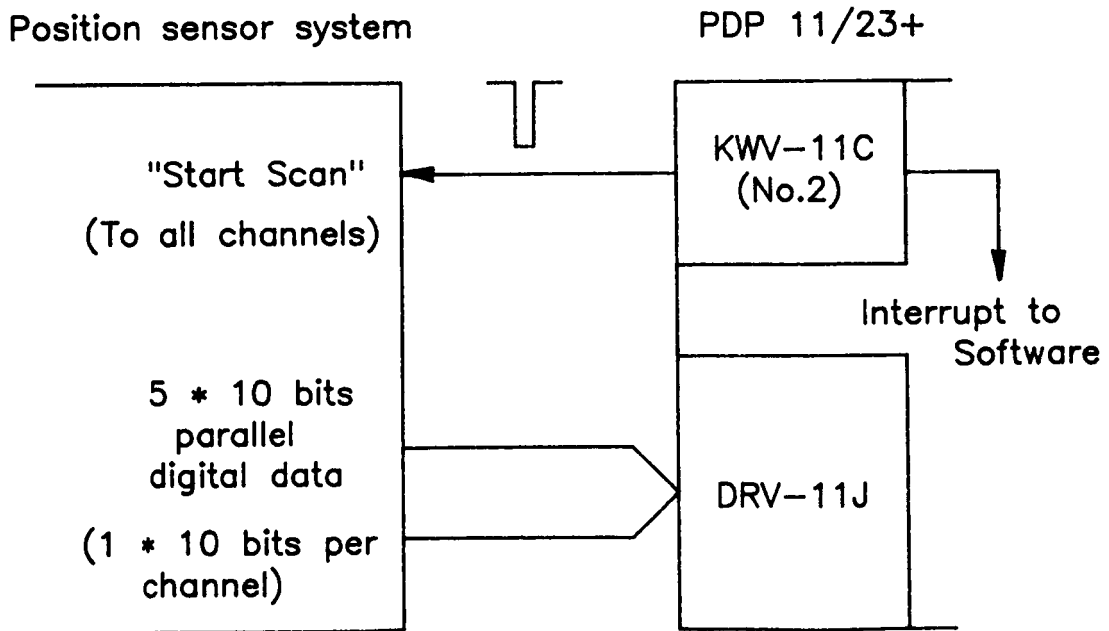


Fig.3.4. Position sensor interface schematic

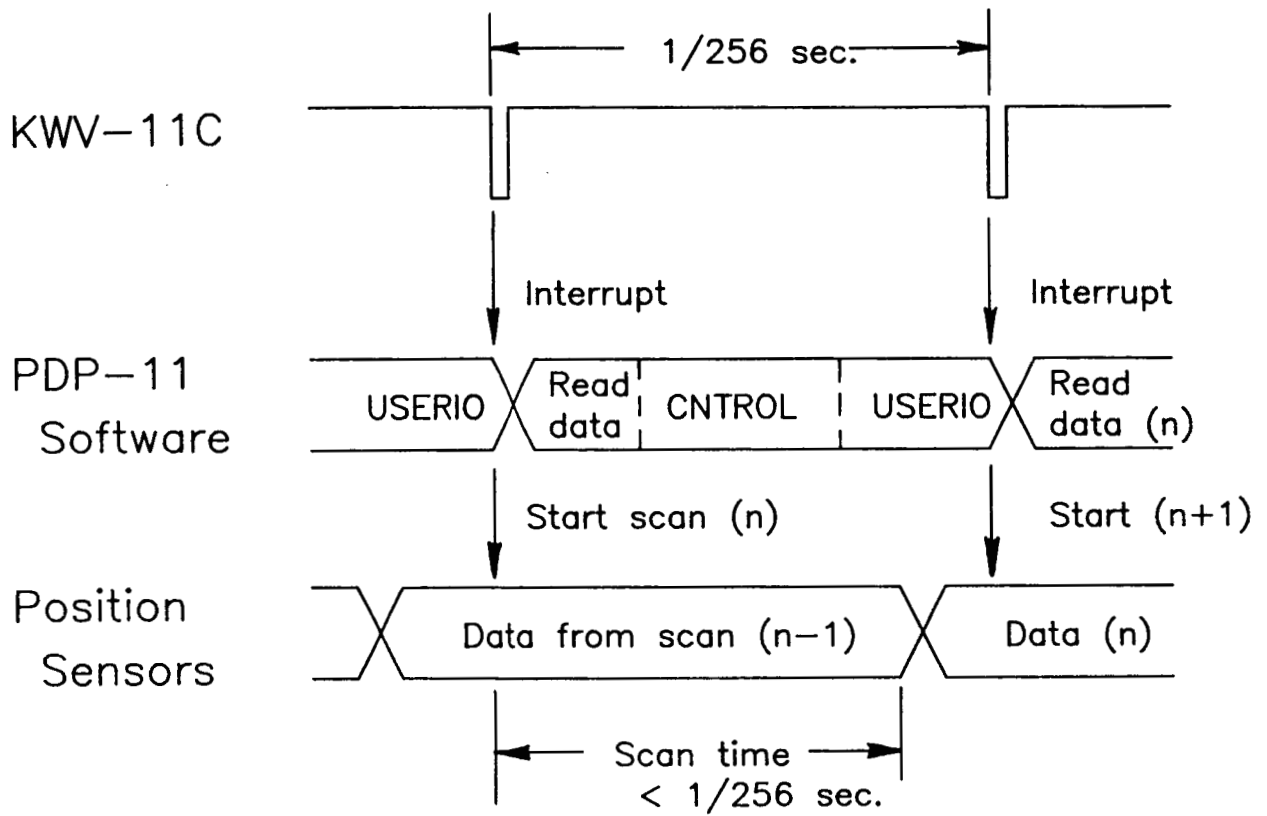


Fig.3.5. Position sensor software/hardware timing diagram

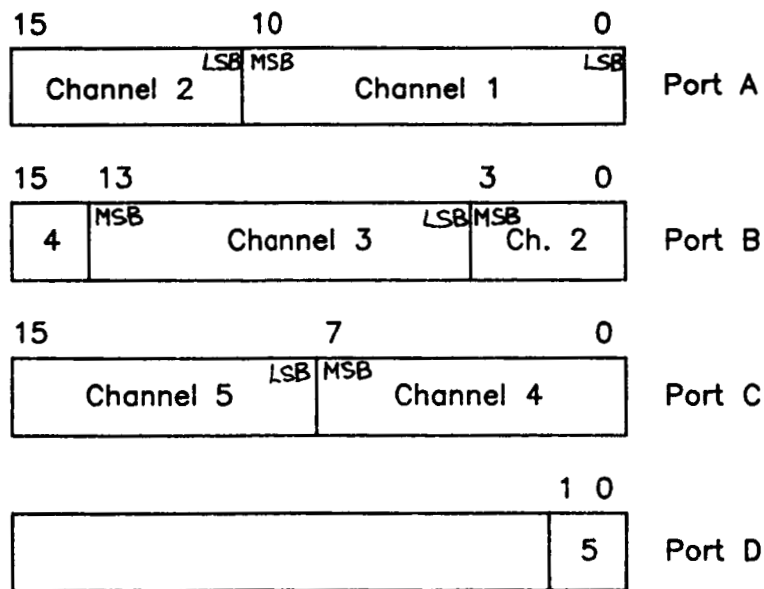


Fig.3.6. DRV-11J port assignments for position sensor data

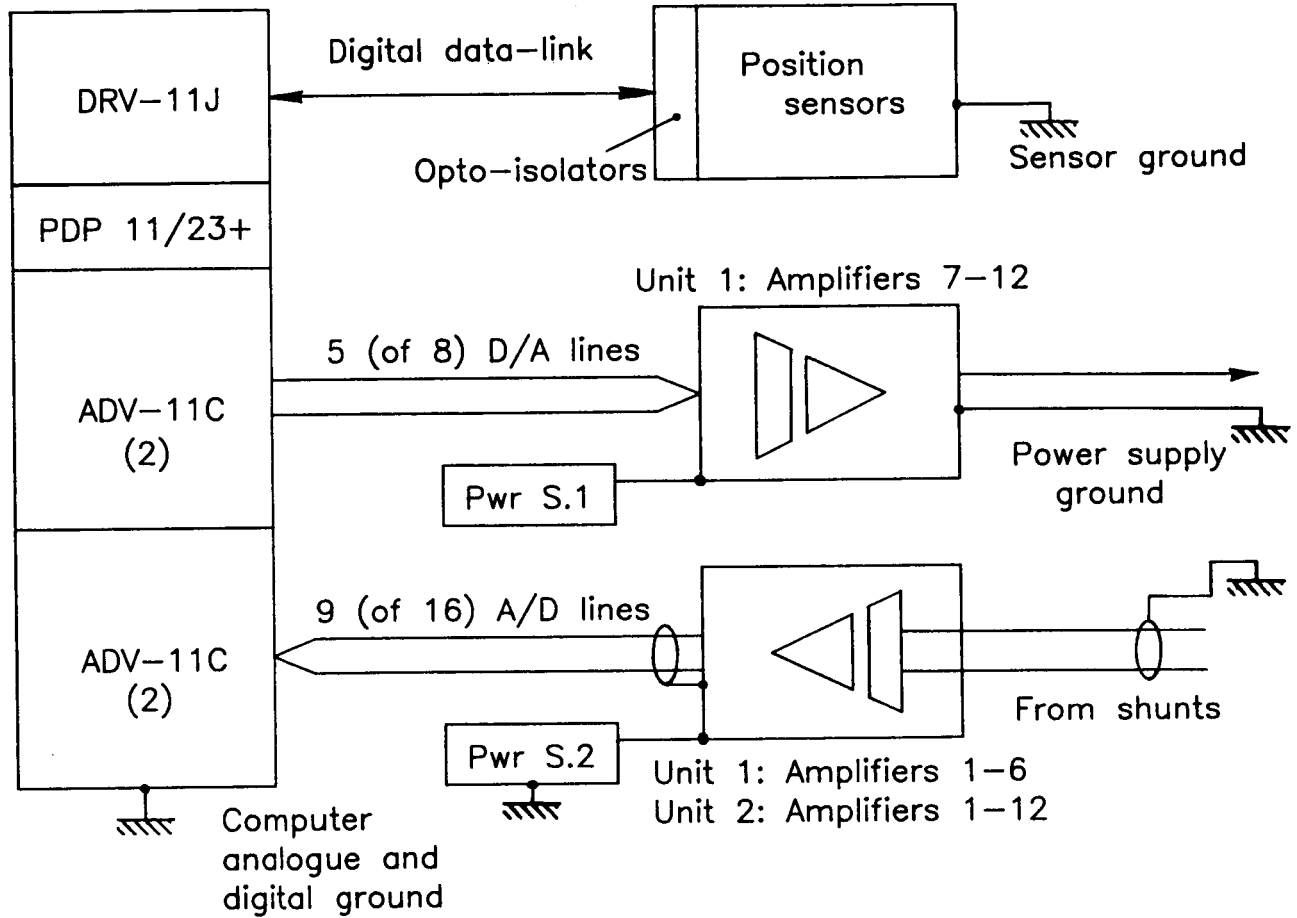


Fig.3.7. PDP 11/23, position sensor and isolation amplifier system

Unit 1

1 A/D 2.3	3 A/D 2.5	5 A/D 2.7	7 D/A 1 Cntrl 1	9 D/A 3 Cntrl 3	11 D/A 5 Drag	Amp no. PDP I/O Function
2 A/D 2.4	4 A/D 2.6	6 A/D 2.8	8 D/A 2 Cntrl 2	10 D/A 4 Cntrl 4	12 D/A 6	

Unit 2

1 A/D 1.1 Drag	3 A/D 1.2 Bias 1	5 A/D 1.4 Bias 3	7 A/D 1.6 Cntrl 1	9 A/D 1.8 Cntrl 3	A/D 2.3 *
2 A/D 2.2	4 A/D 1.3 Bias 2	6 A/D 1.5 Bias 4	8 A/D 1.7 Cntrl 2	10 A/D 2.1 Cntrl 4	A/D 2.4 *

A/D channel numbers are Board.Channel
D/A channel numbers are Channels, across 3 boards

All A/D amplifiers fed from Power Supply 2
All D/A amplifiers fed from Power Supply 1

Fig.3.8. Isolation amplifier channel assignments

3.3. PDP 11/23+

Some relevant details of the hardware and software configuration are given below:

CPU - PDP 11/23 PLUS with FPF11 floating point processor

Operating system - RT11 V5.0
Languages - FORTRAN V2.6, MACRO V5.0

User I/O devices - VT102 CRT, LA100 printer, HP7475 plotter

Real-time I/O - 16/32 A/D channels (differential/single ended)
12 D/A channels
64 digital I/O lines

Clocks - 2 programmable

Serial I/O - 6 lines (console, printer, plotter, 3 spare)

Memory - 256kB (only 56kB normally available for real-time software)

I/O module addresses (relevant to digital controller) -

Module	Function	Address & Vector (octal)	
	Console CRT serial line	177560	60
KXT-11C	Peripheral processor	160100	
KWV-11C	Programmable clock (No.1)	170400	440
KWV-11C	(No.2)	170420	450
ADV-11C	A/D convertors (Board 1)	170430	400
ADV-11C	(Board 2)	170440	420
AAV-11C	D/A convertors (Board 1)	170450	
AAV-11C	(Board 2)	170460	
AAV-11C	(Board 3)	170470	
DRV-11J	64-line Digital I/O	164160	Prog.

4. CONTROLLER USERS GUIDE

4.1 HARDWARE OPERATION

4.1.1 STARTUP

POWER UP PDP 11/23 AND TERMINALS

Switch on CRT	(Switch at lower left side)
Switch on printer	(Switch at lower left rear)
Set printer ON LINE	(Keyboard key)

Check HALT switch up on PDP CPU	(Automatic system bootstrap)
Raise PWR switch on PDP CPU	(PWR and RUN lights come on)
	(Bootstrap message on CRT)

LOAD disc drives (Drives 0 and 1)
Wait for READY lights on both

LOAD SYSTEM SOFTWARE

Respond to CRT bootstrap message :

Type : DL<Ret>
(System software loads)
(Startup message appears on CRT)
(Enter date, eg. 5,24,86)

RT-11 is now running, as indicated by the DOT (".") prompt.

(Power up MSBS INTERFACES and POSITION SENSORS if required)

4.1.2 CLOSEDOWN

Release disc LOAD switches	(READY lights extinguish)
Wait for LOAD lights to come on	(IMPORTANT)

Lower PWR switch on PDP CPU
Switch off CRT and printer

4.2 CONTROLLER SOFTWARE OPERATION

4.2.1 ACTIVATE CONTROL PROGRAM

When all relevant hardware is powered-up, the control software can be started, with the MSBS in almost any status (model in/out etc.).

Type: RUN MSBS<Ret>

Response: ACCESSING DATA FILES (Controller is reading preprogrammed routines, zeroizing data stores and loading the oscillator array)

Run-Time CRT display appears, showing title message, command reminders, etc.

4.2.2 RUN-TIME SCREEN DISPLAY

The VT-102 screen display initializes in broadly the following format:

```

      NASA LaRC 13 inch MAGNETIC SUSPENSION and BALANCE SYSTEM
-----
|Axial   | x | a i r | A | (   )   Integrators=(   )
|Lateral | y | b j s | B | (   )   Drag Int.=(   )
|Vertical| z | c k t | C | (   )   Outputs=(   )
|Pitch   | m | e p v | E | (   )
|Yaw     | n | f q w | F | (   )   Auto routine
-----
      <Screen area A>           <area B>           <area C>

```

<Cursor initializes here>

Screen area A provides "memory joggers" for some of the most commonly used keyboard command codes. The first column (x-n) shows codes for the usual steady position or attitude commands, the next three columns show codes for oscillator amplitudes, frequencies and phases respectively, with the last column showing loop gain codes.

Screen areas B and C are updated only when <Ret> is entered at the keyboard, with the latest available data. Area B shows the measured model position and attitude in each degree of freedom, in units of raw counts from the position sensors. This data serves mainly as a check when powering up and launching. Area C shows the state of various program switches, either ON or OFF and the number of the preprogrammed routine presently loaded (see Section 2.3). The integrator switches are forced OFF and the integrator accumulators are cleared (set to zero) when the output switch is OFF. Otherwise the accumulator values are preserved.

The lower portion of the display scrolls conventionally, echoing commands and data entered from the keyboard, or displaying real-time data (via the ? command).

There is typically no external evidence that the controller is executing correctly, though a spare D/A channel can be used to output a square wave of frequency equal to the loop rate for test purposes. Detection of this signal is a reliable indication that the software is at least looping correctly (hardware interrupt driven). Run-time checks would, however, only normally be necessary following major software amendments.

4.2.3 RUN-TIME KEYBOARD COMMANDS

Communication between operator and system (via CRT/keyboard) is handled exclusively by the real-time software, therefore great care must be exercised by the operator/programmer.

Since lower case command codes correspond to the usual run-time commands, the CAPS LOCK key of the CRT should normally be released (up).

```
*****
*****      TYPING ERRORS ARE LIKELY TO CAUSE      *****
*****      LOSS OF CONTROL OF A SUSPENDED MODEL    *****
*****
```

Certain controller functions are accessible in real-time from the keyboard. In addition, some real-time data may be examined via the CRT, though not amended by keyboard input. All functions are commanded in a similar format :

(Number) (-) (CODE LETTER) <Ret>

Parameters in curved brackets are optional. Note carefully that the (optional) minus sign is trailing, the same format being used for all screen displays. The function of the CODE letters is explained in the following table :

CODE letter	ASCII (octal)	Function and comments
=====		
Special characters		
ESC	033	Abort program
-	055	Minus sign (must trail value)
?	077	Query data
DEL	177	Delete present input line (line is cleared)
Ret	N/A	Enter command (Update screen)
=====		
Examine data. Read only (from keyboard), preceded by ?		
SP	040	AL position sensor data
!	041	AX
"	042	AU
#	043	FU
\$	044	FL
%	045	Unassigned. Reserved for position sensor data
&	046	X position data
'	047	Y
(050	Z
)	051	L orientation data
*	052	M
+	053	N
,	054	
-	055	<<Minus sign>>

	056	
/	057	Remaining number of data sweeps
0	060	Reserved for electromagnet
1	061	current data
2	062	"
3	063	"
4	064	"
5	065	"
6	066	"
7	067	"
8	070	"
9	071	"
:	072	Ramp vector
;	073	Ramp increment
<	074	
=	075	
>	076	
?	077	<<Query contents of array location>>

=====
Data input. Read/write from keyboard

Upper case codes

@	100	
A	101	X (axial) loop gain
B	102	Y (lateral)
C	103	Z (vertical)
D	104	L (roll)
E	105	M (pitch)
F	106	N (yaw)
G	107	Load preprogrammed routine (0-3)
H	110	
I	111	
J	112	
K	113	
L	114	L Auxiliary displacement
M	115	M
N	116	N
O	117	
P	120	
Q	121	
R	122	
S	123	
T	124	
U	125	
V	126	
W	127	
X	130	X Auxiliary displacement
Y	131	Y
Z	132	Z
[133	Data acquisition flag (number of sweeps)
\	134	
]	135	Output flag (0=Off, otherwise On)
^	136	
_	137	

Lower case codes

'	140	
a	141	X oscillator amplitude (Unspecified
b	142	Y oscillator amplitude units)
c	143	Z oscillator amplitude
d	144	L oscillator amplitude
e	145	M oscillator amplitude
f	146	N oscillator amplitude
g	147	Automatic routine flag (number of repeats)
h	150	Integrator switch (0=off, otherwise on)
i	151	X oscillator frequency (Units of
j	152	Y oscillator frequency 0.25 Hz)
k	153	Z oscillator frequency
l	154	L orientation (Unspecified units)
m	155	M orientation
n	156	N orientation
o	157	L oscillator frequency (Units of
p	160	M oscillator frequency 0.25 Hz)
q	161	N oscillator frequency
r	162	X phase (Units of 360/1024 degrees
s	163	Y phase in LEAD relative to ref.)
t	164	Z phase
u	165	L phase
v	166	M phase
w	167	N phase
x	170	X position (Unspecified units)
y	171	Y position
z	172	Z position
{	173	
	174	
}	175	
~	176	Master oscillator counter (ticks of 1/256th
		second, reset to 0 on 1024th tick)
DEL	177	<<Cancel input line>> Also spurious data sink

=====

Quick reference table for normal runtime (lower case) commands

	X	Y	Z	L	M	N
Position	x	y	z	l	m	n
Osc. amp.	a	b	c	d	e	f
Freq.	i	j	k	o	p	q
Phase	r	s	t	u	v	w

] = Outputs on/off	h = Main integrators on/off
H = Drag integrator on/off	[= Data acquisition (no. sweeps)
g = Auto routine (no.repeats)	DEL = delete input (line clears)
ESC = abort program	

Other functions

	X	Y	Z	L	M	N
Loop gain	A	B	C	D	E	F
Aux. disp.	X	Y	Z	L	M	N

All numerical keyboard I/O takes place in INTEGER DECIMAL format.

Examples

```
3000A <Ret> - sets axial (drag) gain to 3000
1000-m <Ret> - calls for small negative pitch
               displacement
20[ <Ret>    - takes burst of 20 data sweeps
h <Ret>      - turns off integrators

?(          - causes display of last Z (heave)
              model position sample
```

Note that ?, DEL or ESC do not require the usual <Ret> terminator

4.2.4 Launch and landing sequences

Launch

With control software activated and most MSBS systems powered-up (NOT "CONTROL" or "BIAS" or "DRAG" supplies) a sequence of operations to launch the model might be:

- 1) Activate BIAS supplies and set to desired current
- 2) Activate CONTROL currents and DRAG current. No significant currents should flow in any controlled coil.
- 3) Type 5000z <Ret> (commands system to suspend model very low, roughly corresponding to its position on the model launcher)
- 4) Type 1] <Ret> (turns outputs ON, small control currents begin to flow, model is now under partial control)
- 4) Type z <Ret> (commands model to lift to datum position. Offset settings in the program usually result in insufficient current to actually lift the model)
- 5) Type 1h <Ret> (turns position error integrators ON. Model will now rise positively to the desired position and orientation)
1H <Ret>

Landing

- 1) Type] <Ret> (turns output currents OFF, model drops onto grabbers, integrators turn OFF)
- 2) Turn off CONTROL and DRAG power supplies
- 3) Turn off BIAS currents
- 4) Abort control program ("ESC")

4.3 LIMITATIONS and RESTRICTIONS

Data acquisition

The limited direct memory addressing capability of PDP-11 computers (32kW (kWords)), together with the overhead of the I/O page (4kW), the vector table (320W) and RT-11 (varies between 2kW and 5KW depending on monitor type and configuration), leaves rather restricted memory available for real-time data. In fact the overlaid structure of the controller program (Section 2.2) is employed solely to free as much memory as possible. Nevertheless, only around 12000 words of real-time data storage can normally be made available, corresponding to 12000 data samples. With a number of samples being taken per program loop (typically at least 14 samples = 5 positions/attitudes + 9 currents), the maximum number of data "sweeps" is given by :

$$S = 12000/14 = 857 \text{ (approx)}$$

At 256 program cycles per second, this is in turn equivalent to less than 3.3 seconds of real-time data. Data can be taken in several bursts at any interval up to the maximum number of sweeps shown above. Data overflow is prevented by repetitively overwriting, when necessary, the last set of valid data locations. Otherwise the program would attempt to write to sequential memory locations beyond the data array, eventually corrupting other program code.

The short-term solution to the memory problem is seen as invoking RT-11 Memory Management instructions to access Extended Memory in real-time. On a PDP 11/23 up to around 2000kW of such memory may be installed. The coding necessary to implement access is certainly feasible, but by no means trivial.

Control Algorithms

Existing control algorithms are rather unsophisticated and are long overdue for replacement.

Prefilter and translator

As presently structured, these modules can implement a fixed linear relationship between input and output, eg.:

(Axial position)	(a b c d e)	(AL)	
(Lateral pos.)	(f g .)	(AX)	
(Vertical pos.)	= ()	(AU)	-(Prefilter)
(Pitch attitude)	(etc.)	(FU)	
(Yaw attitude)	()	(FL)	

and:

(E/M 1 demand)	(l m n o p)	(Vertical demand)	
(E/M 2 demand)	(q r .)	(Pitch demand)	
(E/M 3 demand)	= ()	(Lateral demand)	-(Translator)
(E/M 4 demand)	()	(Yaw demand)	
(E/M 5 demand)	()	(Axial demand)	

The 13" MSBS is inherently non-linear. For instance, the vertical magnetic force, below the point of saturation of the model's core, will vary as the square of the "vertical" current (a sum of currents in the four main E/Ms). If this peculiarity is to be accommodated, much more elaborate translator coding will be necessary. The present coding might be regarded as a piecewise linear approximation, valid for small perturbations about the selected datum position/orientation and force/moment level. The effects of variations in the force/moment level, such as between wind-on and wind-off test points, can be partially accommodated by re-trimming loop gains (from the operator's keyboard). This is, in fact, a technique that was used extensively by MIT with their 6" MSBS for somewhat similar reasons.

If suspension over a wide range of positions and/or attitudes is contemplated, then the inevitable variation of translator coefficients (particularly with attitude) due to the nature and behaviour of the magnetic field gradients will have to be taken into consideration. This issue is very complex (see Ref.6).

Similarly, the variation of prefilter coefficients with model position and/or attitude (particularly attitude) would need to be taken into consideration for large excursions from the datum to be possible. Analysis is relatively more straightforward here however, being dependant exclusively on system geometry.

APPENDIX A

COMPILATION, ASSEMBLY AND LINKING OF EXECUTABLE PROGRAM

A total of 44 files are involved in the creation of the executable program; 10 source code files, 12 command files, 2 library files, 8 object modules, 9 listing or map files, 2 run-time datafiles and finally the executable program file itself.

Table A.1 lists all the file names and functions. Table A.2 illustrates the complete procedure for creation of the executable program. Refer to the source code listings in Appendix B.

TABLE A.1 FILENAMES AND FUNCTIONS

FILENAME	FUNCTION and COMMENTS

Source code files :	

MSBS.FOR	FORTRAN root
DATIN.FOR	Data input/load Calls MSBS.DAT at run-time
DATOUT.FOR	Data output/store Creates DATA.DAT at run-time

TTON.MAC	Initialize (display) Requires CONFIG.MLB
TTOFF.MAC	De-initialize "
START.MAC	Startup module "
USERIO.MAC	User I/O module "
CNTR.LB.MAC	Controller module - and CNTRLB.MLB

CONFIG.MAC	Configuration library source
CNTRLB.MAC	Real-time I/O library source

Command files :	

MSBS.COM	Command file for MSBS.FOR compilation
DATIN.COM	-for compilation of DATIN.FOR
DATOUT.COM	-for compilation of DATOUT.FOR
TTON.COM	-for assembly of TTON.MAC
TTOFF.COM	-for assembly of TTOFF.MAC
START.COM	-for assembly of START.MAC
USERIO.COM	-for assembly of USERIO.MAC
CNTRL.COM	-for assembly of CNTRL.MAC
MSLINK.COM	-for linking of MSBS.SAV

MSDBG.COM	-expanded functions of CNTRL.COM for debugging

CNTRLB.COM	Command file to create CNTRLB.MLB library
CONFIG.COM	-to create CONFIG.MLB library

Numerous temporary or auxiliary files are created during the compilation, assembly and linking processes :	

CONFIG.MLB	MACRO library modules
CNTRLB.MLB	

```

-----
MSBS.OBJ      Object modules
DATIN.OBJ
DATOUT.OBJ
TTON.OBJ
TTOFF.OBJ
START.OBJ
USERIO.OBJ
CNTROL.OBJ
-----

```

```

-----
MSBS.LST      Compiler listing files
DATIN.LST
DATOUT.LST
TTON.LST      Assembler listing files
TTOFF.LST
START.LST
USERIO.LST
CNTROL.LST
MSBS.MAP      Linker map file
-----

```

2 run-time datafiles are required or created :

```

-----
MSBS.DAT      Preprogrammed model motions (required)
DATA.DAT      Acquired real-time data (created/overwritten)
-----

```

The final executable program file is :

```

-----
MSBS.SAV      Executable program, calls MSBS.DAT, creates
               DATA.DAT at run-time.
-----

```

TABLE A.2 MODULE COMPILATION, ASSEMBLY and LINKING

Command files are activated under RT-11 by entering:

```

@FILNAM <Ret>      -where .COM is the default extender

```

CREATE LIBRARIES

Command file	Input module	Output module
CONFIG (.COM)	CONFIG.MAC	CONFIG.MLB
CNTRLB (.COM)	CNTRLB.MAC	CNTRLB.MLB

ASSEMBLE MACRO MODULES

Command file	Input modules	Output modules
TTON (.COM)	TTON.MAC, CONFIG.MLB	TTON.OBJ, TTON.LST
TTOFF (.COM)	TTOFF.MAC, CONFIG.MLB	TTOFF.OBJ, TTOFF.LST

START (.COM)	START.MAC, CONFIG.MLB	START.OBJ, START.LST
USERIO (.COM)	USERIO.MAC, CONFIG.MLB	USERIO.OBJ, USERIO.LST
CNTROL (.COM)	CNTROL.MAC, CNTRLB.MLB, CONFIG.MLB	CNTROL.OBJ, CNTROL.LST

MSDBG (.COM)	CNTROL.MAC, CNTRLB.MAC, CONFIG.MLB	CNTROL.OBJ, CNTROL.LST
--------------	---------------------------------------	------------------------

(MSDBG includes full MACRO expansions)

COMPILE FORTRAN MODULES

Command file	Input modules	Output modules
MSBS (.COM)	MSBS.FOR	MSBS.OBJ, MSBS.LST
DATIN (.COM)	DATIN.FOR	DATIN.OBJ, DATIN.LST
DATOUT (.COM)	DATOUT.FOR	DATOUT.OBJ, DATOUT.LST

LINK OBJECT MODULES

Command file	Input modules	Output modules
MSLINK (.COM)	MSBS.OBJ, DATIN.OBJ, DATOUT.OBJ, TTON.OBJ, TTOFF.OBJ, START.OBJ, USERIO.OBJ, CNTROL.OBJ	MSBS.SAV, MSBS.MAP

Each command file (compilation, assembly or linking process) should be activated only when an input module to that file (process) is modified. For example, if CNTRLB.MAC is altered (edited), the procedure to generate a revised executable program would be :

@CNTRLB <Ret>	-generate revised library
@CNTROL <Ret>	-reassemble modules that utilize CNTRLB routines
@MSLINK <Ret>	-relink executable program

APPENDIX B

SOURCE CODE LISTINGS

MSBS.COM

FORT/LIST:MSBS/WARN MSBS

DATIN.COM

FORT/LIST:DATIN/WARN DATIN

DATOUT.COM

FORT/LIST:DATOUT/WARN DATOUT

TTON.COM

R MACRO
TTON, TTON=CONFIG/M, TTON
^C

TTOFF.COM

R MACRO
TTOFF, TTOFF=CONFIG/M, TTOFF
^C

START.COM

R MACRO
START, START=CONFIG/M, CNTRLB/M, START
^C

USERIO.COM

R MACRO
USERIO, USERIO=CONFIG/M, USERIO
^C

CNTROL.COM

R MACRO
CNTROL, CNTROL=CONFIG/M, CNTRLB/M, CNTROL
^C

MSLINK.COM

R LINK
MSBS, MSBS=MSBS/C
DATIN, TTON/O:1/C
DATOUT, TTOFF/O:1/C
START, USERIO, CNTROL/O:1

MSDBG.COM

R MACRO
CNTROL, CNTROL/L:ME=CONFIG/M, CNTRLB/M, CNTROL
^C

CONFIG.COM

LIBRARY/MACRO CONFIG CONFIG

CNTRLB.COM

LIBRARY/MACRO CNTRLB CNTRLB

MSBS.FOR

```
        DIMENSION ISCILL(1024),ICOMM(960),IDATA(12020),ICHARS(96)
C
C  FETCH DISC DATAFILES, FILL ARRAYS, INITIALIZE VT100
C
        CALL DATIN(ISCILL,ICOMM,IDATA,ICHARS)
C
C  CALL MAIN CONTROL PROGRAM
C
        CALL START(ISCILL,ICOMM,IDATA,ICHARS)
C
C  RESET VT100, STORE DATAFILES IF REQD.
C
        CALL DATOUT(IDATA)
C
C  FINISHED
C
        STOP
        END
```

DATIN.FOR

```
      SUBROUTINE DATIN(ISCILL,ICOMM,IDATA,ICHARS)
      DIMENSION ISCILL(1024),ICOMM(960),IDATA(12020),ICHARS(96)
      DIMENSION IDUM(40)
      LOGICAL*1 ICHAR

C
C READ IN PREPROGRAMMED ROUTINE
C
      TYPE 10
      FORMAT(' ACCESSING DATA FILES')
      OPEN(UNIT=1,NAME='MSBS.DAT',TYPE='OLD',FORM='FORMATTED',
1RECORDSIZE=61)
      DO 15 I=1,960
15      ICOMM(I)=0
      DO 35 I=1,100,3
20      READ(1,30,END=38) ICOMM(I),ICOMM(I+1),ICHAR,(IDUM(J),J=1,40)
30      FORMAT(2I8,2X,A1,2X,40A2)
C
C CONVERT LETTERS TO ARRAY ADDRESS
C
35      ICOMM(I+2)=((ICHAR-32)*2)
38      CLOSE(UNIT=1)
C
C END OF READ IN. CALCULATE SINE WAVE
C
      PI=3.141592654
      DO 40 I=1,1024
40      X=PI*FLOAT(I-1)/512.0
      ISCILL(I)=INT(32767.*SIN(X))
C
C ZEROISE DATA STORAGE
C
      DO 50 I=1,12020
50      IDATA(I)=0
      DO 51 I=1,96
51      ICHARS(I)=0
C
C SET UP SYSTEM LOOP GAINS
C
      ICHARS(34)=12000      !AXIAL GAIN
      ICHARS(35)=7500      !LATERAL GAIN
      ICHARS(36)=7500      !VERTICAL GAIN
      ICHARS(38)=3500      !PITCH GAIN
      ICHARS(39)=3500      !YAW GAIN
      ICHARS(16)=850       !MAX. NUMBER OF DATA SWEEPS

C
C CALL VT100 INITIALIZING ROUTINE
C
      CALL TTON
C
C DONE HERE
C
      RETURN
      END
```

DATOUT.FOR

```
      SUBROUTINE DATOUT(IDATA)
      DIMENSION IDATA(12020)
C
C CALL VT100 RESET ROUTINE
C
      CALL TTOFF
C
C .SAVE DATA?
C
      TYPE 70
70     FORMAT(' CONTROLLER ABORTED, DATA TO FILES? (1=YES, 0=NO)')
      READ(5,*) NSAVE
      IF(NSAVE.NE.1) GO TO 80
C
C OUTPUT SOME DATA
C
      TYPE 90
90     FORMAT(' DATA BEING STORED')
      OPEN(UNIT=2,NAME='DATA.DAT',TYPE='NEW',FORM='UNFORMATTED',
1RECORDSIZE=1)
      DO 100 I=1,12000
100    WRITE (2) IDATA(I)
      CLOSE(UNIT=2)
C
80     CONTINUE
      RETURN
      END
```



```

;Line 1:      Set scrolling area (lines 11-24), move cursor to top/left home
;Lines 2-3:   Title in reverse video
;Line 4:      Define alternate character set (accessed by SI/S0)
;Lines 5-16:  Memory jogger block
;Lines 17-19: Switch display block
;Line 20:     Auto routine display block
;Line 21:     Move cursor to top of scrolling area
;Line 22:     String terminator

```

```

;
BUFF1: .ASCII <33>/[11;24r/<33>/[2J/<33>/[H/<33>
.ASCII /[7m NASA Langley Research Center 13 inch MAGNETIC /
.ASCII /SUSPENSION and BALANCE SYSTEM /<33>/[m/
.ASCII <33>/)0/<33>/[B/
.ASCII <33>/[3;1H/<16>/lqqqqqqqqqqvqqqqvqqqqqqqqvqqqk/<17><15><12>
.ASCII <16>/x/<17>/ Axial /<16>/x/<17>/ x /<16>/x/<17>/ a i r /
.ASCII <16>/x/<17>/ A /<16>/x/<17><15><12>
.ASCII <16>/x/<17>/ Lateral /<16>/x/<17>/ y /<16>/x/<17>/ b j s /
.ASCII <16>/x/<17>/ B /<16>/x/<17><15><12>
.ASCII <16>/x/<17>/ Vertical /<16>/x/<17>/ z /<16>/x/<17>/ c k t /
.ASCII <16>/x/<17>/ C /<16>/x/<17><15><12>
.ASCII <16>/x/<17>/ Pitch /<16>/x/<17>/ m /<16>/x/<17>/ e p v /
.ASCII <16>/x/<17>/ E /<16>/x/<17><15><12>
.ASCII <16>/x/<17>/ Yaw /<16>/x/<17>/ n /<16>/x/<17>/ f q w /
.ASCII <16>/x/<17>/ F /<16>/x/<17><15><12>
.ASCII <16>/mqqqqqqqqqvqqqvqqqqqqqqvqqqj/<17>
.ASCII <33>/[4;45HIntegrators=/
.ASCII <33>/[5;47HDrag Int.=/
.ASCII <33>/[6;49HOutputs=/
.ASCII <33>/[8;45HAuto routine/
.ASCII <33>/[11;1H/
.ASCII <0>

```

```

;
.END

```

```

        .TITLE  TTOFF                                ;PROGRAM TITLE
;
;+++++
; Purpose:      TTOFF performs some system and all VT-102 deinitialization
;
; Environment:  Executes once only
;
; Related modules:  Called from and returns to DATOUT
;
; Data I/O:     None
;
; Variables:    None
;-----
;
        .MCALL  CONFIG                                ;CONFIGURATION MACRO
;
;FETCH I/O DEVICE ADDRESSES
;
        CONFIG                                        ;SYSTEM CONFIGURATION AND DEFINITIONS
;
;DISPLAY ROUTINE - Resets VT-102 screen display. Uses XON/XOFF protocol
;
TTOFF:  MOV      #BUFF1,R0                            ;FETCH ADDRESS OF OUTPUT STRING
FETCH:  MOVB    (R0)+,R1                              ;FETCH NEXT CHARACTER
        BEQ     DONE                                 ;JUMP OUT IF TERMINATOR (ASCII NULL)
TEST:   TSTB   @#TPS                                  ;IS PRINTER INTERFACE READY?
        BPL     TEST                                 ;WAIT IF NOT
        MOVB   R1,@#TPB                              ;OUTPUT CHARACTER
        TSTB   @#TKS                                  ;HAS CRT SENT A CHARACTER?
        BPL     FETCH                                ;CONTINUE IF NOT
        MOVB   @#TKB,R2                              ;FETCH CHARACTER IF AVAILABLE
        CMPB   #23,R2                                ;IS THIS AN XOFF?
        BNE    FETCH                                ;CONTINUE IF NOT
XON:    TSTB   @#TKS                                  ;HAS CRT SENT AN XON?
        BPL     XON                                 ;WAIT FOR ONE IF NOT
        MOVB   @#TKB,R2                              ;FETCH CHARACTER
        CMPB   #21,R2                                ;MAKE SURE IT IS AN XON
        BNE    XON                                 ;KEEP WAITING IF NOT
        BR     FETCH                                ;RESUME OUTPUT IF IT WAS XON
;
;RESTORE SOME SYSTEM INTERRUPTS
;
DONE:   BIS     #100,@#TKS                            ;RESTORE KEYBOARD INTERRUPTS
        BIS     #100,@#LCLK                          ;RESTART LINE CLOCK
        RTS    PC
;
;
BUFF1:  .ASCII  <33>/C1;24r/                          ;Set whole screen scrolling
        .ASCII  <33>/C24;1H/                          ;Cursor to bottom line
        .ASCII  <0>                                    ;String terminator
;
        .END

```

USERIO.MAC

```

        .TITLE  USERIO                      ;PROGRAM TITLE
;
;+++++
; Purpose:      USERIO handles all real-time keyboard input and CRT output
;
; Environment:  Executes continuously, unless interrupted by hardware. No CRT
;              handshaking employed, the rate of character output assumed to
;              be slowed by repeated interruption of execution
;
; Related modules:  System hardware interrupts are initialized by START,
;                  CNTR0L executes once per hardware interrupt
;
; Data I/O:      All I/O takes place via the command array CHARS
;
; Variables:     CHARS array specified seperately. MASK is used to block
;               certain illegal repeated inputs, only Bits 0-2 used:
;               Bit 0 - Numeric value received from keyboard (0=No, 1=Yes)
;                   1 - Implicit or explicit sign received
;                   2 - Code letter received
;-----
;
        .GLOBL  CHARS,COMMAN                ;DATA AND COMMAND ARRAYS
;
        .MCALL  CONFIG                      ;REQUIRED MACROS
;
;FETCH SYSTEM CONFIGURATION
;
        CONFIG                              ;SYSTEM CONFIGURATION AND DEFINITIONS
;
;INPUT CHARACTER FROM KEYBOARD AND VECTOR TO ROUTINES
;
USERIO: JSR    PC,FETCH                      ;FETCH INPUT CHARACTER
        CMP    #177,R0                      ;DELETE?
        BEQ    DELETE                       ;GO TO DELETE HANDLER IF DELETE
        CMP    #100,R0                     ;LETTER?
        BMI    LETTER                       ;GO TO LETTER HANDLER IF LETTER
        CMP    #77,R0                      ;QUERY?
        BEQ    QUERY                        ;GO TO QUERY HANDLER IF QUERY
        CMP    #71,R0                      ;LOOK FOR SOME ILLEGAL CHARACTERS
        BMI    USERIO                     ;GO BACK IF ILLEGAL
        CMP    #57,R0                      ;NUMBER?
        BMI    NUMBER                       ;GO TO NUMBER HANDLER IF NUMBER
        CMP    #55,R0                      ;MINUS?
        BEQ    MINUS                        ;GO TO MINUS HANDLER IF MINUS
        CMP    #15,R0                      ;CARRIAGE RETURN?
        BEQ    RETURN                       ;GO TO CARRIAGE RETURN HANDLER
        CMP    #33,R0                      ;ESCAPE?
        BNE    USERIO                     ;GO BACK SINCE ALL HERE ARE ILLEGAL
ESCAPE: RTS    PC                          ;(EXCEPT ESCAPE) RETURN TO START MACRO
;
;DELETE - Clear screen input line and data
;
DELETE: MOV    #OUT11,R2                    ;FETCH ADDRESS OF LINE BLANKER
        JSR    PC,OUTSTR                    ;DO IT
        CLR    KEYNUM                      ;ZEROISE KEYBOARD INPUT
        MOV    #177,KEYLET                 ;KEYBOARD LETTER STARTS AS DELETE
        CLR    MASK                        ;CLEAR INPUT MASK
        JMP    USERIO                     ;BACK TO START
;
;LETTER HANDLER - Store legal letter code
;

```

```

LETTER: BIT      #4,MASK          ;ONLY ONE LETTER ALLOWED
        BNE      USERIO         ;OTHERWISE BACK TO START
        BIS      #7,MASK         ;WE HAVE LETTER, BLOCK NUMBER/SIGN
        JSR      PC,PRINSC      ;ECHO CHARACTER
        SUB      #40,R0         ;CONVERT TO ARRAY OFFSET
        ASL      R0              ;CONVERT TO WORD ADDRESSING
        MOV      R0,KEYLET      ;STORE LETTER CODE
        JMP      USERIO        ;JUMP BACK TO START
;
;MINUS - Negate present input value
;
MINUS:  BIT      #6,MASK          ;CHECK IF ACCEPTABLE
        BNE      USERIO         ;JUMP OUT IF NOT
        JSR      PC,PRINSC      ;ECHO CHARACTER
        NEG      KEYNUM          ;NEGATE KEYBOARD INPUT
        BIS      #2,MASK         ;WE HAVE A SIGN
        JMP      USERIO        ;DONE HERE
;
;NUMBER - Multiply up present input value
;
NUMBER: BIT      #6,MASK          ;CHECK IF ACCEPTABLE
        BNE      USERIO         ;JUMP OUT IF NOT
        BIS      #1,MASK         ;WE HAVE NUMBER
        JSR      PC,PRINSC      ;ECHO CHARACTER
        SUB      #60,R0         ;CONVERT R0 TO ACTUAL NUMBERS
        MOV      KEYNUM,R1      ;FETCH PRESENT INPUT VALUE ACCUMULATOR
        MUL      #10.,R1        ;INCREMENT EXPONENT
        ADD      R0,R1          ;ADD IN PRESENT DIGIT
        MOV      R1,KEYNUM      ;RESTORE KEYNUM
        JMP      USERIO        ;GO BACK TO START
;
;QUERY ROUTINE - Input letter code to query then output relevant value to CRT
;
QUERY:  BIT      #7,MASK          ;ACCEPT ONLY IF FIRST CHARACTER
        BNE      USERIO         ;OTHERWISE BACK TO START
        JSR      PC,PRINSC      ;ECHO CHARACTER
GET2:   JSR      PC,FETCH        ;GET ANOTHER INPUT CHARACTER
        CMP      #33,R0         ;ESCAPE?
        BEQ      ESCAPE        ;GO TO ESCAPE HANDLER IF YES
        CMP      #37,R0         ;LOOK FOR ILLEGAL CODE
        BPL      GET2          ;JUMP OUT IF ILLEGAL
        JSR      PC,PRINSC      ;ECHO LEGAL CHARACTER
        SUB      #40,R0         ;CONVERT TO ARRAY OFFSET
        ASL      R0              ;CONVERT TO WORD ADDRESSING
        MOV      CHARS(R0),R0   ;FETCH QUERIED VALUE
        JSR      PC,DECOUT      ;PUSH OUT DECIMAL VALUE
        JSR      PC,CARRLF     ;OUTPUT CR/LF
        JMP      USERIO        ;BACK TO START
;
;CARRIAGE RETURN - Output CR/LF, then enter Screen Update section
;
RETURN: MOV      KEYLET,R0       ;FETCH LETTER CODE
        MOV      KEYNUM,CHARS(R0);LOAD KEYBOARD INPUT TO CHARS ARRAY
        JSR      PC,CARRLF     ;OUTPUT CR/LF
        CLR      KEYNUM        ;ZEROISE KEYBOARD INPUT
        MOV      #177,KEYLET    ;KEYBOARD LETTER STARTS AS DELETE
        CLR      MASK          ;CLEAR INPUT MASK
;
;SCREEN UPDATE - Save cursor position before starting work on displays
;
        MOV      #OUT1,R2      ;FETCH ADDRESS OF CURSOR SAVE

```

```

        JSR      PC,OUTSTR      ;DO IT
;
;UPDATE POSITION DISPLAY - Pseudo real-time position sample display area
;
CRSMOV: MOV      #OUT2,R2      ;FETCH ADDRESS OF CURSOR MOVE COMMAND
        JSR      PC,OUTSTR      ;DO IT
        MOVB     OUT3,R1      ;FETCH CURSOR LINE NUMBER (START AT 4)
        SUB      #60,R1      ;CONVERT TO ACTUAL NUMBERS
        CMPB     #6,R1      ;ABOVE 6? (TRYING TO SKIP ROLL DATA)
        BPL      DISPLC      ;JUMP AROUND IF NOT
        INC      R1          ;SKIP 1 ARRAY ELEMENT FOR ROLL
DISPLC: ASL      R1          ;CONVERT TO WORD ADDRESSING
        ADD      #4,R1      ;CONVERT TO ARRAY OFFSET
        MOV      CHARS(R1),R0 ;R0 CONTAINS ARRAY VALUE TO OUTPUT
        JSR      PC,DECOUT     ;OUTPUT ARRAY VALUE
        MOV      #OUT5,R2     ;FETCH ADDRESS OF DUMMY SPACES
        JSR      PC,OUTSTR     ;PUSH OUT SPACES (TO MASK OLD NUMBERS)
        INCB     OUT3        ;INCREMENT LINE NUMBER COUNTER
        CMPB     #70,OUT3     ;IS IT PAST LAST LINE? (END AT 8)
        BPL      CRSMOV      ;LOOP BACK IF NOT
LASTLN: MOVB     #64,OUT3     ;RESET LINE NUMBER COUNTER (ASCII 4)
;
;SWITCH DISPLAY UPDATE - Program switch display area
;
        MOV      #OUT6,R2     ;MOVE TO MAIN INTEGRATOR UPDATE
        JSR      PC,OUTSTR     ;DO IT
        MOV      INTFLG,R0    ;ARE MAIN INTEGRATORS ON?
        JSR      PC,ONOFF     ;OUTPUT ON/OFF MESSAGE
        MOV      #OUT9,R2     ;MOVE TO DRAG INTEGRATOR UPDATE
        JSR      PC,OUTSTR     ;DO IT
        MOV      DRGFLG,R0    ;IS DRAG INTEGRATOR ON?
        JSR      PC,ONOFF     ;OUTPUT ON/OFF MESSAGE
        MOV      #OUT10,R2    ;MOVE TO OUTPUT UPDATE
        JSR      PC,OUTSTR     ;DO IT
        MOV      OUTFLG,R0    ;ARE OUTPUTS ON?
        JSR      PC,ONOFF     ;OUTPUT ON/OFF MESSAGE
;
;LOAD CORRECT AUTO ROUTINE and DISPLAY
;
        TST      AUTFLG      ;IS A ROUTINE ALREADY RUNNING?
        BNE      RSTORE     ;IGNORE POSSIBLE REQUEST IF YES
        MOV      AUTNUM,R1    ;FETCH AUTO ROUTINE NUMBER
        BIC      #177774,R1   ;TRUNCATE ROUTINE NUMBER TO 0-3
        MOV      R1,AUTNUM    ;UPDATE ROUTINE NUMBER
        MUL      #480.,R1     ;CONVERT TO BYTE ADDRESSING
        ADD      COMMAN+4,R1  ;ADD ADDRESS OFFSET
        MOV      R1,COMMAN+2  ;UPDATE 1ST ARCHIVE POINTER
        MOV      R1,COMMAN    ;UPDATE RUN-TIME POINTER
        MOV      #OUT12,R2    ;MOVE TO AUTO ROUTINE NUMBER
        JSR      PC,OUTSTR     ;DO IT
        MOV      AUTNUM,R0    ;FETCH AUTO ROUTINE NUMBER AGAIN
        JSR      PC,DECOUT     ;OUTPUT ROUTINE NUMBER
;
RSTORE: MOV      #OUT4,R2     ;FETCH ADDRESS OF CURSOR RESTORE COMMAND
        JSR      PC,OUTSTR     ;DO IT
DONE:   JMP      USERIO      ;GO BACK TO START
;
;SUBROUTINES - OUTSTR, DECOUT, ONOFF, FETCH, CARRLF, PRINSC
;
;SUBROUTINE OUTSTR - Send stored ASCII string to CRT
;

```

```

OUTSTR: MOVB    (R2)+,R0          ;FETCH FIRST CHARACTER
        BEQ     ENDSTR          ;JUMP OUT IF TERMINATOR
TEST2:  TSTB   @#TPS            ;IS CRT READY?
        BPL    TEST2           ;WAIT IF NOT
        MOVB   R0,@#TPB        ;OUTPUT CHARACTER
        BR     OUTSTR          ;LOOP BACK FOR NEXT CHARACTER
ENDSTR: RTS     PC              ;RETURN FROM SUBROUTINE
;
;SUBROUTINE DECOUT - Decode value in NUMBUF and send to CRT as ASCII numbers
;
DECOUT: MOV     NUMBUF,R2        ;GET ADDRESS OF NUMBER STORE
        TST    R0              ;IS THE VALUE NEGATIVE?
        BPL    DIVIDE         ;JUMP IF NOT
        NEG    R0              ;CHANGE SIGN TO POSITIVE FOR OUTPUT
        MOVB   #55,(R2)+       ;PUSH MINUS INTO NUMBER STORE
DIVIDE: MOV     R0,R1           ;PUSH VALUE DOWN TO LOW HALF OF DIVIDEND
        CLR    R0              ;CLEAR HIGH HALF OF DIVIDEND
        DIV    #10.,R0         ;DIVIDE BY 10, REMAINDER IN R1
        ADD    #60,R1         ;CONVERT TO ASCII
        MOVB   R1,(R2)+       ;STORE CHARACTER AS OCTAL BYTE
        TST    R0              ;ARE WE DOWN TO ZERO YET?
        BNE    DIVIDE         ;LOOP BACK IF ANY MORE DIGITS TO COME
        MOVB   #40,(R2)+       ;PUSH SPACE INTO NUMBER STORE
NUMOUT: TSTB   @#TPS            ;IS CRT READY?
        BPL    NUMOUT         ;WAIT IF NOT
        MOVB   -(R2),@#TPB     ;OUTPUT CHARACTER
        CMP    NUMBUF,R2       ;ARE WE DONE?
        BNE    NUMOUT         ;LOOP BACK IF NOT
        RTS     PC              ;RETURN FROM SUBROUTINE
;
;SUBROUTINE ONOFF - Send ASCII "ON" or "OFF" to CRT depending on value in R0
;
ONOFF:  TST    R0              ;SWITCH ON OR OFF? (0=OFF, OTHERS ON)
        BEQ    OFF            ;GO TO OFF MESSAGE
ON:     MOV    #OUT7,R2        ;FETCH ADDRESS OF OFF MESSAGE
        JSR    PC,OUTSTR      ;DO IT
        RTS    PC              ;DONE HERE
OFF:    MOV    #OUT8,R2        ;FETCH ADDRESS OF ON MESSAGE
        JSR    PC,OUTSTR      ;DO IT
        RTS    PC              ;DONE HERE TOO
;
;SUBROUTINE FETCH - Input single character from keyboard, wait if none ready
;
FETCH:  TSTB   @#TKS            ;IS CHARACTER READY?
        BPL    FETCH          ;WAIT IF NOT
        MOVB   @#TKB,R0       ;FETCH INPUT CHARACTER
        RTS    PC              ;RETURN FROM SUBROUTINE
;
;SUBROUTINE CARRLF - Send CR/LF combination to CRT
;
CARRLF: TSTB   @#TPS            ;IS CRT READY?
        BPL    CARRLF         ;WAIT IF NOT
        MOVB   #15,@#TPB     ;OUTPUT CR
TEST:   TSTB   @#TPS            ;IS CRT READY?
        BPL    TEST           ;WAIT IF NOT
        MOVB   #12,@#TPB     ;OUTPUT LF
        RTS    PC              ;RETURN FROM SUBROUTINE
;
;SUBROUTINE PRINSC - Send (echo) ASCII value to CRT
;
PRINSC: TSTB   @#TPS            ;IS CRT READY?

```

```

        BPL      PRINSC      ;WAIT IF NOT
        MOVB    R0,@#TPB    ;OUTPUT CHARACTER
        RTS     PC          ;RETURN FROM SUBROUTINE
;
;
NUMBUF: .BLKB   10          ;Dummy storage for values from query routine
KEYNUM: .WORD   0          ;Keyboard input value
KEYLET: .WORD  177        ;Keyboard input letter, starts as DELETE
MASK:   .WORD   0          ;Input vectoring mask
;
OUT1:   .ASCII  <33>/7/<0>      ;Save cursor position
OUT2:   .ASCII  <33>/[      ;Start cursor move command
OUT3:   .ASCII  /4;30H/<0>      ;Move cursor to line *, column 30
OUT4:   .ASCII  <33>/8/<0>      ;Restore cursor position
OUT5:   .ASCII  <40><40><40><40><0> ;Spaces for blanking previous value
OUT6:   .ASCII  <33>/[4;57H/<0> ;Move cursor to INTEGRATOR=
OUT7:   .ASCII  /ON /<0>       ;ON message
OUT8:   .ASCII  /OFF/<0>       ;OFF message
OUT9:   .ASCII  <33>/[5;57H/<0> ;Move cursor to DRAG INT.=
OUT10:  .ASCII  <33>/[6;57H/<0> ;Move cursor to OUTPUTS=
OUT11:  .ASCII  <33>/[2K/<15><0> ;Blank present input line
OUT12:  .ASCII  <33>/[8;58H/<0> ;Move cursor to AUTO ROUTINE
;
        .END

```

```

      .TITLE  START                                ;PROGRAM TITLE
;
;+++++
; Purpose:   Access data required by real-time code, initialize hardware
;           interrupts, provide system error traps
;
; Environment: Executes once only, some system interrupts already disabled
;           prior to module entry
;
; Related modules: Called by MSBS, calls USERIO, with interrupt to CNTR0L
;           activated
;
; Data I/O:   Accesses OSCILL, DATA, COMMAN and CHARS arrays from FORTRAN.
;           CHARS array is copied to local version for faster real-time
;           addressing. All arrays remain valid on exit
;
; Variables:  ICHARS is FORTRAN CHARS array, CHARS is local CHARS copy
;-----
;
;           .GLOBL  CHARS,DATA,OSCILL,COMMAN      ;DATA ARRAYS
;
;           .MCALL  .TRPSET,.EXIT,.PRINT         ;SYSTEM MACROS
;           .MCALL  CONFIG,DFCHAR,OUTPUT        ;CONFIGURATION MACRO
;
;FETCH I/O DEVICE ADDRESSES
;
;           CONFIG                                ;SYSTEM CONFIGURATION AND DEFINITIONS
;
;SAVE DATA ACROSS FORTRAN CALLING INTERFACE
;
START:: TST      (R5)+                            ;SKIP SUBROUTINE ARGUMENT COUNT
      MOV      (R5)+,OSCILL                       ;STORE ADDRESS OF OSCILLATOR ARRAY
      MOV      (R5),COMMAN                        ;STORE ADDRESS OF COMMAND ARRAY
      MOV      (R5),COMMAN+2                     ;ARCHIVE COMMAND ARRAY ADDRESS
      MOV      (R5)+,COMMAN+4                   ;TWICE
      MOV      (R5),DATA                         ;STORE ADDRESS OF DATA ARRAY
      MOV      (R5)+,DATA+2                     ;ARCHIVE ARRAY ADDRESS
      MOV      (R5)+,ICHARS                     ;STORE ADDRESS OF CHARS ARRAY
;
;COPY ICHARS ARRAY TO LOCAL ARRAY CHARS
;
      CLR      R1                                ;CLEAR INDEX REGISTER
      MOV      ICHARS,R0                        ;FETCH ADDRESS OF ICHARS ARRAY
COPY:  MOV      (R0)+,CHARS(R1)                  ;COPY VALUE ACROSS TO CHARS ARRAY
      ADD      #2.,R1                            ;INCREMENT CHARS ARRAY INDEX
      CMP      #190.,R1                         ;ARE WE AT THE LAST ELEMENT YET?
      BPL     COPY                              ;LOOP BACK IF NOT
;
;ACTIVATE RUN-TIME TRAP HANDLER
;
      .TRPSET #EMTARG,#SRESET                   ;CALL RT11 TRAP CATCHER
;
;SET UP LOOP RATE CLOCK, WITH INTERRUPTS ENABLED
;
      MOV      #CNTR0L,@#CL2V                   ;SET UP NEW CLOCK VECTOR
      MOV      #300,@#CL2V+2                   ;PROCESSOR STATUS WORD FOR INTERRUPT
;                                           ;PROCESSOR RUNS AT PRIORITY 6 DURING
;                                           ;INTERRUPT
      MOV      #170275,@#CL2B                  ;CLOCK COUNTER IS 3906 (DEC.) COUNTS
      MOV      #113,@#CL2S                     ;SET CLOCK FUNDAMENTAL TO 1 MHZ
;                                           ;LOOP RATE IS 256 HZ

```



```

; INTERRUPTS ENABLED
; CALL KEYBOARD HANDLER - CNTR0L will interrupt in 1/256 seconds
;
;      JSR      PC,USERIO          ;CALL TO USERIO, RETURN ON CLOSEDOWN
;
CLSDWN: BIC      #100,@#CL2S      ;TURN OFF CLOCK INTERRUPTS
        CLR      R0              ;LOAD R0 WITH ZERO CURRENT DEMAND
        OUTPUT  0              ;KILL CHANNEL 0
        OUTPUT  1              ;AND 1
        OUTPUT  2              ;ETC.
        OUTPUT  3              ;
        OUTPUT  4              ;
;
        CMP      #340,@#PSW      ;IS THIS A CLOSEDOWN OR A CRASH?
        BEQ      ABNDN          ;JUMP IF THIS IS A CRASH
        RTS      PC             ;RETURN TO FORTRAN IF NOT
;
ABNDN:  .EXIT                  ;SYSTEM CRASH
;
;SYSTEM SHUTDOWN WITH ERROR DETECTED
;
SRESET: MOV      #340,@#PSW      ;RAISE PROCESSOR STATUS TO 7
        .PRINT  #TRAPM          ;PRINT MESSAGE TO CRT
        BR      CLSDWN         ;JUMP TO CLOSEDOWN ROUTINE
        .EXIT
;
OSCILL: .WORD    0              ;ADDRESS OF OSCILLATOR ARRAY
ICHARS: .WORD    0              ;COMMAND DATA ARRAY ADDRESS
DATA:   .WORD    0,0           ;ADDRESS AND ARCHIVE OF DATA ARRAY
COMMAN: .WORD    0,0,0        ;ADDRESS OF PREPROGRAMMED ARRAY
CHARS:  .BLKW   96.           ;COPY OF COMMAND ARRAY DATA
;
DFCHAR
;
EMTARG: .WORD    0,0
TRAPM:  .ASCIZ  '/?ABORT - TRAPPED TO 4 OR 10?/'
        .END

```

```

        .TITLE CNTROL                                ;PROGRAM TITLE
;
;+++++
; Purpose:      CNTROL performs all real-time control and data acquisition
;
; Environment:  Starts following clock interrupt, executes once, returns to
;              previously executing code
;
; Related modules:  USERIO is the module interrupted and returned to. Only
;                  R0, R1 and R2 are preserved by CNTROL.
;
; Data I/O:     OSCILL contains sinewave data. DATA is used for sequential
;              storage of real-time data. CHARS carries in keyboard commands
;              in near real-time. COMMAN carries in preprogrammed routines.
;
; Variables:    OUTFLG is output on/off flag (0=off, otherwise on)
;              INTFLG is main integrator flag (h)
;              DRGFLG is axial integrator flag (H)
;              DATFLG is data acquisition flag (L, number of sweeps)
;              OVRFLW is number of data sweeps left before overflow
;              SAMPLE is number of channels of real-time data (dummy var.)
;              AUTFLG is number of auto routines to run (g)
;              AUTNUM is loaded auto routine number (G)
;-----
;
; .GLOBL CHARS,DATA,OSCILL,COMMAN                    ;DATA ARRAYS
;
; .MCALL TRANSL,ROTATE,PHASEF,OUTPUT                 ;AND REQUIRED MACROS
; .MCALL INTEG,INTEG2,ADST,DAST                      ;
; .MCALL PREPRG,FLPFLP,OUTPU2,ADCLS                  ;
; .MCALL CONFIG,ADST2,ADCLS2,DIGSCN                  ;
;
; INITIALIZE LABELS
;
; CONFIG                                ;SYSTEM CONFIGURATION
; SAMPLE=0                              ;SET DATA ACQUISITION CHANNEL COUNTER
;
; *****
; MAIN LOOP START, CALLED FROM CLOCK INTERRUPT
;
CNTROL:;BIC      #200,@#CL2S                    ;CLEAR INTERRUPT FLAG OF CLOCK
;      MOV      R0,-(SP)                       ;PUSH R0
;      MOV      R1,-(SP)                       ;AND R1
;      MOV      R2,-(SP)                       ;AND R2
;
; CHANGE R6 TO DATA STACK
;
;      MOV      SP,STACK                       ;ARCHIVE SYSTEM STACK POINTER
;      MOV      DATA,SP                       ;SET UP DATA STACK
;
; INCREMENT MASTER OSCILLATOR
;
;      INC      OSC                            ;INCREMENT MASTER COUNTER
;      BIC      #176000,OSC                    ;TRUNCATE COUNTER TO 10 BITS
;
; *****
; COMPUTE MODEL POSITION AND ATTITUDE
;
;      DIGSCN                                ;SCAN DIGITAL INPUT PORTS
;
;      MOV      AL,R1                          ;FETCH POSITION SENSOR DATA

```

```

MOV      FL,R5          ;AL=AFT LEFT, FL=FORWARD LEFT
MOV      AU,R3          ;AU=AFT UPPER, FU=FORWARD UPPER
MOV      FU,R4          ;AX=AXIAL
MOV      AX,R2          ;
;
;VERTICAL
MOV      #1777,R0       ;FETCH VERTICAL OFFSET (1023. PIXELS)
SUB      R5,R0          ;SUBTRACT FL
SUB      R1,R0          ;AND AL
MOV      R0,VERT        ;STORE VERTICAL POSITION
DAST    R0              ;STORE AS DATA
;
;PITCH
MOV      R5,R0          ;FETCH FL
SUB      R1,R0          ;SUBTRACT AL
MOV      R0,PITCH       ;STORE PITCH ATTITUDE
DAST    R0              ;STORE AS DATA
;
;LATERAL
MOV      #1777,R0       ;FETCH LATERAL OFFSET (1023. PIXELS)
SUB      R3,R0          ;SUBTRACT AU
SUB      R4,R0          ;AND FU
MOV      R0,LAT         ;STORE LATERAL POSITION
DAST    R0              ;STORE AS DATA
;
;YAW
MOV      R3,R0          ;FETCH AU
SUB      R4,R0          ;SUBTRACT FU
MOV      R0,YAW         ;STORE YAW ATTITUDE
DAST    R0              ;STORE AS DATA
;
;AXIAL
MOV      R2,R0          ;FETCH AX
SUB      #777,R0        ;SUBTRACT AXIAL OFFSET
ASL      R0              ;TIMES 2 FOR SIMILAR SENSITIVITY
MOV      R0,AXIAL       ;STORE AXIAL POSITION
DAST    R0              ;STORE AS DATA
;
; LOOP RATE INDICATOR FLIP
;
      FLPFLP  5,2000      ;SETS CHANNEL 5 D/A TO -2.5V APPROX.
;
;*****
; COMPENSATOR, WITH INTERLEAVED DATA ACQUISITION
;
;START A/D CHANNEL 0
      ADST    0          ;CHANNEL 0 IS DRAG
;
;VERTICAL
      PHASEF  VERT,VT,0.64,17.9,3.11 ;PHASE ADVANCE VERTICAL POSITION
      TRANSL  CHARS,OSCILL,3          ;ADD IN USER DEMANDS
      MUL     VG,R2                    ;VERTICAL GAIN. DROP LOW ORDER PRODUCT
      INTEG  CHARS,VINT                ;INTEGRATOR
      MOV     R2,VO                     ;STORE VERTICAL DEMAND
;
;COLLECT A/D AND START NEXT
      ADCLS
      ADST    1          ;CHANNEL 1 IS BIAS 1
;
;PITCH
      PHASEF  PITCH,PT,0.64,17.9,3.11 ;PHASE ADVANCE PITCH ATTITUDE

```

```

        ROTATE CHARS,OSCILL,2      ;ADD IN USER DEMANDS
        MUL    PG,R2                ;PITCH GAIN
        INTEG  CHARS,PINT          ;INTEGRATOR
        MOV    R2,P0                ;STORE PITCH DEMAND
;
;COLLECT A/D AND START NEXT
        ADCLS
        ADST    2                    ;CHANNEL 2 IS BIAS 2
;
;LATERAL
        PHASEF LAT,LT,0.64,17.9,3.11 ;PHASE ADVANCE LATERAL POSITION
        TRANSL CHARS,OSCILL,2      ;ADD IN USER DEMANDS
        MUL    LG,R2                ;LATERAL GAIN
        INTEG  CHARS,LINT          ;INTEGRATOR
        MOV    R2,LO                ;STORE LATERAL DEMAND
;
;COLLECT A/D AND START NEXT
        ADCLS
        ADST    3                    ;CHANNEL 3 IS BIAS 3
;
;YAW
        PHASEF YAW,YT,0.64,17.9,3.11 ;PHASE ADVANCE YAW ATTITUDE
        ROTATE CHARS,OSCILL,3      ;ADD IN USER DEMANDS
        MUL    YG,R2                ;YAW GAIN
        INTEG  CHARS,YINT          ;INTEGRATOR
        MOV    R2,YO                ;STORE YAW DEMAND
;
;COLLECT A/D AND START NEXT
        ADCLS
        ADST    4                    ;CHANNEL 4 IS BIAS 4
;
;AXIAL
        PHASEF AXIAL,AT,0.64,17.9,3.11 ;PHASE ADVANCE AXIAL POSITION
        TRANSL CHARS,OSCILL,1      ;ADD IN USER DEMANDS
        MUL    AG,R2                ;AXIAL GAIN
        INTEG2 CHARS,AINT          ;INTEGRATOR
        MOV    R2,AO                ;STORE AXIAL DEMAND
;
;COLLECT A/D AND START
        ADCLS
        ADST    5                    ;CHANNEL 5 IS CONTROL 1
;
; MODEL OUT DETECTOR
;
        TST    OUTFLG                ;IS J 0?
        BNE   MODIN                 ;JUMP TO OUTPUTS IF NOT
MODOUT: CLR    VO                    ;CLEAR ALL FINAL DEMANDS
        CLR    P0                    ;IF MODEL IS OUT
        CLR    LO
        CLR    YO
        CLR    AO
        CLR    VINT                  ;INCLUDING HIGH ORDER
        CLR    PINT                  ;INTEGRATOR ACCUMULATORS
        CLR    LINT
        CLR    YINT
        CLR    AINT
        CLR    INTFLG                ;TURN MAIN INTEGRATORS OFF
        CLR    DRGFLG               ;ALSO DRAG INTEGRATOR
MODIN:
;
;*****

```

```

; OUTPUTS WITH INTERLEAVED DATA ACQUISITION
;
;E/M 1, AFT PORT, OUTPUT
MOV      YO,R0          ;FETCH YAW DEMAND
SUB      VO,R0          ;SUBTRACT VERTICAL DEMAND
SUB      PO,R0          ;SUBTRACT PITCH DEMAND
SUB      LO,R0          ;SUBTRACT LATERAL DEMAND
OUTPUT   0              ;CALL OUTPUT ROUTINE

;
;COLLECT AND START A/D
ADCLS
ADST     6              ;CHANNEL 6 IS CONTROL 2
;
;E/M 2, FORWARD PORT, OUTPUT
MOV      PO,R0          ;LOAD R0 WITH PITCH DEMAND
SUB      VO,R0          ;SUBTRACT VERTICAL
SUB      LO,R0          ;SUBTRACT LATERAL
SUB      YO,R0          ;SUBTRACT YAW
OUTPUT   1              ;CALL OUTPUT ROUTINE
;
;AFT STARBOARD OUTPUT
MOV      LO,R0          ;LOAD R0 WITH LATERAL
SUB      VO,R0          ;SUBTRACT VERTICAL
SUB      PO,R0          ;SUBTRACT PITCH
SUB      YO,R0          ;SUBTRACT YAW
OUTPUT   2              ;CALL OUTPUT ROUTINE
;
;COLLECT AND START A/D
ADCLS
ADST     7              ;CHANNEL 7 IS CONTROL 3
;
;FORWARD STARBOARD OUTPUT
MOV      PO,R0          ;LOAD R0 WITH PITCH
ADD      LO,R0          ;ADD LATERAL
ADD      YO,R0          ;ADD YAW
SUB      VO,R0          ;SUBTRACT VERTICAL
OUTPUT   3              ;CALL OUTPUT ROUTINE
;
;AXIAL OUTPUT
MOV      AO,R0          ;FETCH AXIAL DEMAND
OUTPU2   4              ;CALL OUTPUT ROUTINE
;
;COLLECT A/D AND START NEXT
ADCLS
ADST2    0              ;CHANNEL (2)0 IS CONTROL 4
;
; LOOP RATE INDICATOR FLOP
;
FLPFLP   5,-2000       ;SETS CHANNEL 5 D/A TO +2.5V APPROX.
;
;COLLECT A/D
ADCLS2
;
;*****
; PRESERVE DATA ?
;
TST      DATFLG         ;IS [ SET +VE (STORE) ?
BEQ      NOSAMP         ;JUMP OUT IF NO
DEC      DATFLG         ;REDUCE CYCLE COUNTER
TST      OVRFLW         ;LOOK FOR DATA OVERFLOW
BEQ      NOSAMP         ;JUMP OUT IF YES

```

```

        DEC      OVRFLW      ;REDUCE OVERFLOW COUNTER
        BR       ENDSMP      ;KEEP DATA AND JUMP OUT
NOSAMP: SUB     #SAMPLE,SP   ;DISCARD DATA
ENDSMP:
;
;PREPROGRAMMED ROUTINE EXECUTION
;
        PREPRG   CHARS,COMMAN      ;CALL PREPROGRAMMED ROUTINE
;
;RESTORE SYSTEM STACKS, RETURN TO PREVIOUSLY EXECUTING CODE
;
JUMPST: MOV     SP,DATA          ;ARCHIVE DATA STACK POINTER
        MOV     STACK,SP        ;RESTORE SYSTEM STACK
        MOV     (SP)+,R2        ;POP R2
        MOV     (SP)+,R1        ;POP R1
        MOV     (SP)+,R0        ;POP R0
;
        RTI                    ;RETURN TO USERIO FROM INTERRUPT
;
;*****
; DECLARE VARIABLES
;
STACK:  .WORD   0                ;SYSTEM STACK POINTER
;
VO:     .WORD   0                ;VERTICAL CURRENT DEMAND
PO:     .WORD   0                ;PITCH DEMAND
LO:     .WORD   0                ;LATERAL DEMAND
YO:     .WORD   0                ;YAW DEMAND
AO:     .WORD   0                ;AXIAL DEMAND
;
VT:     .FLT2   0.,0.           ;INTERMEDIATE DATA STORAGE
PT:     .FLT2   0.,0.           ;FOR PHASE ADVANCERS
LT:     .FLT2   0.,0.           ;
YT:     .FLT2   0.,0.           ;
AT:     .FLT2   0.,0.           ;
;
VINT:   .WORD   0,0             ;INTEGRATOR ACCUMULATORS
PINT:   .WORD   0,0             ;
LINT:   .WORD   0,0             ;
YINT:   .WORD   0,0             ;
AINT:   .WORD   0,0             ;
;
        .END

```

CNTRLB.MAC

```

;*****
;
; THE MACRO ROUTINES CONTAINED IN THIS LIBRARY ARE AS FOLLOWS ::
;
;   TRANSL      X,Y,Z TRANSLATOR/OSCILLATOR
;   ROTATE      YAW,PITCH,ROLL ROTATOR/OSCILLATOR
;
;   PHASEF      PROGRAMMABLE DUAL PHASE ADVANCE
;   INTEG       FIXED PARAMETER ERROR INTEGRATOR
;   INTEG2      AUXILIARY FIXED PARAMETER (DRAG) INTEGRATOR
;
;   OUTPUT      D/A OUTPUT ROUTINE AND OVERFLOW CLAMP
;   OUTPU2      DRAG D/A OUTPUT ROUTINE
;
;   ADCL        FETCH READY A/D DATA (BOARD 1)
;   ADCL2       FETCH READY A/D DATA (BOARD 2)
;   ADCLS       FETCH A/D DATA AND STORE ON STACK (BOARD 1)
;   ADCLS2      FETCH A/D DATA AND STORE ON STACK (BOARD 2)
;
;   ADST        INITIATE A/D CONVERSION (BOARD 1)
;   ADST2       INITIATE A/D CONVERSION (BOARD 2)
;
;   TIME1       MARK TIME FROM CLOCK 1 ON STACK
;   TIME2       MARK TIME FROM CLOCK 2 ON STACK
;   DAST        STORE AVAILABLE DATA ON STACK
;
;   PREPRG      PREPROGRAMMED ROUTINE EXECUTION
;   FLPFLP     LOOP RATE INDICATOR FLIP/FLOP
;
;   DIGSCN      SCAN DIGITAL INPUT PORTS FOR POSITION DATA
;
;*****
;
;+++++
;FULL TRANSLATOR/OSCILLATOR FOR X,Y,Z;
;      Input data in R5, output to R2
;      NUM is d-o-f, 1=axial, 2=lateral, 3=vertical
;-----
;
; .MACRO  TRANSL  CHARS,OSCILL,NUM
; SUB     CHARS+(2*(NUM+127)),R5 ;ADD DC OFFSET TO R5
; SUB     CHARS+(2*(NUM+67)),R5  ;AUXILIARY DISPLACEMENT
; MOV     CHARS+(2*136),R1       ;GET MASTER COUNTER
; MOV     CHARS+(2*(NUM+121)),R2 ;LOAD PHASE
; MOV     CHARS+(2*(NUM+110)),R3 ;LOAD FREQUENCY
; MUL     R3,R1                  ;INCREMENT COUNTER FOR FREQUENCY
;                                     ;LOW ORDER PRODUCT ONLY RETAINED
; ADD     R2,R1                  ;INCREMENT COUNTER FOR PHASE
;                                     ;OVERFLOW IGNORED
; BIC     #176000,R1             ;REDUCE COUNTER TO 10 BITS
; ASL     R1                     ;CONVERT R1 TO WORD ADDRESSING
; ADD     OSCILL,R1              ;ADD OSCILLATOR BASE ADDRESS
; MOV     (R1),R2                ;GET RELEVANT OSCILLATOR VALUE
; MOV     CHARS+(2*(NUM+100)),R4 ;GET AMPLITUDE
; MUL     R4,R2                  ;OSCILLATION AMPLITUDE
;                                     ;HIGH ORDER PRODUCT USED
; NEG     R2                     ;CORRECT SIGN OF OSCILLATOR
; ADD     R5,R2                  ;ADD OSCILLATOR TO R5
; .ENDM  TRANSL
;
;

```

```

;+++++
;FULL ROTATOR/OSCILLATOR FOR ROLL/PITCH/YAW
;      Input in R5, output to R2
;      NUM is d-o-f, 1=roll, 2=pitch, 3=yaw
;-----

```

```

;
; .MACRO ROTATE CHARS,OSCILL,NUM
SUB     CHARS+(2*(NUM+113)),R5 ;INPUT DATA IN R5
SUB     CHARS+(2*(NUM+54)),R5 ;ADD DC OFFSET TO R5
MOV     CHARS+(2*136),R1      ;AUXILIARY DISPLACEMENT
MOV     CHARS+(2*(NUM+124)),R2 ;GET MASTER COUNTER
MOV     CHARS+(2*(NUM+116)),R3 ;LOAD PHASE
MUL     R3,R1                ;LOAD FREQUENCY
ADD     R2,R1                ;INCREMENT COUNTER FOR FREQUENCY
BIC     #176000,R1           ;INCREMENT COUNTER FOR PHASE
ASL     R1                    ;REDUCE COUNTER TO 10 BITS
ADD     OSCILL,R1            ;CONVERT R1 TO WORD ADDRESSING
MOV     (R1),R2              ;ADD OSCILLATOR BASE ADDRESS
MUL     R4,R2                ;GET RELEVANT OSCILLATOR VALUE
NEG     R2                    ;GET AMPLITUDE
ADD     R5,R2                ;OSCILLATION AMPLITUDE
;CORRECT SIGN OF OSCILLATOR
;ADD OSCILLATOR TO R5
;OUTPUT TO R2
.ENDM ROTATE

```

```

;
;+++++
; DUAL PHASE ADVANCER
;      Input data from SOURCE, output to R5
;      Y, Y+4. are intermediate results. L1, L2, GAIN are parameters
;      POS,NEG,FINISH are local symbols
;-----

```

```

; .MACRO PHASEF SOURCE,Y,L1,L2,GAIN,?POS,?NEG,?FINISH
;
AC0=%0 ;DEFINE FLOATING POINT REGISTERS
AC1=%1
AC2=%2
AC3=%3
AC4=%4
AC5=%5

```

```

; INPUT VALUES L1,L2 CORRESPOND TO :
;      L1=T/(T+DT)
;      L2=N*T/DT
;      GAIN=10.*((DT/T)**2)

```

OVERALL D.C. GAIN = 10 (CONVENIENCE ONLY)

APPROXIMATE VALUES FOR 256 HZ LOOP RATE AND N=10 AS FOLLOWS :

T	1	0.007	0.006	0.005	0.004	0.003
L1	1	0.64	0.61	0.56	0.51	0.43
L2	1	17.9	15.4	12.8	10.2	7.7
GAIN	1	3.11	4.24	6.10	9.54	16.95

GET INPUT VALUE

```

LDCIF SOURCE,AC0 ;AC0=DEMAND POSITION (P)
MULF #GAIN,AC0 ;CORRECT GAIN TO 8 OVERALL

```



```

;
; FIRST PHASE ADVANCE
;
    LDF      Y,AC1                ;AC1=Y1(K-1)
    ADDF     AC1,AC0              ;AC0=P+Y1(K-1)
    MULF     #L1,AC0             ;AC0=Y1(K)
    STF      AC0,Y               ;UPDATE Y1(K-1) TO Y1(K)
    SUBF     AC0,AC1             ;AC1=Y1(K-1)-Y1(K)
    MULF     #-L2,AC1           ;AC1=X*(Y1(K)-Y1(K-1))
    ADDF     AC0,AC1             ;AC1=Y2(K)
;
; SECOND
;
    LDF      Y+4.,AC2            ;REPEAT FOR SECOND
    ADDF     AC2,AC1
    MULF     #L1,AC1
    STF      AC1,Y+4.
    SUBF     AC1,AC2
    MULF     #-L2,AC2
    ADDF     AC1,AC2
;
; OVERFLOW CLAMPS
;
    CMPF     #16383.,AC2        ;COMPARE TO HALF INTEGER OVERFLOW
                                ;*****
                                ;THIS IS AN ARBITRARY VALUE AND
                                ;PROCEDURE AND IS CAPABLE OF
                                ;PRODUCING INCORRECT OR UNPREDICTABLE
                                ;RESULTS WITH UNUSUAL OUTPUT DEMAND
                                ;MIXING
                                ;*****
    CFCC
    BMI      POS                ;COPY CONDITION CODES TO CPU
    CMPF     #-16383.,AC2       ;BRANCH TO POSITIVE LIMITER
    CFCC
    BPL      NEG                ;COMPARE TO -HALF INTEGER
    STCFI    AC2,R5             ;COPY CONDITION CODES
    BR       FINISH             ;BRANCH TO NEGATIVE LIMITER
    POS:     MOV      #16383.,R5 ;STORE RESULT IN MAIN REGISTER
    BR       FINISH             ;DONE HERE
    NEG:     MOV      #-16383.,R5;POSITIVE LIMITER
    BR       FINISH             ;DONE HERE
    FINISH:  .ENDM    PHASEF    ;NEGATIVE LIMITER
;
;
;*****
;STANDARD ERROR INTEGRATOR
;
;      Source data in R2, accumulator is INT, uses "h" as flag
;      OFF is local symbol
;-----
;
    .MACRO   INTEG    CHARS,INT,?OFF
    MOV      INT,R0            ;FETCH PRESENT INTEGRATOR ACCUMULATOR
    MOV      INT+2.,R1        ;(HIGH AND LOW WORDS)
    TST      INTFLG           ;ARE MAIN INTEGRATORS ON?
    BEQ      OFF              ;JUMP FORWARD IF NO
    MOV      R2,R4            ;STORE POSITION ERROR
    CLR      R5                ;
    ASHC     #-12,R4          ;POSITION ERROR * KdT
    ADD      R5,R1            ;ADD PRESENT INTEGRATOR VALUE TO
    ADC      R0                ;INTEGRATOR ACCUMULATOR

```

```

OFF:   ADD     R4,R0           ;
      ADD     R0,R2           ;POSITION ERROR+INTEGRATOR ACCUMULATOR
      MOV     R0,INT          ;RESTORE INTEGRATOR ACCUMULATOR
      MOV     R1,INT+2.       ;
      .ENDM   INTEG          ;INTEGRATORS ARE NOT OVERFLOW LIMITED

```

```

;
;
;*****
;AUXILIARY ERROR INTEGRATOR
;      Source data in R2, accumulator is INT, uses "H" as flag
;      OFF is local symbol
;-----
;

```

```

      .MACRO  INTEG2  CHARS,INT,?OFF
MOV     INT,R0                ;FETCH PRESENT INTEGRATOR ACCUMULATOR
MOV     INT+2.,R1             ;(HIGH AND LOW WORDS)
TST     DRGFLG                ;IS DRAG INTEGRATOR ON?
BEQ     OFF                   ;JUMP FORWARD IF NO
MOV     R2,R4                 ;STORE POSITION ERROR
CLR     R5                     ;
ASHC   #-12,R4                ;POSITION ERROR * KdT
ADD     R5,R1                  ;ADD PRESENT INTEGRATOR VALUE TO
ADC     R0                     ;INTEGRATOR ACCUMULATOR
OFF:   ADD     R4,R0           ;
      ADD     R0,R2           ;POSITION ERROR+INTEGRATOR ACCUMULATOR
      MOV     R0,INT          ;RESTORE INTEGRATOR ACCUMULATOR
      MOV     R1,INT+2.       ;
      .ENDM   INTEG2          ;INTEGRATORS ARE NOT OVERFLOW LIMITED

```

```

;
;
;*****
;D/A OUTPUT ROUTINE AND OVERFLOW CLAMPER
;      Input in R0, output direct to D/A channel
;      NEGCLM, POSCLM, FINISH are local symbols
;-----
;

```

```

      .MACRO  OUTPUT  CHANN,?NEGCLM,?POSCLM,?FINISH
;
      ZEROI=6000                ;NOMINAL ZERO CURRENT D/A VALUE
;
      ADD     #ZEROI,R0         ;INCLUDE SOFTWARE BIAS FOR D/A
;                               ;2047.=0 VOLTS
;                               ;LOOK FOR NEGATIVE OVERFLOW
      BMI     NEGCLM            ;BRANCH TO NEGATIVE OVERFLOW CLAMP
      CMP     #7777,R0          ;LOOK FOR POSITIVE OVERFLOW
      BMI     POSCLM            ;BRANCH TO POSITIVE OVERFLOW CLAMP
      BR     FINISH             ;DONE HERE
POSCLM: BIS     #7777,R0        ;FORCE POSITIVE CLAMP
      BR     FINISH             ;DONE HERE
NEGCLM: BIC     #7777,R0        ;FORCE NEGATIVE CLAMP
FINISH: MOV     R0,@#<OUT+<CHANN*2>> ;OUTPUT TO D/A
;
      .ENDM   OUTPUT          ;DONE

```

```

;
;
;*****
;D/A OUTPUT ROUTINE AND OVERFLOW CLAMPER
;      Input in R0, output direct to D/A channel
;      NEGCLM, POSCLM, FINISH are local symbols
;-----
;

```



```

;
;
;+++++
;A/D BOARD 2 COLLECT ROUTINE FOR DATA ACQUISITION
;      Data from A/D board 2, placed on stack
;      TEST2 is a local symbol
;-----
;
;      .MACRO ADCLS2 ?TEST2
;      SAMPLE=SAMPLE+2                ;INCREMENT DATA COUNTER
TEST2:  TSTB   @#AD2SR                 ;IS A/D DONE ?
;      BPL    TEST2                    ;WAIT IF NO
;      MOV    @#AD2BR,(SP)+           ;PUSH DATA ON STACK (CLEARS DONE BIT)
;
;      .ENDM ADCLS2
;
;
;+++++
;START DATA ACQUISITION ON BOARD 1
;      CHANN is channel address (0-8)
;-----
;
;      .MACRO ADST CHANN
;      MOV    #(<<CHANN*400>|1),@#AD1SR ;INITIATE A/D CONVERSION
;      .ENDM ADST
;
;
;+++++
;START DATA ACQUISITION ON BOARD 2
;      CHANN is channel address (0-8)
;-----
;
;      .MACRO ADST2 CHANN
;      MOV    #(<<CHANN*400>|1),@#AD2SR ;INITIATE A/D CONVERSION
;      .ENDM ADST2
;
;
;+++++
;TIME MARK FROM CLOCK 1
;      Push time on stack
;-----
;
;      .MACRO TIME
;      SAMPLE=SAMPLE+2                ;INCREMENT DATA COUNTER
;      MOV    @#CL1B,(SP)+           ;PUSH TIME ON STACK, INC SP
;      .ENDM TIME
;
;
;+++++
;TIME MARK FROM CLOCK 2
;      Push time on stack
;-----
;
;      .MACRO TIME2
;      SAMPLE=SAMPLE+2                ;INCREMENT DATA COUNTER
;      MOV    @#CL2B,(SP)+           ;PUSH TIME ON STACK, INC SP
;      .ENDM TIME
;
;
;+++++
;DATA STORE

```

```

;          Push SOURCE data on stack
;-----
;
;          .MACRO DAST      SOURCE
SAMPLE=SOURCE+2          ;INCREMENT DATA COUNTER
MOV      SOURCE,(SP)+    ;PUSH DATA ON STACK,INCREMENT SP
.ENDM    DAST
;
;
;+++++
;PREPROGRAMMED ROUTINE EXECUTION
;          CHARS and COMMAN arrays are accessed
;          AUTO, NOAUTO, AUTCOM, AUTEND are local symbols
;-----
;
;          .MACRO PREPRG   CHARS,COMMAN,?AUTO,?NOAUTO,?AUTCOM,?AUTEND
;
AUTO:  TST      CHARS+(2*107)      ;IS AUTO ROUTINE CONTROL PARAMETER
;          ;(G) SET POSITIVE?
      BEQ      NOAUTO             ;JUMP OUT IF NOT
      TST      CHARS+(2*134)      ;IS THIS AUTO COMMAND COMPLETED?
      BNE      AUTCOM             ;JUMP OUT IF NOT
      MOV      COMMAN,R0          ;LOAD POINTER TO THIS COMMAND DURATION
      MOV      (R0)+,CHARS+(2*134);LOAD THIS COMMAND DURATION
      BEQ      AUTEND             ;JUMP IF THIS IS THE TERMINATOR
      MOV      (R0)+,R1           ;FETCH AUTO COMMAND VALUE
      MOV      (R0)+,R2           ;FETCH INSTRUCTION CODE
      MOV      R1,CHARS(R2)       ;DEPOSIT COMMAND VALUE WHERE REQUIRED
      MOV      R0,COMMAN          ;UPDATE COMMAND POINTER
      BR       NOAUTO            ;DONE HERE
;
AUTCOM: DEC     CHARS+(2*134)      ;DECREMENT DELAY COUNTER
      BR       NOAUTO            ;DONE HERE
;
AUTEND: DEC     CHARS+(2*107)      ;DECREMENT (CLEAR) AUTO COMMAND FLAG
      MOV     <COMMAN+2>,R0       ;RESTORE COMMAND POINTER
      MOV     R0,COMMAN          ;
NOAUTO:
;
;          .ENDM    PREPRG
;
;
;+++++
; LOOP RATE INDICATOR FLIP/FLOP ROUTINE
;          Preset VALUE is passed to specified D/A CHANNEL
;-----
;
;          .MACRO FLPFLP   CHANN,VALUE
MOV      #(VALUE+4000),@#<OUT+<CHANN*2>> ;OUTPUT VALUE TO D/A
.ENDM    FLPFLP
;
;
;+++++
;DIGITAL INPUT PORT SCAN FOR POSITION DATA
;          Input port assignments as follows:
;          Bits          Port      Channel Sensor
;          0-9           A          1      AL
;          10-15 +      A          1      AL
;          0-3           B          2      AX
;          4-13         B          3      AU
;          14-15 +      B          3      AU
;

```

```

;          0-7          C          4          FU
;          8-15 +      C          5          FL
;          0-1          D          5          FL
;
;          Outputs direct to AL, FU etc.
;-----

```

```

;          .MACRO  DIGSCN
;
;          MOV      @#DBRA,R3          ;FETCH DIGITAL INPUT PORT DATA
;          MOV      @#DBRB,R2          ;
;          MOV      @#DBRC,R1          ;
;          MOV      @#DBRD,R0          ;
;
;DISASSEMBLE FOR INDIVIDUAL CHANNELS
;
;          MOV      R3,R4              ;COPY PORT A
;          BIC      #176000,R4         ;MASK HIGH BITS
;          MOV      R4,AL              ;
;
;          MOV      R3,R5              ;COPY PORT A
;          MOV      R2,R4              ;AND PORT B
;          ASHC     #-10.,R4           ;JUSTIFY DATA
;          BIC      #176000,R5         ;MASK HIGH BITS
;          MOV      R5,AX              ;
;
;          MOV      R2,R4              ;COPY PORT B
;          ASH      #-4.,R4            ;JUSTIFY DATA
;          BIC      #176000,R4         ;MASK HIGH BITS
;          MOV      R4,AU              ;
;
;          MOV      R2,R5              ;COPY PORT B
;          MOV      R1,R4              ;AND PORT C
;          ASHC     #-14.,R4           ;JUSTIFY DATA
;          BIC      #176000,R5         ;MASK HIGH BITS
;          MOV      R5,FU              ;
;
;          MOV      R0,R4              ;COPY PORT D
;          MOV      R1,R5              ;AND PORT C
;          ASHC     #-8.,R4            ;JUSTIFY DATA
;          BIC      #176000,R5         ;MASK HIGH BITS
;          MOV      R5,FL              ;
;
;          .ENDM  DIGSCN

```

CONFIG.MAC

```
*****
;
; THE MACRO ROUTINES CONTAINED IN THIS LIBRARY ARE AS FOLLOWS :
;
;     CONFIG          SYSTEM I/O ADDRESS CONFIGURATION
;     DFCHAR          CHARS ARRAY DEFINITIONS
;
;*****
;*****
;SYSTEM CONFIGURATION - Defines all hardware addresses, vectors etc. No
;                       run-time content
;-----
;
;     .MACRO CONFIG
;
; PROGRAMMABLE CLOCKS
;
;     CL1S=170400      ;CLOCK 1 STATUS REGISTER
;     CL1B=CL1S+2     ;CLOCK 1 BUFFER/PRESET REGISTER
;     CL1V=440        ;CLOCK 1 INTERRUPT VECTOR
;
;     CL2S=170420      ;CLOCK 2 STATUS REGISTER
;     CL2B=CL2S+2     ;CLOCK 2 BUFFER/PRESET REGISTER
;     CL2V=450        ;CLOCK 2 INTERRUPT VECTOR
;
; A/D CONVERTERS
;
;     AD1SR=170430     ;A/D BOARD 1 STATUS REGISTER
;     AD1BR=AD1SR+2   ;A/D BOARD 1 DATA/BUFFER REGISTER
;
;     AD2SR=170440     ;A/D BOARD 2 STATUS REGISTER
;     AD2BR=AD2SR+2   ;A/D BOARD 2 DATA/BUFFER REGISTER
;
; D/A CONVERTERS
;
;     OUT=170450       ;FIRST D/A PORT (CHANNEL 0, BOARD 1)
;                       ;FOLLOWING CHANNELS IN INCREMENTS OF 2
;                       ;LAST CHANNEL IS 13 (OCTAL, BOARD 3), =170476
;
; DIGITAL I/O PORTS
;
;     CSRA=164160      ;PORT A STATUS REGISTER
;     DBRA=CSRA+2     ;PORT A DATA/BUFFER REGISTER
;     CSRB=CSRA+4     ;PORT B STATUS
;     DBRB=CSRA+6     ;PORT B DATA
;     CSRC=CSRA+10    ;PORT C STATUS
;     DBRC=CSRA+12    ;PORT C DATA
;     CSRD=CSRA+14    ;PORT D STATUS
;     DBRD=CSRA+16    ;PORT D DATA
;
; KEYBOARD/CRT
;
;     TKS=177560       ;KEYBOARD STATUS REGISTER
;     TKB=TKS+2       ;KEYBOARD DATA/BUFFER REGISTER
;
;     TPS=TKS+4       ;CRT STATUS REGISTER
;     TPB=TKS+6       ;CRT DATA/BUFFER REGISTER
;
; SYSTEM FUNCTIONS
;
;     LCLK=177546      ;SYSTEM (LINE FREQUENCY) CLOCK
```

```

        PSW=177776      ;PROCESSOR STATUS WORD
;
        .ENDM  CONFIG
;
;+++++
;CHARS ARRAY DEFINITIONS - Sets default assignments of key variables to CHARS
;                           array locations. All are GLOBAL references
;-----
;
        .MACRO  DFCHAR
;
; POSITIONS/ATTITUDES
;
        AXIAL==:CHARS+14      ;AXIAL POSITION
        LAT==:CHARS+16       ;LATERAL POSITION
        VERT==:CHARS+20      ;VERTICAL POSITION
        PITCH==:CHARS+24     ;PITCH ATTITUDE
        YAW==:CHARS+26       ;YAW ATTITUDE
;
; LOOP GAINS
;
        AG==:CHARS+102      ;AXIAL GAIN
        LG==:CHARS+104      ;LATERAL GAIN
        VG==:CHARS+106      ;VERTICAL GAIN
        PG==:CHARS+112      ;PITCH GAIN
        YG==:CHARS+114      ;YAW GAIN
;
; POSITIONS SENSORS
;
        AL==:CHARS          ;POSITION SENSOR SIGNALS
        AX==:CHARS+2        ;AL=AFT LEFT,
        AU==:CHARS+4        ;FU=FORWARD UPPER,
        FU==:CHARS+6        ;ETC.
        FL==:CHARS+10      ;
;
; PROGRAM SWITCHES
;
        DATFLG==:CHARS+(2*73) ;DATA ACQUISITION FLAG ("I")
        OVRFLW==:CHARS+(2*17) ;DATA OVERFLOW FLAG ("/")
        OUTFLG==:CHARS+(2*75) ;OUTPUT ON/OFF FLAG ("J")
        INTFLG==:CHARS+(2*110) ;MAIN INTEGRATOR FLAG ("h")
        DRGFLG==:CHARS+(2*50) ;DRAG INTEGRATOR FLAG ("H")
        AUTFLG==:CHARS+(2*107) ;AUTO ROUTINE FLAG ("g")
        AUTNUM==:CHARS+(2*47)  ;AUTO ROUTINE NUMBER ("G")
;
; MISCELLANEOUS
;
        OSC==:CHARS+(2*136)   ;OSCILLATOR MASTER COUNTER ("~")
;
        .ENDM  DFCHAR

```


APPENDIX C -PDP 11/23 SYSTEM CONFIGURATION

CPU - PDP 11/23 PLUS with FPF11 floating point processor

Operating system - RT11 V5.0

Languages - FORTRAN V2.6, MACRO V5.0

User I/O devices - VT102 CRT, LA100 printer, HP7475 plotter

Real-time I/O - 16/32 A/D channels (differential/single ended)
 12 D/A channels
 64 digital I/O lines

Clocks - 2 programmable

Serial I/O - 6 lines (console, printer, plotter, 3 spare)

Memory - 256kB (only 56kB normally available for real-time software)

SYSTEM CONFIGURATION and MODULE SPECIFICATION

 SYSTEM SERIAL NUMBER - BT02952

DEC - 84065849X

11/23 BE - System box

Module	Etch/rev	Function	Address	Vector
M8189	C A	11/23 CPU		
M8188	C C	FPF-11 floating point		
M8067	C A	MSV11 memory		
M8061	C C	RLV22 disk controller		
M8377		KXT11-C processor	160100	
M4002	J M	KWV11-C clock	170400	440
M4002		KWV11-C clock	170420	450
M8043	E M	DLV11-J serial I/O	176500	300
			510	310
			520	320
			530	330

M9404 Bus extender

BA11-SE expansion box

M9405		Bus extender		
A8000	E H	ADV11-C A/D	170430	400
A8000		ADV11-C A/D	170440	420
A6006	E F	AAV11-C D/A	170450	
A6006	E F	AAV11-C D/A	460	
A6006		AAV11-C D/A	470	
M8049		DRV11-J Digital I/O	164160	Prog.

Serial line baud rates are:

RT11 unit 1	Console	9600
	Printer	9600
2	SLU1	300
3	SLU2 (plotter)	9600
4	SLU3	9600
5	SLU4	9600

APPENDIX D

Individual hardware subsystems can be tested and exercised to some degree without the MSBS operating and without the controller software running. Such procedures are frequently necessary for isolation of hardware or software malfunctions.

Position sensors

"TSTDIO" scans the digital input ports and decodes the input data to position sensor counts, following the same procedures documented in the controller module "CNTRLB" (APPENDIX 2). The position sensor "backup" clock or the computer clock must be running in order to generate valid sensor outputs. "TSTDIO" does not start or stop the computer clocks.

The Figure shows the normal allocation of position sensor channels:

- 1 - Aft lateral (AL)
- 2 - Axial (AX)
- 3 - Aft upper (AU)
- 4 - Forward upper (FU)
- 5 - Forward lateral (FL)

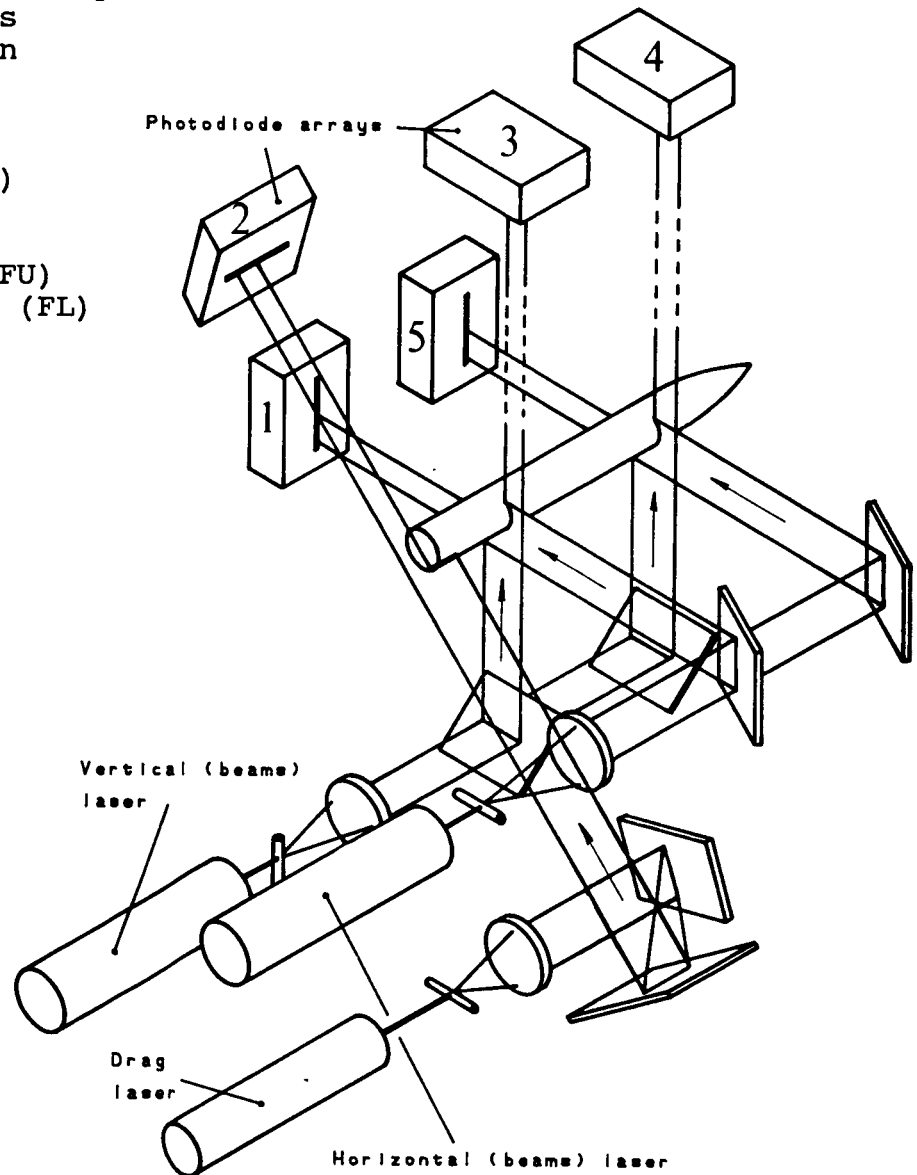


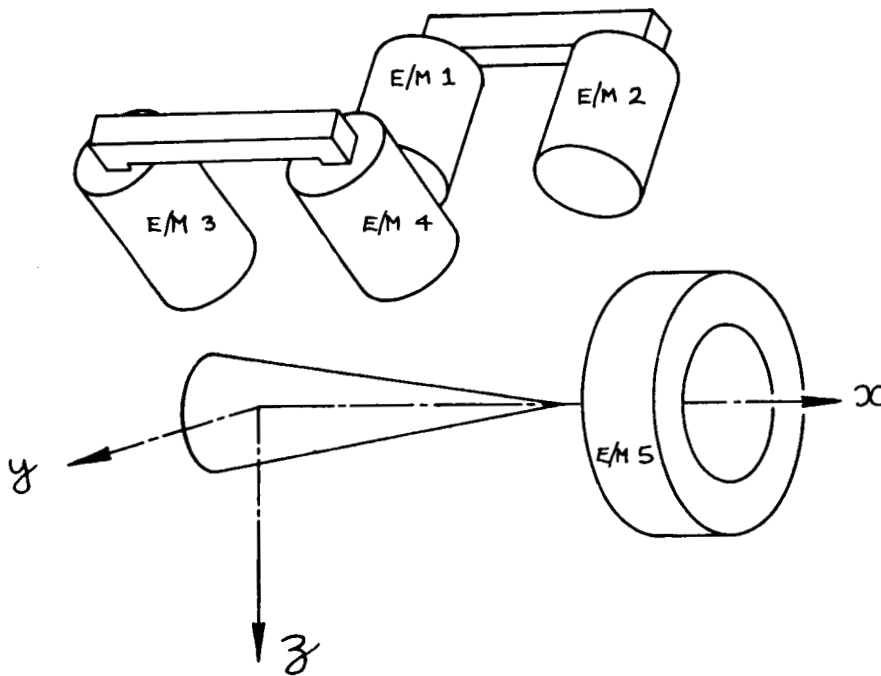
Fig.A.1 Position sensor channel allocation

Current monitors

"TSTAD1" and "TSTAD2" scan all input channels to A/D boards 1 and 2 respectively. The usual channel assignments (see Section 3) are summarized below:

Board	Computer channel	External channel	Function
1	0	1	Drag current
	1	2	Bias 1
	2	3	Bias 2
	3	4	Bias 3
	4	5	Bias 4
	5	6	Control 1
	6	7	Control 2
2	7	8	Control 3
	0	9	Control 4

The numbering sequence for electromagnets is shown below. A/D voltage ranges are liable to be changed between 0 to +10V and -10V to +10V. Consult PDP-11 system documentation for further details.



Electromagnet numbering sequence

Power supplies

"TSTOUT" can be used to set the D/A outputs to any desired steady-state level, thereby commanding steady currents from the

(Control) electromagnet power supplies. Normal channel assignments are shown below :

D/A board	Computer channel	External channel	Function
1	0	1	Control 1
	1	2	Control 2
	2	3	Control 3
	3	4	Control 4
(2)	4 (sequential) (or 0)	5	Drag

The D/A converters will normally be set to the following ranges:

Counts	Range point	Nominal voltage
0	+Full scale	+10 V
4000	0 V	0 V
7777	-Full scale	-10 V

TSTDIO.FOR

```
C*****
C PROGRAM TO SWEEP TEST MSBS POSITION SENSOR DIGITAL CHANNELS
C*****
C
C SCANS DIGITAL INPUT PORTS ASYNCHRONOUSLY, DECODES TO POSITION SENSOR COUNTS
C OUTPUT IS POSITION SENSOR COUNTS (DECIMAL) FOR SENSOR CHANNELS 1-5 IN
C COLUMNS 1-5.
C
C ** WARNING ** ASYNCHRONOUS SCAN TECHNIQUE CAN GIVE INVALID DATA DURING
C INPUT PORT UPDATING.
C
      DIMENSION IDATA(5)
      TYPE 10
10     FORMAT(' DIGITAL INPUT (POSITION SENSOR) SWEEP TEST')
      TYPE 20
20     FORMAT(' NUMBER OF SWEEPS?')
      READ(5,*) NSWEEP
50     DO 30 I=1,NSWEEP
      CALL DIOTST(IDATA)
30     TYPE 40,(IDATA(J),J=1,5)
40     FORMAT(5I8)
      PAUSE
      GO TO 50
      STOP
      END
```

DIOTST.MAC

```
.TITLE DIOTST
;
; Input port assignments as follows:
; Bits          Port      Channel Sensor
; 0-9           A         1      AL
; 10-15 +      A         2      AX
; 0-3           B         3      AU
; 4-13         B         4      FU
; 14-15 +      C         5      FL
; 0-7           C
; 8-15 +      D
; 0-1          D
;
; .GLOBL DIOTST
;
CSRA=164160      ;DIGITAL PORT A STATUS
DBRA=CSRA+2      ;PORT A DATA BUFFER
CSRB=CSRA+4      ;PORT B STATUS
DBRB=CSRA+6      ;ETC.
CSRC=CSRA+10
DBRC=CSRA+12
CSRD=CSRA+14
DBRD=CSRA+16
;
CLS=170420      ;CLOCK NO.2 STATUS REGISTER
CLB=CLS+2       ;CLOCK NO.2 BUFFER REGISTER
;
DIOTST: TST      (R5)+      ;SKIP ARGUMENT COUNT
MOV          (R5)+,DATA    ;SAVE ARRAY ADDRESS FOR DATA RETURN
;
MOV          @#DBRA,R3     ;FETCH DIGITAL INPUT PORT DATA
MOV          @#DBRB,R2
MOV          @#DBRC,R1
MOV          @#DBRD,R0
;
;DISASSEMBLE FOR INDIVIDUAL CHANNELS
;
MOV          R3,R4        ;COPY PORT A
BIC          #176000,R4   ;MASK HIGH BITS
MOV          R4,@DATA     ;STORE
ADD          #2,DATA      ;INCREMENT STORE ADDRESS
;
MOV          R3,R5        ;COPY PORT A
MOV          R2,R4        ;AND PORT B
ASHC        #-10.,R4     ;JUSTIFY DATA
BIC          #176000,R5   ;MASK HIGH BITS
MOV          R5,@DATA     ;STORE
ADD          #2,DATA      ;INCREMENT STORE ADDRESS
;
MOV          R2,R4        ;COPY PORT B
ASH         #-4.,R4      ;JUSTIFY DATA
BIC          #176000,R4   ;MASK HIGH BITS
MOV          R4,@DATA     ;STORE
ADD          #2,DATA      ;INCREMENT STORE ADDRESS
;
MOV          R2,R5        ;COPY PORT B
MOV          R1,R4        ;AND PORT C
ASHC        #-14.,R4     ;JUSTIFY DATA
BIC          #176000,R5   ;MASK HIGH BITS
MOV          R5,@DATA     ;STORE
```

```

;
ADD      #2,DATA          ;INCREMENT STORE ADDRESS
;
MOV      R0,R4           ;COPY PORT D
MOV      R1,R5           ;AND PORT C
ASHC    #-8.,R4         ;JUSTIFY DATA
BIC     #176000,R5      ;MASK HIGH BITS
MOV     R5,@DATA        ;STORE
ADD     #2,DATA         ;INCREMENT STORE ADDRESS
;
RTS     PC              ;GO BACK TO FORTRAN
;
DATA:   .WORD 0         ;DATA ARRAY ADDRESS STORAGE
;
        .END

```


TSTAD1.FOR

```
C*****
C PROGRAM TO SWEEP TEST A/D CONVERTER BOARD 1
C*****
C
C REPETITIVELY SCANS ALL (8) CHANNELS OF A/D BOARD 1 IN OCTAL OR DECIMAL FORM
C REQUESTED CYCLES OVER 1000 CAUSES CONTINUOUS SWEEPING UNTIL <CONTROL>C
C OUTPUT IS DATA FROM CHANNELS 0-7 IN COLUMNS 1-8
C
      DIMENSION IDATA(16)
      TYPE 10
10     FORMAT(' A/D BOARD 1 SWEEP TEST')
      TYPE 20
20     FORMAT('// NUMBER OF CYCLES? (>1000 CONTINUOUS)')
      READ(5,*) ICYCLE
24     TYPE 25
25     FORMAT(' OCTAL(0), OR DECIMAL(1) OUTPUT?')
      READ (5,*) IDUMMY
      IF (IDUMMY) 24,100,200
C
C OCTAL OUTPUT
C
100    JCYCLE=1
30     DO 40 ICHANN=0,7
40     CALL AD1TST(ICHANN, IDATA(ICHANN+1))
      TYPE 50, (IDATA(J), J=1,8)
50     FORMAT(808)
      IF (JCYCLE.GT.ICYCLE) GO TO 60
      JCYCLE=JCYCLE+1
      GO TO 30
60     IF (ICYCLE.GT.1000) GO TO 30
      PAUSE
      GO TO 100
C
C DECIMAL OUTPUT
C
200    JCYCLE=1
130    DO 140 ICHANN=0,7
140    CALL AD1TST(ICHANN, IDATA(ICHANN+1))
      TYPE 150, (IDATA(J), J=1,8)
150    FORMAT(817)
      IF (JCYCLE.GT.ICYCLE) GO TO 160
      JCYCLE=JCYCLE+1
      GO TO 130
160    IF (ICYCLE.GT.1000) GO TO 130
      PAUSE
      GO TO 200
      END
```

AD1TST.MAC

```

      .TITLE AD1TST                                ;PROGRAM TITLE
;
      .GLOBL AD1TST                                ;GLOBAL SYMBOL
;
      ADSR=170430                                  ;DEFINE A/D COMMUNICATIONS
      ADBR=ADSR+2                                  ;PORTS
;
AD1TST: TST      (R5)+                              ;SKIP ARGUMENT COUNT
        MOV      (R5)+,CHANN                        ;GET ADDRESS OF CHANNEL NUMBER (0-15)
        MOV      (R5)+,DATA                        ;SAVE ADDRESS LOCATION FOR DATA RETURN
;
        MOV      @CHANN,R0                          ;FETCH CHANNEL NUMBER
        ASH      #8.,R0                             ;SHIFT NUMBER TO BITS 8-11
        BIS      #1,R0                              ;SET GAIN (1) AND START BIT
        MOV      R0,@#ADSR                          ;SET UP CHANNEL AND GAIN
;
TEST:   TSTB     @#ADSR                             ;IS A/D DONE?
        BPL     TEST                                ;WAIT IF NOT
        TST     R0                                 ;DUMMY WAIT
        MOV     @#ADBR,R0                          ;FETCH DATA
        MOV     R0,@DATA                           ;AND STORE
;
        RTS     PC                                 ;GO BACK TO FORTRAN
;
CHANN:  .WORD   0                                  ;CHANNEL NUMBER
DATA:   .WORD   0                                  ;RETURNED DATA
;
.END
```

TSTAD2.FOR

```
C*****
C PROGRAM TO SWEEP TEST A/D CONVERTER BOARD 2
C*****
C
C REPETITIVELY SCANS ALL (8) CHANNELS OF A/D BOARD 2 IN OCTAL OR DECIMAL FORM
C REQUESTED CYCLES OVER 1000 CAUSES CONTINUOUS SWEEPING UNTIL <CONTROL>C
C OUTPUT IS DATA FROM CHANNELS 0-7 IS COLUMNS 1-8
C
      DIMENSION IDATA(16)
      TYPE 10
10     FORMAT(' A/D BOARD 2 SWEEP TEST')
      TYPE 20
20     FORMAT('/' NUMBER OF CYCLES? (>1000 CONTINUOUS)')
      READ(5,*) ICYCLE
24     TYPE 25
25     FORMAT(' OCTAL(0), OR DECIMAL(1) OUTPUT?')
      READ (5,*) IDUMMY
      IF (IDUMMY) 24,100,200
C
C OCTAL OUTPUT
C
100    JCYCLE=1
30     DO 40 ICHANN=0,7
40     CALL AD2TST(ICHANN, IDATA(ICHANN+1))
      TYPE 50, (IDATA(J), J=1,8)
50     FORMAT(808)
      IF (JCYCLE.GT.ICYCLE) GO TO 60
      JCYCLE=JCYCLE+1
      GO TO 30
60     IF (ICYCLE.GT.1000) GO TO 30
      PAUSE
      GO TO 100
C
C DECIMAL OUTPUT
C
200    JCYCLE=1
130    DO 140 ICHANN=0,7
140    CALL AD2TST(ICHANN, IDATA(ICHANN+1))
      TYPE 150, (IDATA(J), J=1,8)
150    FORMAT(8I7)
      IF (JCYCLE.GT.ICYCLE) GO TO 160
      JCYCLE=JCYCLE+1
      GO TO 130
160    IF (ICYCLE.GT.1000) GO TO 130
      PAUSE
      GO TO 200
      END
```

AD2TST.MAC

```
      .TITLE  AD2TST                ;PROGRAM TITLE
;
      .GLOBL  AD2TST                ;GLOBAL SYMBOL
;
      ADSR=170440                   ;DEFINE A/D COMMUNICATIONS
      ADBR=ADSR+2                   ;PORTS
;
AD2TST: TST      (R5)+               ;SKIP ARGUMENT COUNT
        MOV      (R5)+,CHANN        ;GET ADDRESS OF CHANNEL NUMBER (0-15)
        MOV      (R5)+,DATA        ;SAVE ADDRESS LOCATION FOR DATA RETURN
;
        MOV      @CHANN,R0         ;FETCH CHANNEL NUMBER
        ASH      #8.,R0            ;SHIFT NUMBER TO BITS 8-11
        BIS      #1,R0            ;SET GAIN (1) AND START BIT
        MOV      R0,@ADSR         ;SET UP CHANNEL AND GAIN
;
TEST:   TSTB     @#ADSR            ;IS A/D DONE?
        BPL      TEST            ;WAIT IF NOT
        TST      R0              ;DUMMY WAIT
        MOV      @#ADBR,R0        ;FETCH DATA
        MOV      R0,@DATA        ;AND STORE
;
        RTS      PC              ;GO BACK TO FORTRAN
;
CHANN:  .WORD    0                ;CHANNEL NUMBER
DATA:   .WORD    0                ;RETURNED DATA
;
.END
```

TSTOUT.FOR

```
C*****
C PROGRAM TO TEST D/A CONVERTER FUNCTION
C*****
C
C TAKES (DECIMAL) CHANNEL NUMBER AND (OCTAL OR VOLTAGE) DATA AND SETS
C RELEVANT D/A CHANNEL. PRESENT VALID CHANNEL NUMBERS ARE 0-11, VALID DATA
C IS FROM 0-7777 (OCTAL), OR -10 TO +10 (VOLTS)
C
C           0 = +FULL SCALE (9.9951V)
C           4000 = 0V
C           7777 = -FULL SCALE (-10V)
C
C NEGATIVE CHANNEL NUMBER OR <CONTROL>C ABORTS PROGRAM
C OUTPUT IS OLD (OCTAL) D/A PORT DATA, NEW (OCTAL) DATA, CHANNEL NUMBER
C (DECIMAL) AND NOMINAL D/A OUTPUT VOLTAGE
C
100      TYPE 10
100      FORMAT(' D/A SINGLE CHANNEL TEST'//,' SPECIFY OCTAL(0) OR VOLTS(1)
100      1 INPUT')
100      READ(5,*) IDUMMY
20       TYPE 30
30       FORMAT('/' CHANNEL NUMBER (DECIMAL, 0-11)?')
30       READ(5,*) ICHANN
30       IF(ICHANN.LT.0) GO TO 70
30       IF(ICHANN.GT.11) GO TO 20
30       IF (IDUMMY) 100,110,120
C
C OCTAL INPUT SECTION
C
110      TYPE 40
40       FORMAT(' DATA (OCTAL, 0-7777)?')
40       READ(5,50,ERR=20) IDATA2
50       FORMAT(04)
50       IF (IDATA2.LT.0.OR.IDATA2.GT."7777") GO TO 110
50       CALL OUTTST(ICHANN,IDATA1,IDATA2)
50       EQVOLT=-10.*(IDATA2-"4000)/FLOAT("3777)
50       TYPE 60, IDATA1, IDATA2, ICHANN, EQVOLT
60       FORMAT(' OLD DATA=',06,' NEW DATA=',06,' CHANNEL=',I4,
60       1 ' NOMINAL VOLTS=',F7.3)
60       GO TO 20
C
C VOLTAGE INPUT SECTION
C
120      TYPE 130
130      FORMAT(' DATA (VOLTAGE, -10 TO +10)?')
130      READ(5,*) VOLTS
130      IF (ABS(VOLTS).GT.10) GO TO 120
130      IDATA2=IFIX("4000-"3777*VOLTS*0.1)
130      CALL OUTTST(ICHANN,IDATA1,IDATA2)
130      TYPE 80, IDATA1, IDATA2, ICHANN, VOLTS
80       FORMAT(' OLD DATA=',06,' NEW DATA=',06,' CHANNEL=',I4,
80       1 ' NOMINAL VOLTS=',F7.3)
80       GO TO 20
C
C DONE
C
70       STOP
70       END
```

OUTTST.MAC

```

        .TITLE  OUTTST                ;PROGRAM TITLE
;
        .GLOBL  OUTTST                ;DECLARE GLOBAL SYMBOL
;
        DAONE=170450                  ;ADDRESS OF FIRST D/A CONVERTER
;
OUTTST: TST      (R5)+                 ;SAVE DATA ACROSS FORTRAN CALL
        MOV      (R5)+,CHANN           ;CHANNEL NUMBER
        MOV      (R5)+,DATA1          ;FOR OLD DATA
        MOV      (R5)+,DATA2          ;NEW DATA
;
        MOV      @CHANN,R0             ;FETCH CHANNEL NUMBER
        ASL      R0                    ;CONVERT TO CHANNEL ADDRESS INCREMENTS
        MOV      #DAONE,R1            ;FETCH BASE ADDRESS
        ADD      R1,R0                 ;ADD BASE
;
        MOV      (R0),@DATA1           ;FETCH THE CURRENT D/A BUFFER CONTENTS
        MOV      @DATA2,(R0)          ;WRITE IN THE NEW VALUE
;
        RTS      PC                    ;GO BACK TO FORTRAN
;
CHANN:  .WORD    0                     ;CHANNEL NUMBER
DATA1:  .WORD    0                     ;D/A BUFFER DATA BEFORE WRITE
DATA2:  .WORD    0                     ;NEW D/A BUFFER DATA
;
        .END
```

REFERENCES

1. Goodyer, M.J.: The Magnetic Suspension of Wind Tunnel Models for Dynamic Testing. University of Southampton, Department of Aeronautics and Astronautics PhD Thesis, April 1968.
2. Luh, P.B., Covert, E.E., Whitaker, H.P. Haldeman, C.W.: Application of Digital Control to a Magnetic Suspension and Balance System. NASA CR-145316, January 1978.
3. Fortescue, P.W., Bouchalis, C.: Digital Controllers for the Vertical Channels of a Magnetic Suspension System. NASA CR-165684, May 1981.
4. Britcher, C.P.: The Southampton University Magnetic Suspension and Balance System - A Partial User Guide. Southampton University Department of Aeronautics and Astronautics AASU Tech. Memo 83/8, April 1984.
5. Britcher, C.P., Fortescue, P.W., Allcock, G.A., Goodyer, M.J.: Investigation of Design Philosophies and Features Applicable to Large Magnetic Suspension and Balance Systems. NASA CR-162433, November 1979.
6. Britcher, C.P.: Some Aspects of Wind Tunnel Magnetic Suspension and Balance Systems with Special Application at Large Physical Scales. NASA CR-172154, September 1983.
7. LaFleur, S.S.: Advanced Optical Position Sensors for Magnetically Suspended Wind Tunnel Models. 11th ICIASF Conference, Stanford, August 1985.

1. Report No. NASA CR-178210		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle User Guide for the Digital Control System of the NASA/ Langley Research Center's 13-inch Magnetic Suspension and Balance System				5. Report Date March 1987	
				6. Performing Organization Code	
7. Author(s) Colin P. Britcher				8. Performing Organization Report No.	
				10. Work Unit No.	
9. Performing Organization Name and Address Old Dominion University Department of Mechanical Engineering and Mechanics Norfolk, VA 23508				11. Contract or Grant No. NAS1-17993, Task 24	
				13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546				14. Sponsoring Agency Code 505-61-01-02	
15. Supplementary Notes Technical Monitor: R. P. Boyden, Langley Research Center, Hampton, Virginia. Final Report for the period October 23, 1985 to September 30, 1986.					
16. Abstract The technical background to the development of the digital control system of the NASA/Langley Research Center's 13-inch Magnetic Suspension and Balance System (MSBS) is reviewed. The implementation of traditional MSBS control algorithms in digital form is examined. Extensive details of the 13-inch MSBS digital controller and related hardware are given, together with introductory instructions for system operators. Full listings of software are included in Appendices.					
17. Key Words (Suggested by Author(s)) Magnetic Suspension Digital Control Wind Tunnels			18. Distribution Statement Unclassified - Unlimited Subject Category - 09		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 79	22. Price A05