

AD-A075 060

UNITED TECHNOLOGIES RESEARCH CENTER EAST HARTFORD CONN F/6 9/2
MICROPROCESSOR REQUIREMENTS FOR IMPLEMENTING MODERN CONTROL LOG--ETC(U)
SEP 79 F A FARRAR, J R KRODEL F49620-79-C-0078

UNCLASSIFIED

UTRC/R79-944590-1

AFOSR-TR-79-1044

NL

| OF |

AD
A075060



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 79 - 1044	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MICROPROCESSOR REQUIREMENTS FOR IMPLEMENTING MODERN CONTROL LOGIC	5. TYPE OF REPORT & PERIOD COVERED Interim	
7. AUTHOR(s) F.A. Farrar and James R. Krodel	6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS United Technologies Research Center Controls & Operations Analysis Group East Hartford, CT 06108	8. CONTRACT OR GRANT NUMBER(s) F49620-78-C-0017	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A1	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	12. REPORT DATE September 14, 1979	
	13. NUMBER OF PAGES 24	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The objective of this research investigation is to develop and evaluate analytical procedures for establishing microprocessor requirements for implementing modern control logic. Key issues in microprocessor implementation of modern control logic include: (1) accuracy, (2) computational capability, and (3) memory requirements. These requirements must be established for the interface as well as for the microprocessor control code.		

DDC
OCT 15 1979
RECEIVED

UNITED TECHNOLOGIES RESEARCH CENTER

East Hartford, Connecticut 06108

Report No.: R79-944590-1
Date: September 14, 1979
Prepared by: Florence A. Farrar
James R. Krodel

Director
Mathematical and Information Sciences
Air Force Office of Scientific Research
United States Air Force
Building 410
Bolling Air Force Base
Washington, D.C. 20332

Attention: Major Charles L. Nefzger
AFOSR/NM

Subject: Microprocessor Requirements for Implementing Modern
Control Logic, Contract F49620-79-C-0078, Technical
Progress Report No. 1 for the period March 1, 1979
through August 31, 1979.

Reference: Farrar, F. A. and R. S. Eidens: Microprocessor
Requirements for Implementing Modern Control Logic.
United Technologies Research Center Report R79-944258-2,
Final Technical Report prepared under Air Force Office
of Scientific Research Contract F49620-78-C-0017,
March 1979.

Gentlemen:

1a. The objective of this research investigation is to develop
and evaluate analytical procedures for establishing microprocessor
requirements for implementing modern control logic. Key issues in
microprocessor implementation of modern control logic include (1) accuracy,
(2) computational capability, and (3) memory requirements. These require-
ments must be established for the interface as well as for the micro-
processor control code.

1b. A Phase I first-year study directed toward establishing
microprocessor requirements for control of linear systems was performed
at United Technologies Research Center (UTRC) from 1 February 1978 to
31 January 1979 under Contract F49620-78-C-0017 with the Air Force Office



of Scientific Research (AFOSR). This Phase I program resulted in successful development of analytical procedures for establishing microprocessor requirements for multivariable feedback control of linear stochastic dynamic systems. The developed procedures were evaluated and illustrated by application to (1) a second-order system and (2) a linearized fifth-order F100 turbofan engine model. These results are documented in the referenced report. Based on these results, the subject contract was awarded to UTRC for a Phase II second-year program. This second-year study is directed toward (1) validating Phase I results by demonstrating microprocessor implementation of linear quadratic Gaussian (LQG) logic for control of linear systems and (2) extending the analysis of microprocessor requirements for control of nonlinear systems.

2. During this report period the UTRC effort has been directed toward validating Phase I results by demonstrating LQG control on an Intel 8080 microprocessor for a second-order continuous system. The second-order system is the same system used in the Phase I study. Accuracy, computational capability, and memory requirements of the microprocessor demonstration system will be compared with requirements predicted using the analytical procedures developed in the Phase I study. In addition, a paper presenting the Phase I study results was submitted and accepted for publication in the IEEE Transactions on Automatic Control and for presentation at the 18th IEEE Conference on Decision and Control (CDC). Approval to submit the paper was obtained from Major Charles L. Nefzger, the AFOSR contract monitor. A copy of the cover letter and paper submitted for publication in the Transactions is enclosed for Major Nefzger with this report.

3a. To demonstrate LQG control on an Intel 8080 microprocessor for the second-order continuous system (analog model), the system shown in Fig. 1 is being employed. The system consists of the analog model, an Intel 8080 microprocessor, A/D and D/A converters, and a strip chart recorder. The continuous system dynamics (analog model) are represented by

$$\begin{aligned} \ddot{y} + 4\dot{y} + 2y &= u_p; y(0) = 0.5 \\ \dot{y}(0) &= 0.5 \\ z_p &= y + \eta_p \end{aligned} \quad (1)$$

where y , u_p , z_p , and η_p represent the system output, input, measurement, and measurement noise, respectively. The dot notation denotes differentiation with respect to time. The absolute maximum values are given by

$$\begin{aligned} |y| &= 1.0 \\ |\dot{y}| &= 1.0 \\ |\ddot{y}| &= 4.0 \\ |u|_{\max} &= 2.0 \end{aligned} \quad (2)$$

The analog system shown in Fig. 1 is scaled so that 10 volts corresponds to the maximum value.

3b. For LQG control design, continuous system dynamics are represented in state-space notation. For digital control implementation, system dynamics are normalized so that the numbers in the control computations range between -1.0 and +1.0. Normalized continuous system dynamics are described by

$$\begin{aligned}\dot{x} &= Ax + Bu + \xi \\ y &= Cx + Du \\ z &= Ex + n\end{aligned}\tag{3}$$

where the vectors $x(n \times 1)$, $u(m \times 1)$, $y(p \times 1)$ and $z(l \times 1)$ represent the normalized system states, inputs, outputs, and measurements, respectively. The random process vectors ξ and n represent white zero-mean Gaussian n -dimensional process and l -dimensional measurement noise, respectively. The second-order system matrices (A , B , C , D , and E) are shown in Table I. The companion form is shown for ease in relating the state equations (Eq. (3)) to the system input-output equation (Eq. (1)). (In the referenced report, the system matrices were displayed in standard form.)

3c. Normalized discrete control dynamics implemented on the micro-processor are given by

$$\begin{aligned}\hat{w}(k+1) &= \phi_D \hat{w}(k) + H_D z(k+1) \\ u(k+1) &= G_D \hat{w}(k+1)\end{aligned}\tag{4}$$

where

$$\begin{aligned}\hat{w} &= T^{-1} \hat{x} \\ \phi_D &= (I - T^{-1} H E T \Delta t) (I + \phi) \\ \phi &= T^{-1} F T \Delta t + \frac{(T^{-1} F T)^2 \Delta t^2}{2!} + \frac{(T^{-1} F T)^3 \Delta t^3}{3!} + \dots \\ H_D &= T^{-1} H \Delta t \\ G_D &= G T\end{aligned}\tag{5}$$

The vector $\hat{w}$ represents estimated transformed state variables. The matrices G and H denote the deterministic control gains and filter gains, respectively. The gain matrices for the second-order system are shown in Table I. The transformation matrix T is an identity matrix.

4a. The software to implement Eq. (4) on a microprocessor consists of three matrix/vector multiplications ($\Phi_D \hat{W}$, $H_D Z$, and $G_D \hat{W}$) and one vector addition ($(\Phi_D \hat{W}) + (H_D Z)$). In addition, software for the LQG control system includes a routine to save past state estimates and an interrupt service routine. The interrupt service routine (1) performs a timing check to assure that all control computations are completed within the sample time and (2) reads in measurement data and updates the system inputs.

4b. The overall block diagram of the control software described above is shown in Fig. 2. The block diagram of the matrix/vector multiplication code is displayed in Fig. 3. Figure 3 indicates that several minor changes (e.g., the order in which pointers are initialized and updated) were made in the matrix/vector multiplication block diagram presented in the referenced report. These changes result in more efficient microprocessor implementation. Note that the matrix/vector multiplication algorithm is not changed but rather the way the algorithm is implemented has been slightly modified. Block diagrams of the vector addition code, the store state estimates, and the interrupt service routine are shown in Fig. 4.

4c. Computation times as a function of computer cycles to execute the blocks of code shown in Figs. 3 and 4 are displayed in Table 2. The cycle time to execute a block of code varies with the microprocessor used. Note also that the computation time is defined in number of cycles. Once the number of cycles for the code on a given microprocessor has been defined conversion to seconds for different clock frequencies is easily done ($t(\text{sec}) = t(\text{cycles}/\text{clock frequency (hertz)})$).

5a. To compute the cycles for a given microprocessor, code must be written to execute the functions shown in the block diagrams of Figs. 3 and 4. Control code for the Intel 8080 microprocessor is shown in Fig. 5. The code consists of software to execute the control equations (Eq. (4)) and interface software. A software multiplication subroutine is used. Improvements were made in the preliminary matrix/vector multiplication code presented in the referenced report. These improvements result in reduced execution time. The code changes include (1) more efficient use of the registers in the multiplication algorithm (11.5% reduction in cycle time) and (2) more efficient memory (data array) accessing as well as more efficient use of the registers in the matrix/vector multiplication algorithm (reduction in cycle time dependent on system order, e.g., a 24% reduction in the matrix/vector multiplication control algorithm is obtained for a 2×2 matrix times a 2×1 vector). In addition, the interface software was added to the preliminary code. The code is also completely documented. The flow chart for the code is shown in Fig. 2.

5b. Table III shows times associated with executing the blocks of code in Figs. 3 and 4. Using the Phase I procedure to determine the computation time for the second-order system (Tables II and III) indicates that 6252 cycles are required for one pass through the code. For a 2 MHz clock (on-board ROM memory) the predicted computation time is equal to 3.13 ms; for a 1.47 MHz clock (off-board RAM memory) the predicted computation time is 4.25 ms. For convenience off-board RAM memory will be employed in this study.

6. Intel 8080 memory requirements as a function of system state, input and output orders are shown in Table IV. These requirements include the LQG control logic code as well as the interface logic code (interrupt service routine). Phase I procedures indicate that the total memory required for the second-order system is 490 bytes (15 bytes of RAM and 475 bytes of ROM).

7. Phase I study results indicate that the Intel 8080 with software multiply can be used to implement the LQG control law for the second-order system. The requirements for the second-order controller using the standard form are analytically computed to be (1) an 8-bit word length accuracy, (2) a minimum sample time (based on computations) of 4.25 ms and a maximum sample time (based on controller poles) of 700 ms, and (3) 490 words of memory (475 words of PROM and 15 words of RAM). These requirements include interface as well as microprocessor requirements.

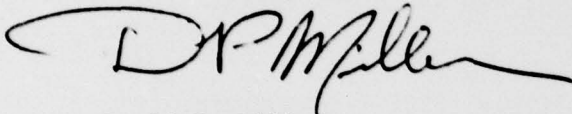
8. The system shown in Fig. 1 has been constructed. The code has been implemented on the Intel 8080 microprocessor. The memory used agrees with the analytically predicted requirements. The actual computation time/sampling interval has been computed using an oscilloscope. The minimum sample time (based on actual code) was 4.70 ms. This actual computation time is approximately 10% higher than the analytically predicted code. This difference is within the $\pm 20\%$ variation that is expected. Figure 6 compares the actual system response with the Phase-I predicted response. Figure 6 shows that, as predicted in the Phase I study, 8-bit accuracy is sufficient for implementing the second-order controller.

9. In the next report period the second-order and fifth-order controller analysis will be completed. Procedures for determining microprocessor requirements will be extended to nonlinear systems. In addition, results obtained in the Phase I study will be prepared for presentation at the 18th IEEE CDC.

10. The Principal Investigator, Mrs. Florence Farrar, is starting full-time studies for her Ph.D. at the University of Connecticut (UCONN) in September. Dr. Robert Guile, who received his Ph.D. from UCONN in 1970, has assumed the responsibilities of Principal Investigator effective September 20, 1979. Dr. Guile has most recently been engaged in research in the area of peak seeking adaptive systems. He has previously been involved with analysis and design of digital electronic control systems. He has 10 years of engineering experience, including 3 years at UTRC. Mr. James Krodel is continuing as Co-investigator.

Very truly yours,

UNITED TECHNOLOGIES CORPORATION
Research Center



David P. Miller
Assistant Manager,
Engineering Operations

DPM/dip
Enclosure

TABLE I

SECOND-ORDER MODEL DYNAMICS

Matrix	Matrix Elements	
A	0.0 -2.0	1.0 -4.0
B	0.0 2.0	
C	1.0	0.0
D	0	
E	1.0	0.0
G	-0.871	-0.207
H	1.130 -0.360	

TABLE II
COMPUTATION TIME

Function	Matrix/Vector Multiplications	$n^2 + n\ell + nm$
	Matrix/Vector Addition	$n(n-1) + n(\ell-1) + m(n-1)$
	Vector Additions	n
Time* (Cycles)	Matrix/Vector Multiplication	$(\bar{A} * \bar{L}_{a1}) (N^2 + n\ell + mn - 2n - m) + (\bar{L}_{a2} + \bar{L}_{a3})$ $(2n + m) - 3\bar{L}_{a3} + (\bar{m} + \bar{L}_{m1}) (n^2 + n\ell + nm)$ $+ \bar{L}_{m2} (2n + \ell) + 3 \bar{L}_{m0}$
	Vector Addition State Store	$(\bar{V}_{a1} + \bar{V}_{a2})n - \bar{V}_{a2} + \bar{V}_{a0}$ $(\bar{V}_{d1} + \bar{V}_{d2})n - \bar{V}_{d2} + \bar{V}_{d0}$
	Interrupt Service	$\overline{TC} + (\overline{AD})\ell + (\overline{DA})m$

* See Figs. 3 and 4 for definitions of time notation (e.g., $\bar{A}$ defined on Fig. 3).

TABLE III

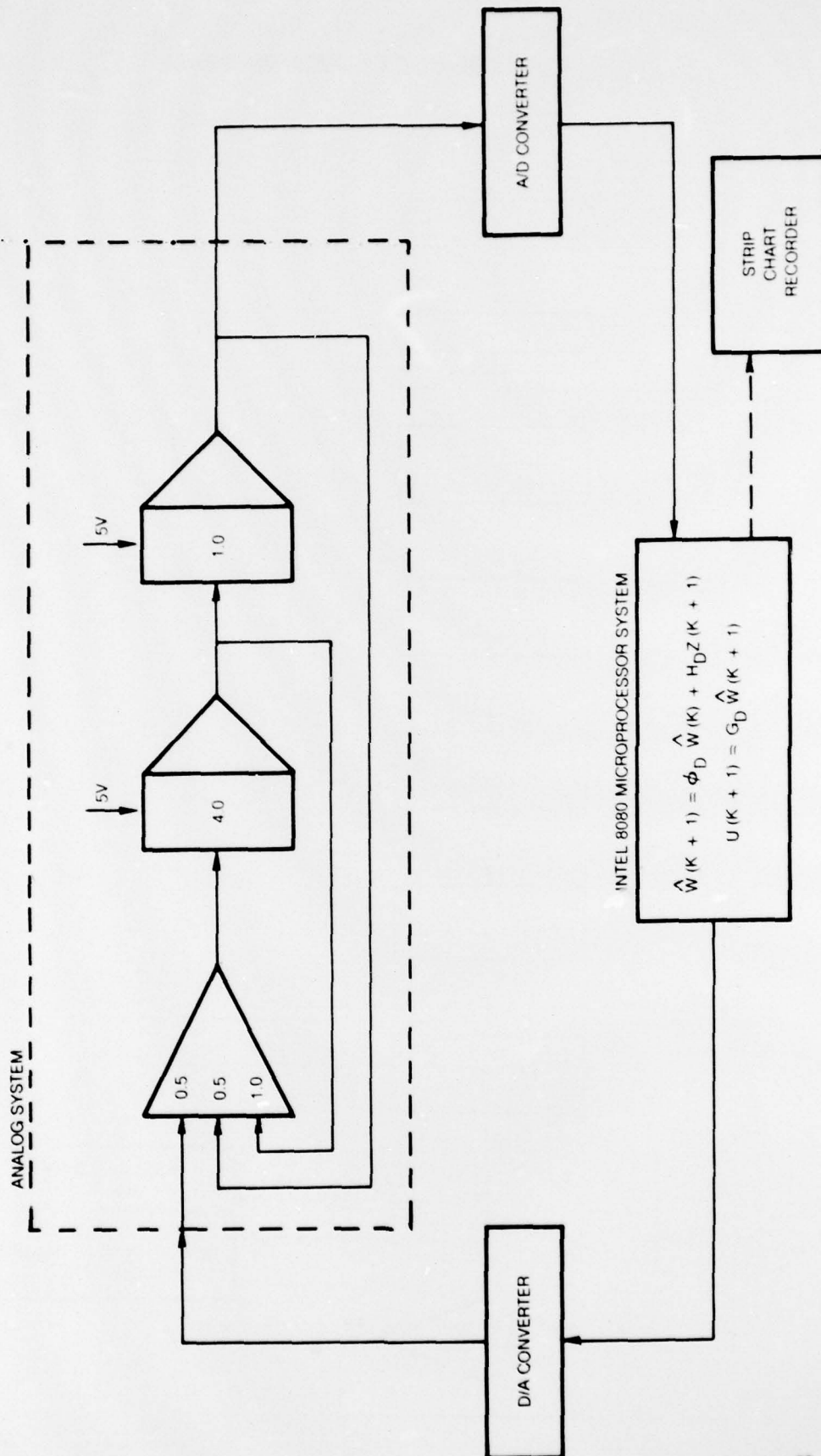
Intel 8080 Execution Times

Figure	Function	Cycle Time
3	$\bar{M}$	433
	L_{m0}	54
	L_{m1}	103
	L_{m2}	55
	$\bar{A}$	17
	L_{a1}	15
	L_{a2}	71
	L_{a3}	79
4	$\bar{V}_{a0}$	60
	$\bar{V}_{a1}$	122
	$\bar{V}_{a2}$	56
	$\bar{U}_{d0}$	72
	$\bar{U}_{d1}$	47
	$\bar{U}_{d2}$	56
	$\overline{TC}$	128
	$\overline{AD}$	174
	$\overline{DA}$	34

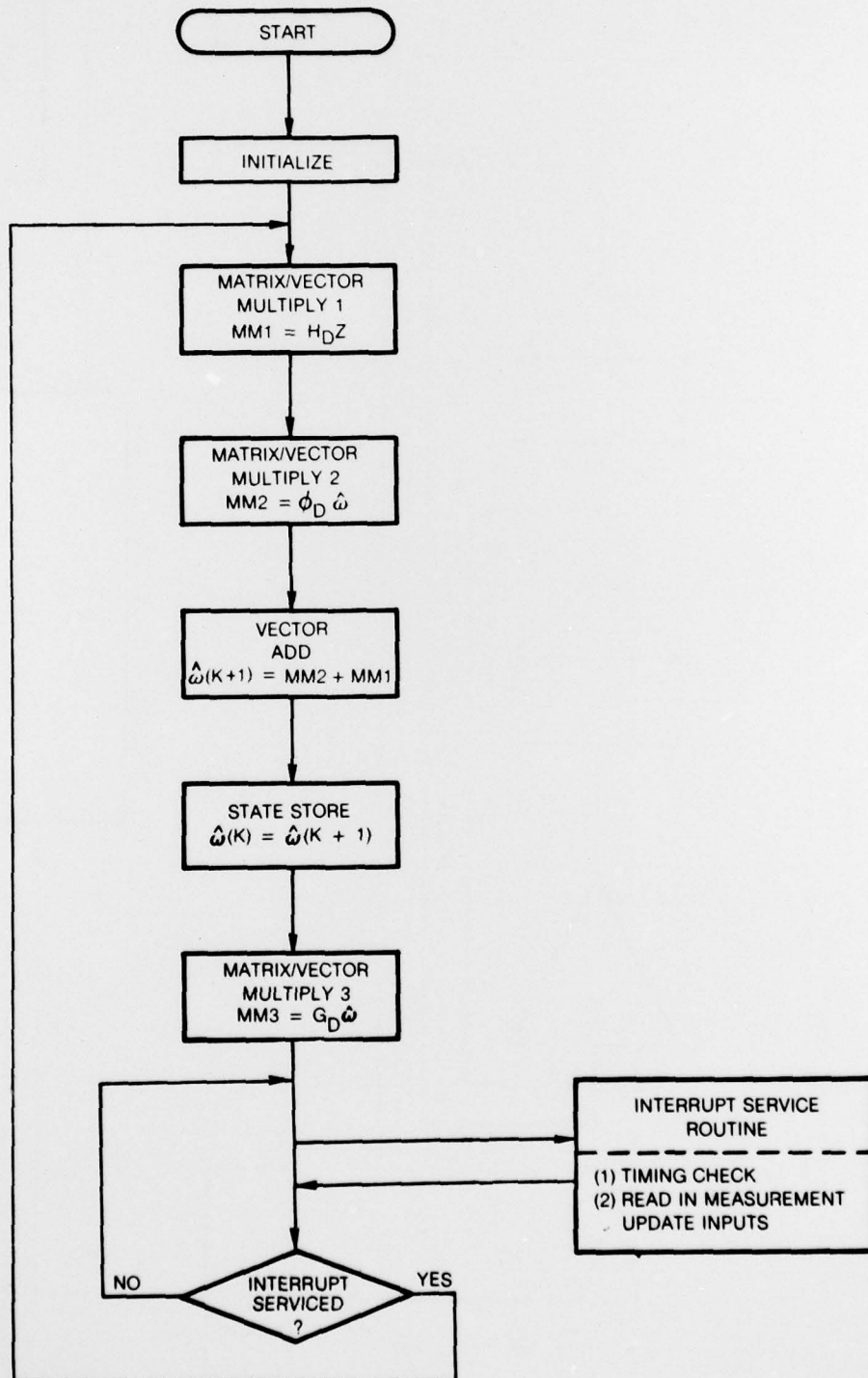
TABLE IV
MEMORY REQUIREMENTS

Variable		Memory Type	Memory (BYTES)
Past state estimate ($\hat{w}(k)$)		RAM	n
Current state estimate ($\hat{w}(k+1)$)		RAM	n
Measurement ($z(k+1)$)		RAM	l
Control ($u(k+1)$)		RAM	m
System matrix (Φ_D)		PROM	n^2
Kalman gain matrix (H_D)		PROM	nl
Control gain matrix (G_D)		PROM	mn
Temporary Storage		RAM	$n^2 + 3nl + nm + ml$
Computer Code	Main Control Logic	PROM	393
	Interrupt Service Routine	PROM	74
TOTAL		RAM	$n^2 + 3nl + nm + ml + m + l$
		PROM	$n^2 + nl + mn + 467$

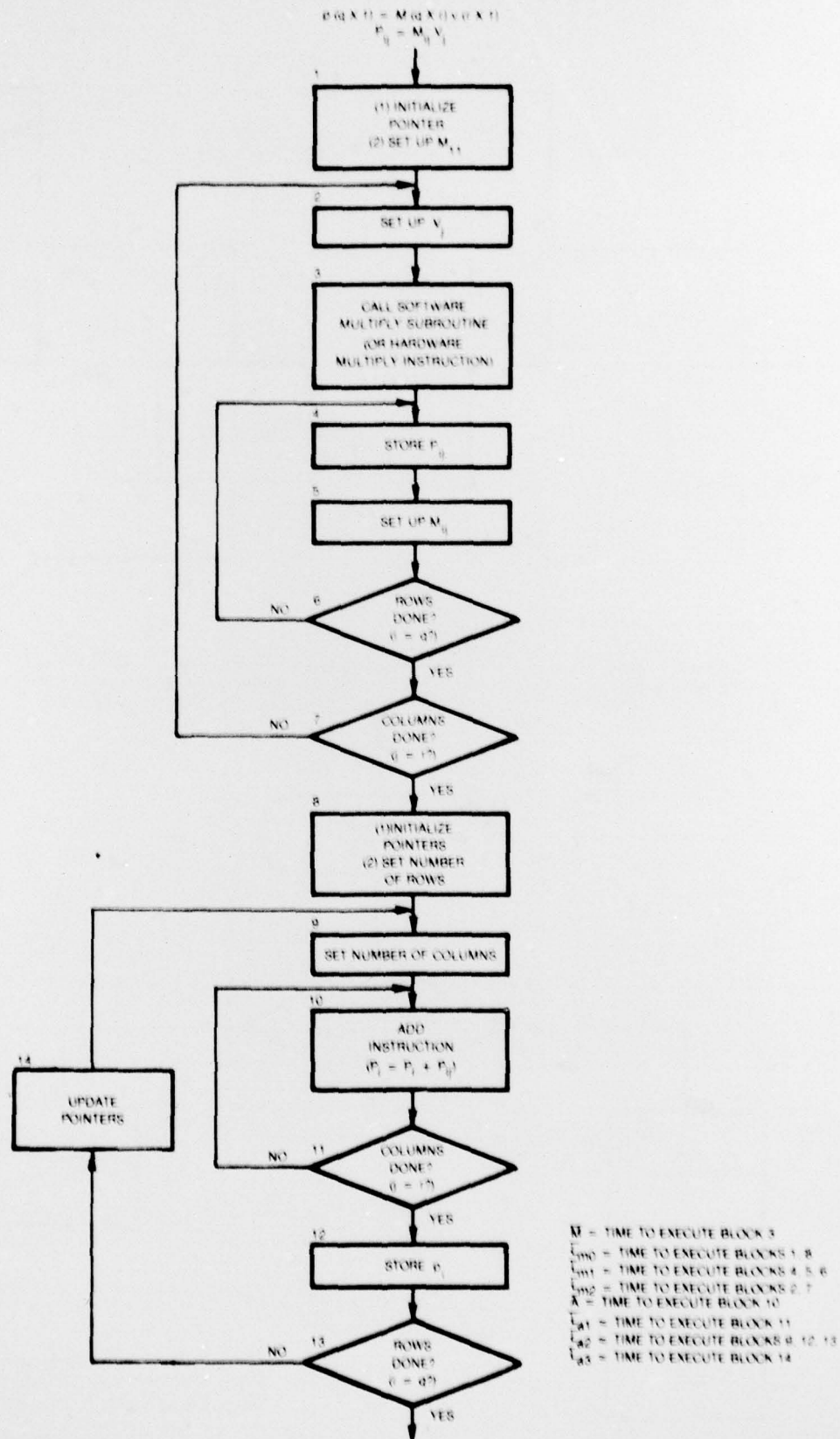
SYSTEM STRUCTURE FOR DEMONSTRATING INTEL 8080 MICROPROCESSOR CONTROL



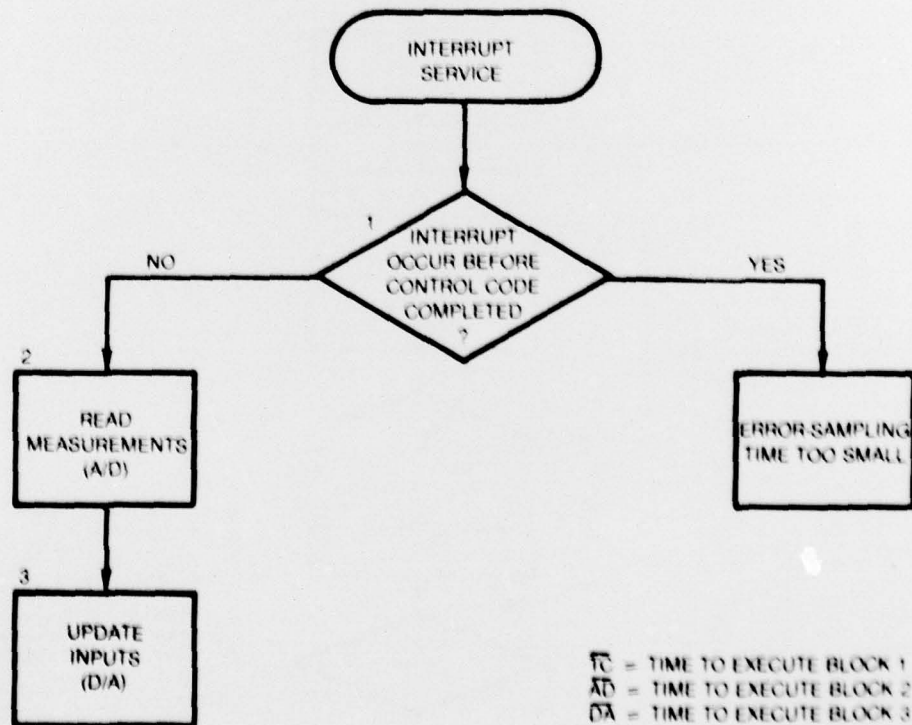
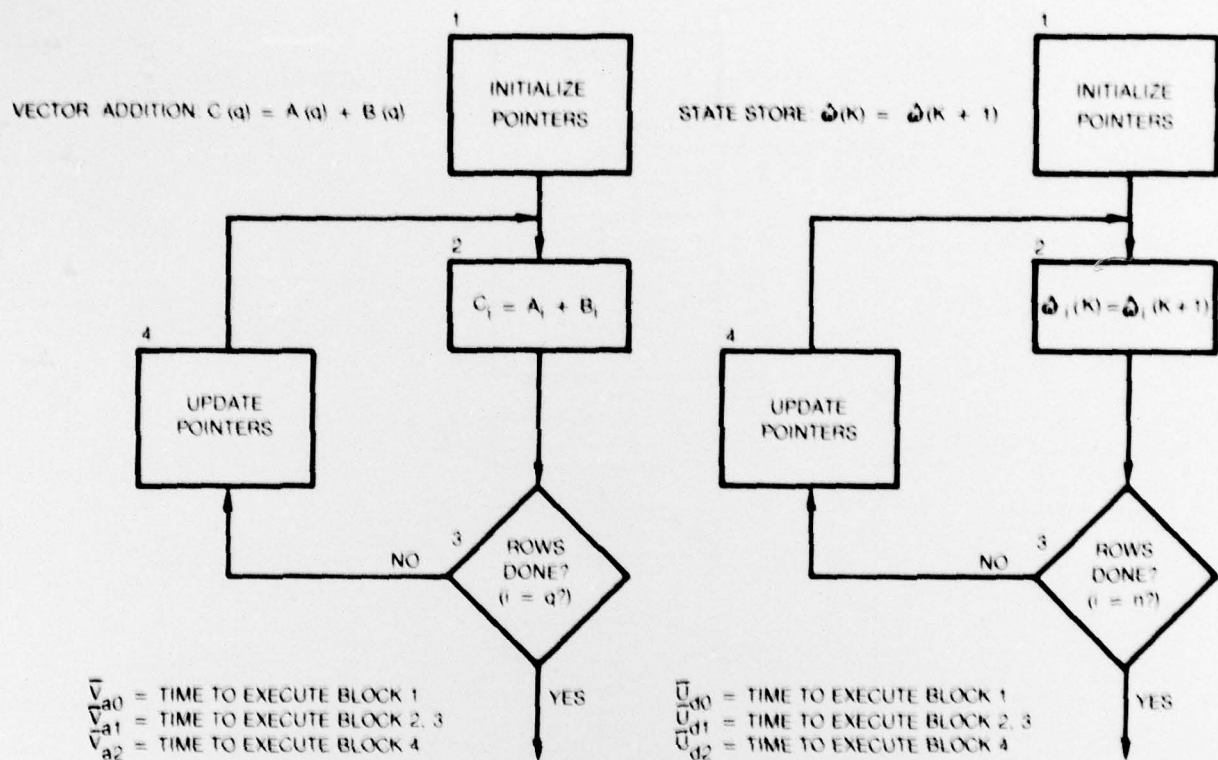
CONTROL CODE BLOCK DIAGRAM



BLOCK DIAGRAM FOR MATRIX/VECTOR MULTIPLICATION



BLOCK DIAGRAM FOR VECTOR ADDITION, STATE STORE AND INTERRUPT SERVICE



INTEL 8080 SOFTWARE FOR LQG CONTROLLER

```

1:
2: 8080 MACRO ASSEMBLER, VER 2.0  ERRORS - 0 PAGE 1
3:
4:
5:
6:
7: ;
8: ; KALMAN FILTER/CONTROLLER
9:
10: ; PROJECT: MICROPROCESSOR IMPLEMENTATION OF MODERN C
11: ; PROGRAMMR: J. KRODEL, DIGITAL COMPUTER LAB
12: ; DATE: 17-AUG-79
13: ; VERSION: 00.00
14: ; REVISION: 00.00
15:
16: ; FUNCTIONAL DESCRIPTION:
17: ; THIS PROGRAM CONTROLS A 2ND
18: ; ORDER SYSTEM, USING MODERN CONTROL
19: ; METHODS. THE BASIC EQUATIONS FOLLOW
20:
21: ;
22: ; 
$$U_{K+1} = PHIDU_K + MDZ_K$$

23: ;
24: ; 
$$U_K = U_{K+1}$$

25: ;
26: ; 
$$U_{K+1} = GDZU_{K+1}$$

27: ;
28: ;
29: ; WHERE:
30: ;
31: ; Z - SYSTEM MEASUREMENT VECTOR
32: ; K
33: ;
34: ; MD - KALMAN FILTER GAIN MATRIX
35: ;
36: ;
37: ; U - PAST STATE ESTIMATE VECTOR
38: ; K
39: ;
40: ; PHID - CLOSED LOOP SYSTEM MATRIX
41: ;
42: ;
43: ; U - NEXT STATE ESTIMATE VECTOR
44: ; K+1
45: ;
46: ; GD - CLOSED LOOP FEEDBACK GAIN MATRIX
47: ;
48: ; U - SYSTEM INPUT VECTOR
49: ; K+1
50: ;
51: ;
52: ; FOR THE SECOND ORDER SYSTEM:
53: ;
54: ; Z - 1X1 VECTOR
55: ; MD - 2X1 MATRIX
56: ; U - 2X1 VECTOR
57: ; PHID - 2X2 MATRIX
58: ; GD - 1X2 MATRIX
59: ; U - 1X1 VECTOR
60: ;
61: ; REVISION HISTORY:
62: ;
63: ;
64: ; SYSTEM EQUATES:
65: ;
66: ;
67: ;
68: ;
69: 0001 US1 EQU 1 ; 8 OF ELEMENTS IN VECTOR
70: 0002 NS1 EQU 2 ; 8 ROWS IN MATRIX
71: 5000 DMA1 EQU 5000H ; DATA MATRIX START ADDR 81
72: 5100 PMMA1 EQU 5100H ; PARTIAL MATRIX MPV START ADDR
73: 5200 MMMA1 EQU 5200H ; MATRIX MPV RESULT START ADDR
74:
75: ;

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONTINUED)

```

76: 0002      US2 EQU 2      ; 8 OF ELEMENTS IN VECTOR
77: 0002      RS2 EQU 2      ; 8 ROWS IN MATRIX
78: 5300      DMA2 EQU 5300H  ; DATA MATRIX START ADDR 82
79: 5400      PMMA2 EQU 5400H ; PARTIAL MATRIX MPV START ADDR
80: 5500      MMRA2 EQU 5500H ; MATRIX MPV RESULT START ADDR
81:
82:
83: 0002      US3 EQU 2      ; 8 OF ELEMENTS IN VECTOR
84: 0001      RS3 EQU 1      ; 8 ROWS IN MATRIX
85: 5600      DMA3 EQU 5600H  ; DATA MATRIX START ADDR 83
86: 5700      PMMA3 EQU 5700H ; PARTIAL MATRIX MPV START ADDR
87: 5800      MMRA3 EQU 5800H ; MATRIX MPV RESULT START ADDR
88:
89:
90:
91: 5800      UOUT EQU 5800H   ; OUTPUT VAR LOCATION
92: 5300      STRIPC EQU 5300H ; STRIP CHART VAR LOCATION
93:
94:
95: 3C3D      ORG 3C3DH
96: 3C3D      JMP INTR        ; SET UP INTERRUPT SERV ROUT.
97:
98:
99: 4000      ORG 4000H
100:
101: ; INITIALIZATION
102:
103: 4000      F3              ; DISABLE INTERRUPTS
104: 4001      310040        LXI SP,4000H ; SET UP STACK POINTER
105:
106:
107:
108: ; PROGRAM START:
109:
110: ; MULTIPLY MATRIX MD (KALMAN FILTER GAIN MATRIX) WITH
111: ; VECTOR Z (SYSTEM MEASUREMENT VECTOR).
112: ; MD IS 2 ROWS X 1 COL
113: ; Z IS 1 ROW X 1 COL
114:
115: 4004      STR1:
116: 4004      3E01          MVI A,US1      ; ESTABLISH VECTOR SIZE
117: 4006      110051        LXI D,PMMA1    ; PARTIAL MATRIX MPV START ADDR
118: 4009      210050        LXI H,DMA1     ; DATA MATRIX START ADDR
119:
120: 400C      COLN1:
121: 400C      F5           PUSH PSW        ; START NEXT COLUMN MPV, SAVE
122: 400D      0602        MVI B,RS1       ; ESTABLISH MATRIX ROW SIZE
123: 400F      4E         MOV C,M         ; GET OPERAND 81
124: 4010      23         INX H           ; POINT TO OPERAND 82
125:
126: 4011      ROWN1:
127: 4011      C5         PUSH B          ; MPV THE ELEMENTS IN EACH ROW
128: 4012      E5         PUSH H          ; SAVE B,C,M & L
129: 4013      66         MOV M,M,M      ; GET OPERAND 82
130: 4014      CD4F41     CALL MULT       ; 8 BIT SIGNED
131: 4017      7C         MOV A,M        ; PREPARE TO STORE
132: 4018      12         STAX D          ; SAVE PARTIAL MATRIX MPV
133: 4019      13         INX D          ; ADJUST SAVE POINTER
134: 401A      E1         POP M          ; RESTORE M & L
135: 401B      23         INX H          ; POINT TO NEXT OPERAND
136: 401C      C1         POP B          ; RESTORE VECTOR COUNT (B)
137:
138: 401D      05         DCR B           ; RESTORE OPERAND 81 (C)
139: 401E      C21140     JNZ ROWN1      ; COLUMN ALL DONE ?
140: 4021      F1         POP PSW        ; NO
141: 4022      3D         DCR A          ; YES, RESTORE VECT SIZE
142: 4023      C20C40     JNZ COLN1     ; ALL MULTIPLIES COMPLETE ?
143:
144: 4026      0602        MVI B,RS1     ; YES, SUM PARTIALS TO COMPLET
145: 4028      110052      LXI D,MMRA1    ; REESTABLISH MATRIX ROW SIZE
146: 402B      210051      LXI H,PMMA1    ; MATRIX MPV RESULT START ADDR
147:
148: 402E      SUMA1:
149: 402E      0E01        MVI C,US1     ; PARTIAL MAT MPV START ADDR
150: 4030      AF         XRA A          ; REINITIALIZE VECTOR SIZE

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER

(CONTINUED)

```

151: 4031 D5          PUSH D          ; OBTAIN POINTER OFFSET IN D&E
152: 4032 1600       MUI D,0         ; CLEAR UPPER PORTION OF D&E
153: 4034 1E02       MUI E,RS1      ; SET OFFSET, MATRIX ROW SIZE
154:
155: 4036             ; SUMB1:
156: 4036 86          ADD A           ; A = A + HL
157: 4037 19          DAD D           ; HL = HL + DE (DE = OFFSET)
158: 4038 0D          DCR C           ; ALL TERMS SUMMED ?
159: 4039 C23640     JNZ SUMB1       ; NO
160: 403C D1          POP D           ; YES, RESTORE RESULT ADDR
161: 403D 12          STAX D          ; STORE RESULT
162: 403E 05          DCR B           ; MATRIX MPY COMPLETE ?
163: 403F CA5240     JZ DONE1        ; YES
164: 4042 13          INX D           ; NO, POINT TO NEXT RESULT ADDR
165: 4043 D5          PUSH D          ; ADJUST TERM POINTER TO LAST
166: 4044 210051     LXI H,PMMA1     ; GET BASE ADDR
167: 4047 3E02       MUI A,RS1      ; GET MATRIX ROW SIZE
168: 4049 90          SUB B           ;
169: 404A 5F          MOV E,A         ; GET OFFSET
170: 404B 1600       MUI D,0         ; CLEAR UPPER PORTION OF DE
171: 404D 19          DAD D           ; HL = PMMA + US - REG B
172: 404E D1          POP D           ;
173: 404F C32E40     JMP SUMA1        ;
174:
175: 4052             ; DONE1:
176:
177:
178:             ; MULTIPLY MATRIX PHID (CLOSED LOOP SYSTEM MATRIX) UI
179:             ; VECTOR U (PAST STATE VECTOR).
180:             ; PHID IS 2 ROWS X 2 COLS
181:             ; U IS 2 ROWS X 1 COL
182:
183: 4052 3E02       MUI A,US2        ; ESTABLISH VECTOR SIZE
184: 4054 110054     LXI D,PMMA2     ; PARTIAL MATRIX MPY START ADDR
185: 4057 210053     LXI H,DMA2      ; DATA MATRIX START ADDR
186:
187: 405A             ; COLN2:
188: 405A F5          PUSH PSW        ; START NEXT COLUMN MPY, SAVE
189: 405B 0602       MUI B,RS2      ; ESTABLISH MATRIX ROW SIZE
190: 405D 4E          MOV C,M         ; GET OPERAND 81
191: 405E 23          INX H           ; POINT TO OPERAND 82
192:
193: 405F             ; ROWN2:
194: 405F C5          PUSH B          ; MPY THE ELEMENTS IN EACH ROW
195: 4060 E5          PUSH H          ; SAVE B,C,H & L
196: 4061 66          MOV H,M         ; GET OPERAND 82
197: 4062 CD4F41     CALL MULT        ; 8 BIT SIGNED
198: 4065 7C          MOV A,M         ; PREPARE TO STORE
199: 4066 12          STAX D          ; SAVE PARTIAL MATRIX MPY
200: 4067 13          INX D           ; ADJUST SAVE POINTER
201: 4068 E1          POP H           ; RESTORE H & L
202: 4069 23          INX H           ; POINT TO NEXT OPERAND
203: 406A C1          POP B           ; RESTORE VECTOR COUNT (B)
204:
205: 406B 05          DCR B            ; RESTORE OPERAND 81 (C)
206: 406C C25F40     JNZ ROWN2       ; COLUMN ALL DONE ?
207: 406F F1          POP PSW        ; NO
208: 4070 3D          DCR A           ; YES, RESTORE VECT SIZE
209: 4071 C25A40     JNZ COLN2       ; ALL MULTIPLIES COMPLETE ?
210:
211: 4074 0602       MUI B,RS2      ; NO
212: 4076 110055     LXI D,PMMA2     ; YES, SUM PARTIALS TO COMPLET
213: 4079 210054     LXI H,PMMA2     ; REESTABLISH MATRIX ROW SIZE
214:
215: 407C             ; SUMA2:
216: 407C 0E02       MUI C,US2      ; MATRIX MPY RESULT START ADDR
217: 407E AF          XRA A           ; PARTIAL MAT MPY START ADDR
218: 407F D5          PUSH D          ; REINITIALIZE VECTOR SIZE
219: 4080 1600       MUI D,0         ; CLEAR REG A
220: 4082 1E02       MUI E,RS2      ; OBTAIN POINTER OFFSET IN D&E
221:
222: 4084             ; SUMB2:
223: 4084 86          ADD A           ; CLEAR UPPER PORTION OF D&E
224: 4085 19          DAD D           ; SET OFFSET, MATRIX ROW SIZE
225: 4086 0D          DCR C           ;

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER

(CONTINUED)

```

226: 4087 C2B440 JNZ SUMB2 ; NO
227: 408A D1 POP D ; YES, RESTORE RESULT ADDR
228: 408B 12 STAX D ; STORE RESULT
229: 408C 05 DCR B ; MATRIX RPY COMPLETE ?
230: 408D CAA040 JZ DONE2 ; YES
231: 4090 13 INX D ; NO, POINT TO NEXT RESULT ADDR
232: 4091 05 PUSH D ; ADJUST TERM POINTER TO LAST
233: 4092 210054 LXI H,PMMA2 ; GET BASE ADDR
234: 4095 3E02 MVI A,MS2 ; GET MATRIX ROW SIZE
235: 4097 90 SUB B
236: 4098 5F MOV E,A ; GET OFFSET
237: 4099 1600 MVI D,0 ; CLEAR UPPER PORTION OF DE
238: 409B 19 DAD D ; HL = PMMA + US - REG B
239: 409C D1 POP D
240: 409D C37C40 JMP SUMA2
241:
242:
243: ; DO VECTOR ADDS TO COMPUTE U(K+1) I.E.
244:
245: ; U(K+1)1 = PHIDBU(K)1 + HDSZ(K)1
246: ; . . .
247: ; . . .
248: ; U(K+1)N = PHIDBU(K)N + HDSZ(K)N
249:
250:
251:
252: 40A0 DONE2: LXI H,DMA3 ; GET U(K+1) STORAGE AREA ADDR
253: 40A0 210056 PUSH H ; TEMP SAVE
254: 40A3 E5 LXI H,MMA1 ; OBTAIN HDSZ RESULT START A
255: 40A4 210052 LXI D,MMA2 ; OBTAIN PHIDBU RESULT START A
256: 40A7 110055 LXI B,US3 ; OBTAIN # OF TERMS TO ADD
257: 40AA 0602 MVI C,MS3 ; OBTAIN OFFSET TO STORE SUMS
258: 40AC 0E01 INR C
259: 40AE 0C
260:
261: ; ADDNEXT:
262: 40AF AF XRA A ; CLEAR A REG
263: 40B0 86 ADD M ; GET GDSZ TERM
264: 40B1 EB XCHG
265: 40B2 86 ADD M ; ADD PHIDBU TERM
266: 40B3 37 STC ; SET CARRY = 0
267: 40B4 3F CMC
268: 40B5 17 RAL ; ADJUST FOR 2.0 SCALING
269: 40B6 DAC440 JC NEG ; NEG OR POS ?
270: 40B9 D7 ORA A ; SET CONDITION CODE
271: 40BA F2BF40 JP NOFLO ; POSITIVE, ANY OVERFLOW ?
272: 40BD F87F ORI 7FH ; YES, FORCE TO LARGEST 8
273:
274: ; NOFLO:
275: 40BF E67F ANI 7FH ; FORCE TO POSITIVE 8
276: 40C1 C3C640 JMP SCAL1
277:
278: ; NEG:
279: 40C4 F6B0 ORI 80H ; FORCE TO NEGATIVE 8
280:
281: ; SCAL1:
282: 40C6 E3 XTHL ; GET STORAGE ADDRESS
283: 40C7 77 MOV B,A ; STORE SUM
284: 40C8 05 DCR B ; CHECK IF ALL TERMS ADDED
285: 40C9 CAD640 JZ DONE4
286: 40CC 79 MOV A,L ; ADJUST STORAGE ADDRESS
287: 40CD 81 ADD C
288: 40CE 6F MOV L,A
289: 40CF E3 XTHL ; RESTORE POINTERS
290: 40D0 EB XCHG
291: 40D1 23 INX H ; POINT TO NEXT GDSZ TERM
292: 40D2 13 INX B ; POINT TO NEXT PHIDBU TERM
293: 40D3 C3AF40 JMP ADDNEXT ; ADD NEXT TERMS
294:
295:
296: ; UPDATE U(K) WITH NEWLY CALCULATED U(K+1)
297:
298:
299: 40D6 DONE4: POP H ; RESTORE STACK
300: 40D8 E1

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONTINUED)

```

301: 40D7 2E02      MUI  L,US3      ; GET 8 OF TERMS TO STORE
302: 40D9 E5       PUSH  M      ; TEMP SAVE
303: 40DA 210056    LXI  H,DMA3  ; GET U(K+1) STORAGE AREA ADDR
304: 40DD 0E01      MUI  C,RS3  ; GET OFFSET OF U(K+1)
305: 40DF 0C       INR   C      ;
306: 40E0 110053    LXI  D,DMA2  ; GET U(K) STORAGE AREA ADDR
307: 40E3 0602      MUI  B,RS2  ; GET OFFSET OF U(K)
308: 40E5 04       INR   B      ;
309:
310: 40E6           ; STRNXT:
311: 40E6 7E       MOV   A,M      ; GET U(K+1)
312: 40E7 12       STAX  D      ; STORE INTO OLD U(K)
313: 40E8 E3       XTHL  D      ; GET 8 OF TERMS REMAINING TO
314: 40E9 2D       DCR   L      ; ANY LEFT ?
315: 40EA CAF740    JZ    DONES   ; NO
316: 40ED E3       XTHL  D      ; YES, RESTORE POINTER
317: 40EE 7D       MOV   A,L      ; ADJUST FOR NEXT U(K+1)
318: 40EF 81       ADD   C      ;
319: 40F0 6F       MOV   L,A      ;
320: 40F1 7B       MOV   A,E      ; ADJUST FOR NEXT U(K)
321: 40F2 80       ADD   B      ;
322: 40F3 5F       MOV   E,A      ;
323: 40F4 C3E640    JMP   STRNXT ;
324:
325:
326:           ;
327:           ; MULTIPLY MATRIX GD (CLOSED LOOP FEEDBACK MATRIX) UI
328:           ; VECTOR U (NEXT STATE VECTOR).
329:           ; GD IS 1 ROW X 2 COLS
330:           ; U IS 2 ROWS X 1 COL
331:
332: 40F7 E1       POP   M      ; RESTORE STACK
333: 40F8 3E02      MUI  A,US3  ; ESTABLISH VECTOR SIZE
334: 40FA 110057    LXI  D,PMA3  ; PARTIAL MATRIX MPV START ADDR
335: 40FD 210056    LXI  H,DMA3  ; DATA MATRIX START ADDR
336:
337: 4100           ; COLN3:
338: 4100 F5       PUSH  PSW      ; START NEXT COLUMN MPV, SAVE
339: 4101 0601      MUI  B,RS3  ; ESTABLISH MATRIX ROW SIZE
340: 4103 4E       MOV   C,M      ; GET OPERAND 81
341: 4104 23       INX   M      ; POINT TO OPERAND 82
342:
343: 4105           ; ROUN3:
344: 4105 C5       PUSH  B      ; MPV THE ELEMENTS IN EACH ROW
345: 4106 E5       PUSH  M      ; SAVE B,C,M & L
346: 4107 66       MOV   M,M      ; GET OPERAND 82
347: 4108 CD4F41    CALL  MULT   ; 8 BIT SIGNED
348: 4109 7C       MOV   A,M      ; PREPARE TO STORE
349: 410C 12       STAX  D      ; SAVE PARTIAL MATRIX MPV
350: 410D 13       INX   D      ; ADJUST SAVE POINTER
351: 410E E1       POP   M      ; RESTORE M & L
352: 410F 23       INX   M      ; POINT TO NEXT OPERAND
353: 4110 C1       POP   B      ; RESTORE VECTOR COUNT (B)
354:
355: 4111 05       DCR   B      ; RESTORE OPERAND 81 (C)
356: 4112 C20541    JNZ  ROUN3   ; COLUMN ALL DONE ?
357: 4115 F1       POP   PSW      ; NO
358: 4116 3D       DCR   A      ; YES, RESTORE VECT SIZE
359: 4117 C20041    JNZ  COLN3   ; ALL MULTIPLIES COMPLETE ?
360:
361: 411A 0601      MUI  B,RS3  ; NO
362: 411C 110058    LXI  D,PMA3  ; YES, SUM PARTIALS TO COMPLET
363: 411F 210057    LXI  H,PMA3  ; REESTABLISH MATRIX ROW SIZE
364:
365: 4122           ; SUMA3:
366: 4122 0E02      MUI  C,US3  ; MATRIX MPV RESULT START ADDR
367: 4124 AF       XRA   A      ; PARTIAL MAT MPV START ADDR
368: 4125 B5       PUSH  D      ; REINITIALIZE VECTOR SIZE
369: 4126 1600      MUI  D,0     ; CLEAR REG A
370: 4128 1E01      MUI  E,RS3  ; OBTAIN POINTER OFFSET IN D&E
371:
372: 412A           ; SUMB3:
373: 412A 86       ADD   M      ; CLEAR UPPER PORTION OF D&E
374: 412B 19       DAD   D      ; SET OFFSET, MATRIX ROW SIZE
375: 412C 0B       DCR   C      ; A = A + HL
                        ; HL = HL + DE (DE = OFFSET)
                        ; ALL TERMS SUMMED ?

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONTINUED)

```

376: 412D C28A41 JNZ SUBB3 ; NO
377: 4130 B1 POP D ; YES, RESTORE RESULT ADDR
378: 4131 12 STAX B ; STORE RESULT
379: 4132 06 DCR B ; MATRIX RPY COMPLETE ?
380: 4133 CA4641 JZ DONE3 ; YES
381: 4136 13 INX D ; NO, POINT TO NEXT RESULT ADDR
382: 4137 D5 PUSH D ; ADJUST TERM POINTER TO LAST
383: 4138 210057 LXI H,PMMA3 ; GET BASE ADDR
384: 4139 3E01 MVI A,RS3 ; GET MATRIX SIZE
385: 413D 90 SUB B
386: 413E 5F MOV E,A ; GET OFFSET
387: 413F 1600 MVI D,0 ; CLEAR UPPER PORTION OF DE
388: 4141 19 DAD D ; HL = PMMA + US - REG B
389: 4142 B1 POP D
390: 4143 C38241 JMP SUBA3
391:
392: 4146 ;
393: 4146 37 STC ; SET CARRY = 0
394: 4147 3F CMC ; CHECK FOR INTERRUPT COMPLETE
395: 4148 FB EI ; NEEDED FOR 1ST PASS ONLY
396:
397: 4149 ;
398: 4149 B24941 JNC WAITLP ; WAIT FOR INTERRUPT
399: 414C C30440 JMP START1 ; INTERRUPT SERVICED
400: ; CALCULATE NEXT OUTPUT
401:
402: ; SUBROUTINE 'MULT' --- 8 BIT SIGNED MULTIPLY
403: ;
404: ; HL = HSC
405: ;
406: ; INPUTS: C - MULTIPLICAND 8 BIT SIGNED
407: ; M - MULTIPLIER 8 BIT SIGNED
408: ;
409: ; OUTPUTS: H&L - PRODUCT 16 BIT SIGNED
410: ;
411: ; DESTROYS: A,B,C,H,L
412: ;
413: 414F ;
414: 414F 7C MULT: MOV A,M ; CHECK SIGN OF MULTIPLIER (M)
415: 4150 B7 ORA A
416: 4151 F28E41 JP MULNP ; M IS POSITIVE
417: 4154 2F CMA ; MULTIPLIER (M) IS NEGATIVE
418: 4155 3C INR A ; TAKE 2'S COMPLIMENT
419: 4156 67 MOV M,A ;
420: 4157 7D MOV A,C ; CHECK SIGN OF MULTIPLICAND
421: 4158 B7 ORA A
422: 4159 F26341 JP MULOS ; INPUTS HAVE OPPOSITE SIGNS
423: 415C 2F CMA ; MULTIPLICAND (C) IS
424: 415D 3C INR A ; TAKE 2'S COMPLIMENT
425: 415E 4F MOV C,A ;
426: ;
427: 415F ;
428: 415F CD7941 MULSS: CALL INUL ; SAME SIGN, MULTIPLY AND RETU
429: 4162 C9 RET
430: ;
431: 4163 ;
432: 4163 CD7941 MULOS: CALL INUL ; M & C HAVE OPPOSITE SIGNS
433: 4166 2D DCX M ; TAKE 2'S COMPLIMENT OF PRODU
434: 4167 7D MOV A,L
435: 4168 2F CMA
436: 4169 8F MOV L,A ; 2'S COMP OF L
437: 416A 7C MOV A,M
438: 416B 2F CMA
439: 416C 67 MOV M,A ; 2'S COMP OF M
440: 416D C9 RET ; RETURN WITH FINAL RESULT IN
441: ;
442: 416E ;
443: 416E 7D MULNP: MOV A,C ; M (MULTIPLIER) IS POSITIVE
444: 416F B7 ORA A ; CHECK SIGN OF MULTIPLICAND
445: 4170 F25F41 JP MULSS ; MULTIPLICAND (C) IS NEGATIVE
446: 4173 2F CMA ; TAKE 2'S COMPLIMENT
447: 4174 3C INR A
448: 4175 4F MOV C,A
449: 4176 C36341 JMP MULOS ; DO OPPOSITE SIGN MULTIPLY
450: ;

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONTINUED)

```

451:                                     ;
452:                                     ; SUBROUTINE 'INUL' --- 8 BIT UNSIGNED FRACTIONAL
453:                                     ;
454:                                     ; INPUTS: C - MULTIPLICAND          8 BIT UNSIGNED
455:                                     ;          M - MULTIPLIER          8 BIT UNSIGNED
456:                                     ;
457:                                     ; OUTPUTS: HL - PRODUCT          16 BIT UNSIGNED
458:                                     ;
459:                                     ; DESTROYS: A,B,H,L
460:                                     ;
461: 4179 INUL:                                     ;
462: 4179 0600 RUI B,0          ; CLEAR FOR FOLLOWING 'DAD' IN
463: 4179 68   ROU L,8          ; CLEAR BOTTOM HALF OF HL
464: 417C 3E00 RUI A,8          ; INITIALIZE LOOP COUNTER
465:                                     ;
466: 417E INUL1:                                     ;
467: 417E 29   DAD H            ; SHIFT RESULT
468: 417F D20341 JNC INUL2      ; IF MSB SET, ADD MULTIPLICAND
469: 4182 00   DAD B            ; HL = HL + BC
470:                                     ;
471: 4183 INUL2:                                     ;
472: 4183 3D   DCR A            ; DECREMENT & TEST LOOP COUNT
473: 4184 C27E41 JNZ INUL1      ;
474: 4187 29   DAD H            ; ADJUST FOR FRACTIONAL MPY
475: 4188 C0   RET
476:                                     ;
477:                                     ;
478:                                     ; INTERRUPT SERVICE ROUTINE
479:                                     ;
480:                                     ; FUNCTIONAL DESCRIPTION:
481:                                     ;
482:                                     ; THIS ROUTINE IS ENTERED WHEN THE CLOCK (DELTA T)
483:                                     ; GOES OFF. DELTA T IS A SQUARE WAVE CLOCK INPUT
484:                                     ; FROM A FUNCTION GENERATOR. HENCE IT IS PRESETTABLE
485:                                     ; IT PROVIDES:
486:                                     ;
487:                                     ; A) A DELTA T TIME STEP
488:                                     ; B) A MEANS OF INDICATING END OF CONVERSION FOR
489:                                     ;
490:                                     ; A2D MAX CONVERSION TIME IS .05 MS. HENCE DELTA T
491:                                     ; BE SET HIGHER
492:                                     ;
493:                                     ; THE ROUTINE READS IN A 12 BIT A2D INPUT Z(K), WHICH
494:                                     ; TRUNCATES THE LEAST SIGNIFICANT 4 BITS SINCE ONLY 8
495:                                     ; BITS ARE NEEDED.
496:                                     ;
497:                                     ; THE ROUTINE ALSO OUTPUTS U(K) TO THE ANALOG SYSTEM
498:                                     ; MEANS OF A D2A. THE D2A IS 8 BIT MEMORY MAP IO.
499:                                     ; 2 OUTPUTS ARE PROVIDED: 1 TO THE SYSTEM
500:                                     ; 1 TO A STRIP CHART RECORDER
501:                                     ;
502:                                     ; IN ADDITION, A CHECK IS DONE TO SEE IF THE
503:                                     ; INTERRUPT OCCURRED DURING CONTROL CODE
504:                                     ; CALCULATIONS, WHICH COULD RESULT IN
505:                                     ; INACCURATE CONTROL COMMANDS. IF THIS
506:                                     ; ERROR OCCURS, THEN PGM CONTROL IS
507:                                     ; PASSED TO THE MONITOR.
508:                                     ;
509: 4189 INTR:                                     ;
510: 4189 F3   DI              ; DISABLE INTERRUPTS
511: 418A F5   PUSH PSW        ; SAVE A
512: 418B E5   PUSH H          ; SAVE HL
513:                                     ;
514:                                     ; CHECK IF CONTROL CODE COMPLETED
515:                                     ;
516: 418C 33   INX SP          ; CHECK IF CONTROL CODE COMPLE
517: 418D 33   INX SP          ; FIND RET ADDR FROM INTERRUPT
518: 418E 33   INX SP
519: 418F 33   INX SP
520: 4190 214941 LXI H, WAITLP ; GET WAIT LOOP HI ADDR
521: 4193 7C   ROU A, H
522: 4194 E3   XTML           ; GET INTERRUPT RET HI ADDR
523: 4195 94   SUB H           ; ARE THEY EQUAL ?
524: 4196 C2D241 JNZ ERR1      ; NO, INDICATE ERR, STOP PGM
525: 4199 7D   ROU A, L        ; GET INTERRUPT RET LO ADDR

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONTINUED)

```

526: 419A E3          XTHL          ; GET WAIT LOOP LO ADDR
527: 419B 95          SUB          L          ; ARE THEY EQUAL ?
528: 419C C2D241      JNZ ERR1        ; NO, INDICATE ERR, STOP PGM
529: 419F 3B          DCX SP          ; YES, CONTROL CODE WAS COMPLE
530: 41A0 3B          DCX SP          ; RESTORE STACK PNTR TO NORMAL
531: 41A1 3B          DCX SP
532: 41A2 3B          DCX SP
533:
534:                ; READ A/D
535:
536: 41A3 3E82          MUI A,B2H      ; SET A/D READ CHANNEL
537: 41A5 D3E7          OUT          0E7H  ; AMPLIFIER INPUT
538: 41A7 DBE5          IN          0E5H  ; READ LOW BYTE
539: 41A9 0F          RRC            ; ADJUST FOR 4 BIT THROUAAWAY
540: 41AA 0F          RRC
541: 41AB 0F          RRC
542: 41AC 0F          RRC
543: 41AD E60F          ANI          0FH   ; PRESERVE LOW NIBBLE
544: 41AF 6F          MOV          L,A    ; TEMP SAVE
545: 41B0 DBE4          IN          0E4H  ; READ HIGH BYTE
546: 41B2 0F          RRC            ; ADJUST FOR HIGH NIBBLE
547: 41B3 0F          RRC
548: 41B4 0F          RRC
549: 41B5 0F          RRC
550: 41B6 E6F0          ANI          0F0H  ; PRESERVE HIGH NIBBLE
551: 41B8 95          ORA          L      ; MERGE TO FORM 8 BIT INPUT
552: 41B9 210050        LXI          M,DRA1 ; STORE INPUT IN MATRIX DATA A
553: 41BC 77          MOV          M,A
554:
555:                ;
556:                ; OUTPUT D/A (U)
557:                ; MEMORY MAPPED I/O
558:                ;
559:                ;
560: 41BD 210050        LXI          M,UOUT  ; OUTPUT U
561: 41C0 7E          MOV          A,M
562: 41C1 2100F7        LXI          M,0F700H
563: 41C4 77          MOV          M,A
564: 41C5 210053        LXI          M,STRIPC ; OUTPUT MEMORY LOC TO STRIP C
565: 41C8 7E          MOV          A,M
566: 41C9 2101F7        LXI          M,0F701H
567: 41CC 77          MOV          M,A
568: 41CD E1          POP          M
569: 41CE F1          POP          PSU
570: 41CF 37          STC            ; INDICATE INTERRUPT COMPLETE
571: 41D0 FB          EI
572: 41D1 C9          RET            ; ENABLE INTERRUPTS & RETURN
573:
574:                ; CONTROL CODE NOT COMPLETED, ERROR
575:                ;
576: 41D2          ERR1: RST          1      ; BRANCH TO MONITOR IMMEDIATELY
577: 41D2 CF
578:
579:                ;
580:                ; DATA MATRIX STORAGE AREA
581:                ;
582:                ; THE FOLLOWING DATA IS FOR THE 2ND ORDER SYSTEM
583:                ; ALL 0'S ARE REPRESENTED AS FRACTIONS WHERE:
584:                ; +0 = FRAC.3128+0.5
585:                ; -0 = 2'S COMP OF (-FRAC.3128+0.5)
586:                ;
587:                ;
588:                ;
589:                ; NOTE: GAINS BELOW ARE FOR T = 0.1 SEC
590:                ;
591:                ;
592:                ;
593:                ; Z - INPUT VECTOR FROM A2D
594:                ; MD - MATRIX
595:                ;
596:                ; MD = .113 = .057 WHEN SCALED ON 2.0
597:                ;      -.036 = -.018
598:                ;
599: 5000          ORG          5000H
600:

```

INTEL 8080 SOFTWARE FOR LQG CONTROLLER (CONCLUDED)

```

601:      ; NOTE: MD(1) & MD(2) HAVE BEEN COMPLIMENTED
602:      ; DUE TO INVERTED INPUT.
603:
604:      5000  00      DB  0H      ; Z(K) VECTOR A/D INPUT 01
605:      5001  00      DB  0F9H    ; MD MATRIX ROW1 COL1
606:      5002  16      DB  01H     ; MD MATRIX ROW2 COL1
607:
608:
609:      ; U(K) - PAST STATE VECTOR
610:      ; PHID - CLOSED LOOP SYSTEM MATRIX
611:
612:      PHID = .87264 .07127  * .43632 .03563
613:             -.86524 .63205  * -.13862 .31603
614:             ; WHEN SCALED ON 2.0
615:
616:      5300      ORG  5300H
617:      5300  40      DB  40H      ; U(K) PAST STATE 01 INIT .5
618:      5301  30      DB  30H      ; PHID MATRIX ROW1 COL1
619:      5302  F0      DB  0F0H     ; PHID MATRIX ROW2 COL1
620:      5303  40      DB  40H      ; U(K) PAST STATE 02 INIT .5
621:      5304  05      DB  5H       ; PHID MATRIX ROW1 COL2
622:      5305  28      DB  28H      ; PHID MATRIX ROW2 COL2
623:
624:
625:      ; U(K+1) - PRESENT STATE VECTOR
626:      ; GD - CONTROL GAIN MATRIX
627:
628:      GD = -1.742  -.41412  *  -.871  -.20706
629:             ; WHEN SCALED ON 2.0
630:
631:      5600      ORG  5600H
632:      5600  00      DB  0H       ; U(K+1) PRESENT STATE 01
633:      5601  92      DB  92H      ; GD MATRIX ROW1 COL1
634:      5602  00      DB  0H       ; U(K+1) PRESENT STATE 02
635:      5603  E6      DB  0E6H     ; GD MATRIX ROW1 COL2
636:
637:
638:      ;
639:      NO PROGRAM ERRORS
640:
641:
642:      SYMBOL TABLE
643:
644:      $ 01
645:
646:      A 0007 ADDHX 40AF B 0000 C 0001
647:      COLN1 400C COLN2 405A COLN3 4100 D 0002
648:      DRA1 5000 DRA2 5300 DRA3 5600 DONE1 4052
649:      DONE2 40A0 DONE3 4146 DONE4 40D6 DONE5 40F7
650:      E 0003 ERR1 41D2 H 0004 INUL 4179
651:      INUL1 417E INUL2 4183 INTR 4189 L 0005
652:      N 0006 NARA1 5200 NARA2 5500 NARA3 5800
653:      NS1 0002 NS2 0002 NS3 0001 NULMP 416E
654:      NULOS 4163 NULSS 415F NULT 414F NEG 40C4
655:      NOFLO 40BF PRAA1 5100 PRAA2 5400 PRAA3 5700
656:      PSU 0006 ROUN1 4011 ROUN2 405F ROUN3 4105
657:      SCAL1 40C6 SP 0006 STRIP 5300 STRHX 40E6
658:      STR71 4004 SUMA1 402E SUMA2 407C SUMA3 4122
659:      SUBB1 4036 SUBB2 40B4 SUBB3 412A UOUT 5800
660:      US1 0001 US2 0002 US3 0002 WAITL 4149
661:

```

SECOND-ORDER OUTPUT RESPONSE AS A FUNCTION OF WORD LENGTHRESPONSE TO INITIAL CONDITION: $x_0 = (0.5, 0.5)$

STANDARD STRUCTURE WITHIN CONTROLLER

 $\Delta t = 0.1$ SEC, WORD LENGTH = 8 BITS