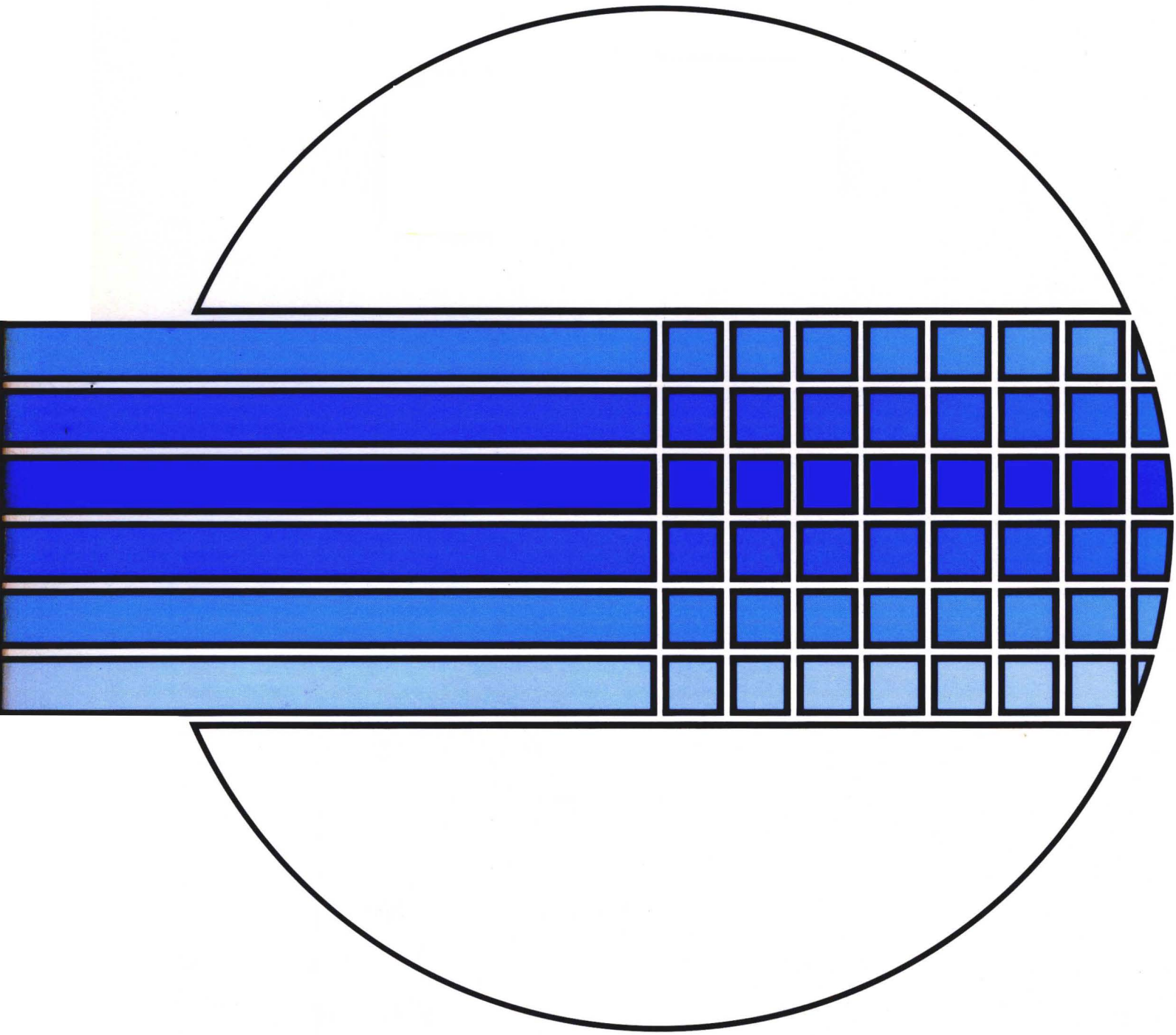
SIGNETICS FIELD PROGRAMABLE LOGIC ARRAYS

AN APPLICATIONS MANUAL

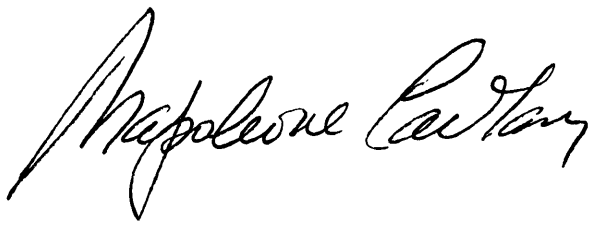


FOREWORD

Today, more than ever, beating your competitors to the market has become the focal point of marketing strategy. To be the winner, it is no longer enough to compact in a system more functions, speed options, and cost advantages. You have got to design in flexibility! Flexibility to speed prototype development, and reconfigure a basic product to meet new market demands and opportunities.

These considerations are putting more pressure on designers already burdened by the choice between custom logic, or standard circuits. Today, Signetics' Field Programmable Logic Arrays offer a new alternative: a fast, *user programmable,* TTL logic element with memory, which will streamline logic system design by integrating the equivalent of 528 TTL gates in 196 packages into a *single* IC package.

Since their introduction to the market, FPLAs have steadily inched their way into a growing number of applications. This book contains a representative sample of the direction of current design activities with FPLAs. It is intended to illustrate the basic role of FPLAs in logic design, and stimulate the transfer of these ideas to other practical applications.



NAPOLEONE CAVLAN

Manager, Advanced Products Marketing
February, 1977

TABLE OF CONTENTS

Acknowledgements ii
Basic Review iii
Program Table vi
Data Cartridge Tape Drive Controller 1
Serial Impact Printer Controller 5
Distributed Controller 7
Pneumatic Valve Controller 9
Sequential Traffic Controller 10
Sequential Control System Using an FPLA ... 11
"N" Expandable System Status and Control Logic Unit 12
Ignition Timing For I/C Engine Using Multi-Dimensional Curve Fit 13
Using FPLAs and SRGs to Store Huffman Codes 14
N-Out-Of-8 Decode 18
16-Bit Shift and Logic Unit 22
FPLA Modifies Sequential Circuit Design 27
Bandswitching Code Converter 28
RF-Channel Control Selector 29
Serial Data Filter 33
8-Channel Pulse Width Discriminator 35
Timing Decoder & Sequence Controller For Data Acquisition System 36
Controller for Single TDM Data Channel 41
ATC Transponder Encoder 44
Mobile FM-VHF Transceiver Uses FPLA for Quick Channel Selection 45
Programmable Divider for Frequency Synthesizer 47
Programmable Frequency Synthesizer 50
An IEEE-488 to Speech Synthesizer Interface 50
Function Decoding in Camac Modules 52
Programmable Waveform Generator 53
Programmable Function Generator 55
Register Port Selector for 4-Bit Slice Microprocessors 56
OP-Code Decoder for PDP-11 MPCU 58
Target Instruction Decoder for Micro-programmed Emulator 61
Single or Block Address Decoding for M6800μP 63
Interrupt Priority Control for M6800μP 63
Unibus Address Monitor 65
Hardware-Macro Generator 66
FPLA Enhances Pipelined Process Architecture 66
I/O Port Decoder 67
Peripheral Address Decoder 68
Memory Address Trap 68
Maskable Interrupt Vector Generator 68
15-Input Priority Encoder and I.D. Generator 70
FPLA Simplifies Alpha-Numeric Display Operation 70
An Economical Digital Display Conditioner ... 73
Rate Measurement of Low Frequency Events 79
Logic Analyzer 81
Combination Sync & Video Generator 82
Court and Scoreboard Generator for Ping Pong T.V. Game 86
Intelligent Tape Reader to T.V. Video Display Interface 87
Digital Message Decoder 88
Calculator Chip Interface 90
Microprocessor to Calculator-Chip Interface .. 93
Hand-Held ASCII Keyboard 95
7-Segment to BCD Converter 97
Decimal Point & Legend Display Driver 98
One to Sixteen Digit Combination Lock 99
Data Security Encoder 100
Calendar Clock Date Reminder 101
Real Time Preventive Maintenance Monitor .. 102
Signetics Sales Offices & Distributors 105

ACKNOWLEDGEMENTS

Signetics gratefully acknowledges the work of the following contributors to this book, and their effort in pioneering the application of new technology:

Abbott, Douglas
Armstrong, R.

Barton, Jeff
Baum, Allen
Bookstein, Ben
Borotz, Richard
Brown, G. E.

Corl, Ernest R.

Elger, Rodney
Equizabal, Antonio

Farrow, John F.
Flick, Richard
Fuchs, Kenneth J.

Gajski, Daniel D.
Glissmann, Randy
Graden, Maynard
Grinberg, Bernard

Hackleman, David
Hank, Max
Hicks, Arthur N., II
Hinds, James J.

Jaworski, Joe

Kanitz, Bruce R.
Karalis, Ed
Kiraly, Louis J.

Lai, Kwok-Woon
Lange, Arthur F.
Larson, Robert L.
Lawton, Jeffrey C.
LaZar, Les
Lewis, Gene R.

McMullen, John
Menningen, Lee A.
Miller, Wayne
Mitchell, Thomas
Moore, Thomas J., III

Nelson, Edward M.
Nelson, Robert L., Jr.

Otto, Vedan
Overhouse, Van

Pinkus, Walter H.

Rentz, Mark
Roberts, Larry

Schaubel, Wm.
Shaffer, Stephan D.

Trinh, Huynh

Venhaus, Don
Vermeulen, John C.
Vogt, Pete D.

Whatcott, Gary L.
Wolcott, James L.
Wooten, Curran
Woughter, W. R.

Signetics' FPLAs are logic elements with memory, and can be viewed in two basic ways.

In terms of logic, the FPLA is a two level AND-OR, AND-NOR *combinatorial* logic element, consisting of a system of logic gates with programmable inputs and outputs as shown in Figure 1. These, by means of on-chip programmable connectors, enable the user to quickly implement 8 logic functions with a maximum of 48 product (AND) terms, involving up to 16 input variables.

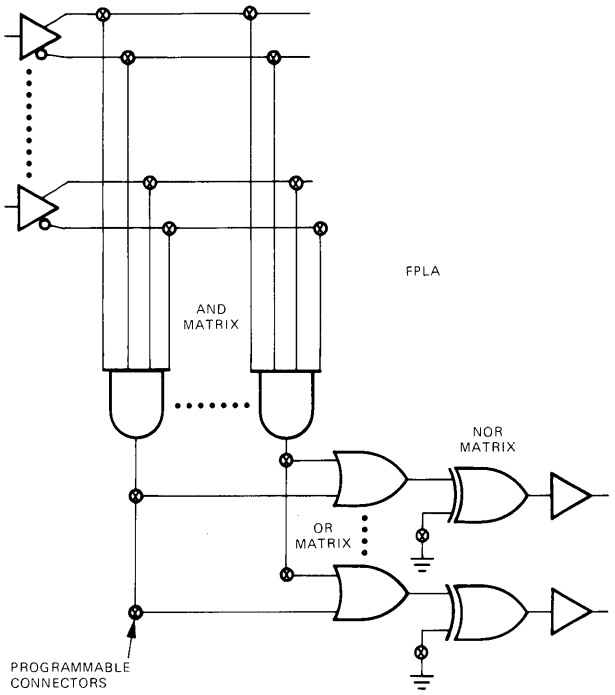


Figure 1: FPLA as a two-level, programmable logic element.

A more detailed organization of the FPLA is shown in Figure 2. The first logic level, the AND matrix, consists of 48 resistor-diode AND gates each connected to 16 True and Complement inputs via 32 fusible nichrome links. The second logic level, the OR matrix, consists of 8 emitter follower OR gates each connected by fusible links to all 48 outputs from the AND matrix. The output of each of these 8 OR gates is in turn buffered through an EX-OR gate which allows changing the logic to NOR via a fusible link to ground.

If your design involves a random logic structure, the FPLA can be used in place of discrete gates and wires. You gain

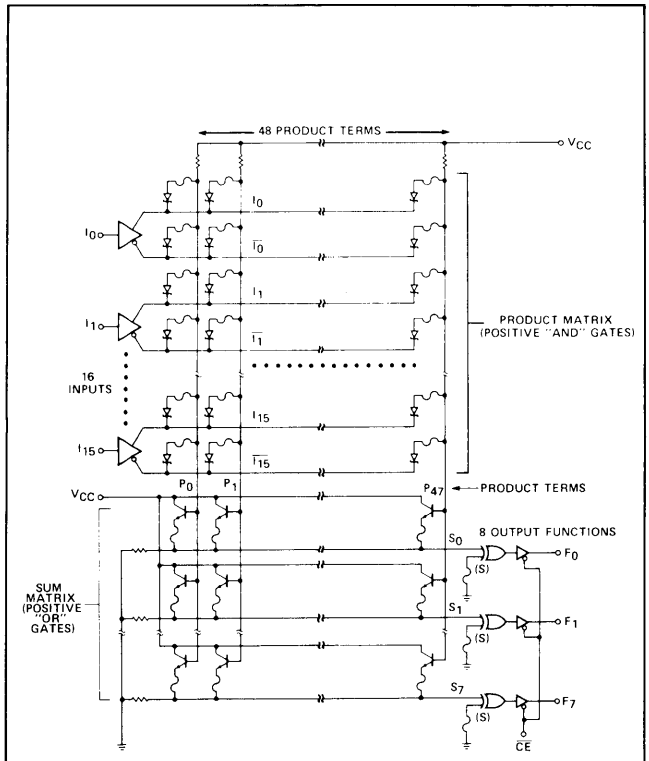


Figure 2: Logic structure of FPLA, illustrating AND, OR, and EX-OR arrays.

total flexibility from the immediate availability of your logic function for which you need only use as much of the FPLA as necessary for your particular application, with the rest available for later expansion or modification. But, if your design needs to be structured more like a memory, then the FPLA can be used in another, perhaps more effective way.

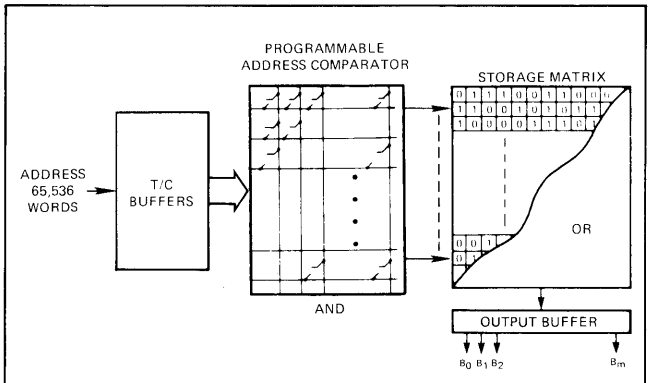


Figure 3: FPLA as a conditionally addressable 48 x 8 memory.

From the partitioning shown in Figure 3, the FPLA can also be viewed as a conditionally addressable memory; one in which the AND matrix functions as a programmable address comparator, recognizing only the preprogrammed address combinations used to activate the system, and ignoring all others. Plus, an OR matrix which then functions as a storage array for 48 output words, each containing 8 bits representing active or idle system commands. Or, to put it another way, you have a memory system that can logically scan a total address field up to 65,536 words deep, to linearly select down to 48 replies randomly scattered within that address space.

No matter how you look at it, generally FPLAs are effectively used in design situations where you have many input variables and few active logic states; and, with a maximum access time of 50ns, the FPLA is a practical alternative to the long logic chains necessary when dealing with several input variables.

The following example is a brief, but concise illustration of how to integrate random logic with discrete gates into a Signetics' FPLA.

Given the set of logic equations $F_{1\text{-}4}$ below:

$F_1 = X_1 + \overline{X}_2\overline{X}_3$ $\qquad$ $F_3 = X_3 + \overline{X}_1\overline{X}_2$

$F_2 = X_2 + \overline{X}_1\overline{X}_3$ $\qquad$ $F_4 = \overline{X}_1X_3 + \overline{X}_1\overline{X}_2$

These can be implemented with discrete gates as in the AND-OR-NOT logic network of Figure 4.

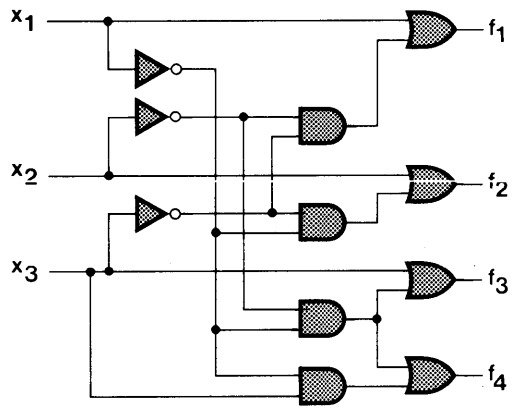


Figure 4: Discrete gate logic network for equation set $F_{1\text{-}4}$.

This method is practical for simple systems; but in more complex applications, it soon produces a distributed logic network with many I.C. packages and types, difficult to design, troubleshoot and modify.

On the other hand, the same set of equations can be easily coded in an FPLA Program Table* and programmed in a device using inexpensive field equipment. Typically, F_1 would require the FPLA to contain the fused link pattern shown in Figure 5, as specified in the accompanying Program Table slice. Overall, all four logic functions would use 3 inputs, 4 outputs, and 7 product terms of the FPLA, leaving remaining resources spare for later modifications.

FPLA PROGRAM TABLE

PRODUCT TERM NO	INPUT VARIABLE (I_m) 5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
																	H H H H								
0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H						•	•	•	A
1	–	–	–	–	–	–	–	–	–	–	–	–	L	L	–							•	•	•	A

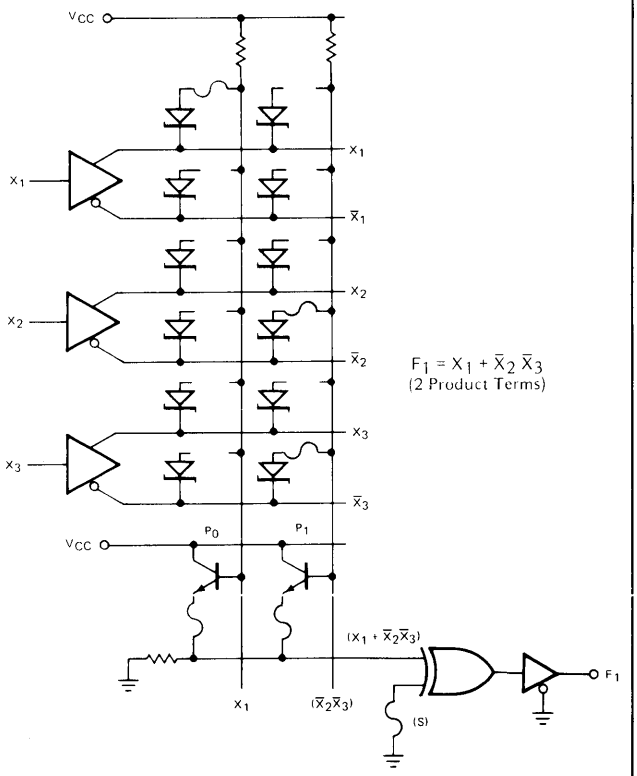


Figure 5: Integrating logic with FPLAs.

*See page vi.

For example, if it becomes necessary to change the X_1 product term in F_1 to $\overline{X}_1$, deleting the wrong product term and adding the new one becomes a trivial task, as indicated in the modified pattern and revised Program Table of Figure 6.

FPLA PROGRAM TABLE

PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE (I_m)																			H	H	H	H
NO	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H					•	•	•	•
1	–	–	–	–	–	–	–	–	–	–	–	–	–	L	L	–					•	•	•	A
2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L					•	•	•	A

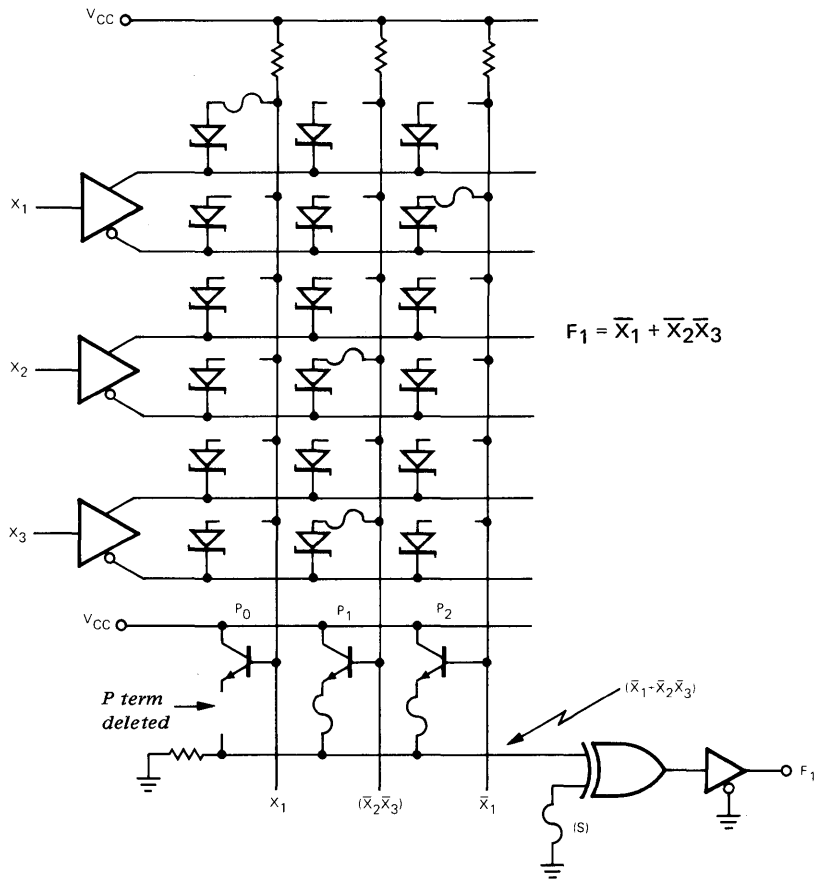


Figure 6: Modifiying random logic with FPLAs. F_1 modified by deleting term X_1 in the OR matrix and adding new term $\overline{X}_1$.

16X48X8 FPLA PROGRAM TABLE

THIS PORTION TO BE COMPLETED BY SIGNETICS

CF (XXXX) _____

CUSTOMER SYMBOLIZED PART # _____

DATE RECEIVED _____

COMMENTS _____

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # _____

TOTAL NUMBER OF PARTS _____

PROGRAM TABLE # _____

REV _____ DATE _____

PROGRAM TABLE ENTRIES						
INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
Im	$\overline{Im}$	Don't Care	Prod. Term Present in F_P	Prod. Term Not Present in F_P	Active High	Active Low
H	L	— (dash)	A	• (period)	H	L
NOTE Enter (—) for unused inputs of used P-terms.			NOTES 1. Entries independent of output polarity. 2. Enter (A) for unused outputs of used P-terms.		NOTES 1. Polarity programmed once only. 2. Enter (H) for all unused outputs.	

	PRODUCT TERM[1]																ACTIVE LEVEL[1]							
	INPUT VARIABLE[1]																OUTPUT FUNCTION[1]							
NO.	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0																								
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9																								
10																								
11																								
12																								
13																								
14																								
15																								
16																								
17																								
18																								
19																								
20																								
21																								
22																								
23																								
24																								
25																								
26																								
27																								
28																								
29																								
30																								
31																								
32																								
33																								
34																								
35																								
36																								
37																								
38																								
39																								
40																								
41																								
42																								
43																								
44																								
45																								
46																								
47																								

(1) Input and Output fields of unused P-terms can be left blank. Unused inputs and outputs are FPLA terminals left floating.

The circuit and associated state diagram in Figures 1 and 2 illustrate a 14-state logic machine which controls the tape motion of a Data Cartridge Tape Drive (Qantex Model 600). It allows the tape drive to be operated externally by a computer and internally according to the routines programmed into the FPLA. The FPLA contains all of the combinational logic required to implement the external commands and internal routines. And, by reducing total IC count, it contributed to reduce from 2 to 1 the number of circuit boards used in the tape drive.

In the circuit of Figure 1, all inputs are derived from I/O commands, the status of the tape drive, and the output of the four J-K flip-flops. The states of the flip-flops determine which state the logic machine is in as defined in Figure 3. The outputs are used to operate the velocity servo which drives the Data Cartridge to form I/O status signals, and to enable the writing of data.

The current state of the logic machine (Q1, Q2, Q3, Q4,) is decoded by the FPLA and combined with the other input signals to determine which state the machine will be switched to next. The possible sequences are described by the Flow Diagram of Figure 2. A jump to a following state can occur when the intervening conditions are true. All desired jumps are programmed in the FPLA Program Table of Figure 4. Jumps occur on the trailing edge of the clock pulse. The state codes (Figure 3) are decoded by the PROM

SCHEMATIC OF TAPE DRIVE CONTROLLER

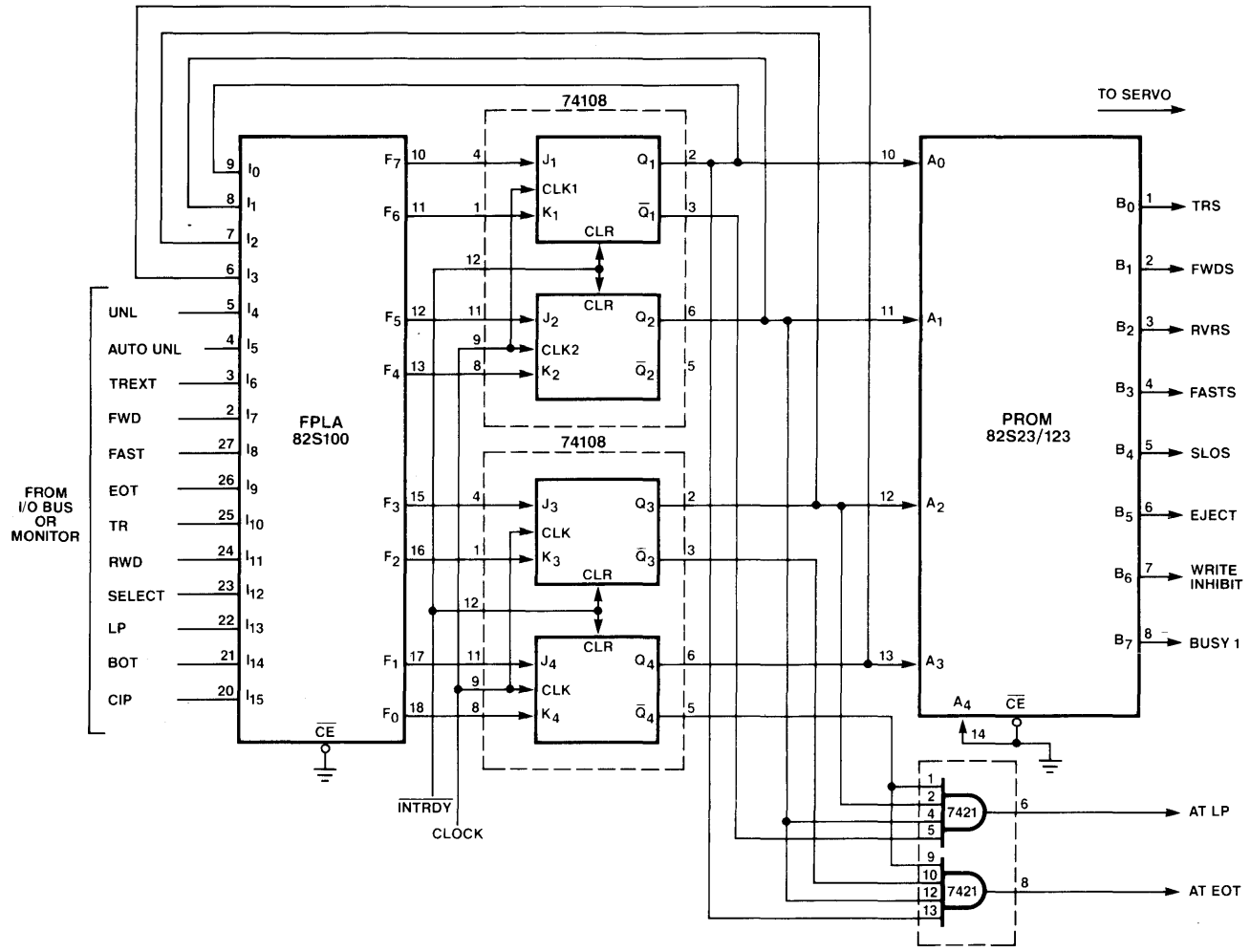


Figure 1.

FLOW DIAGRAM OF CONTROLLER

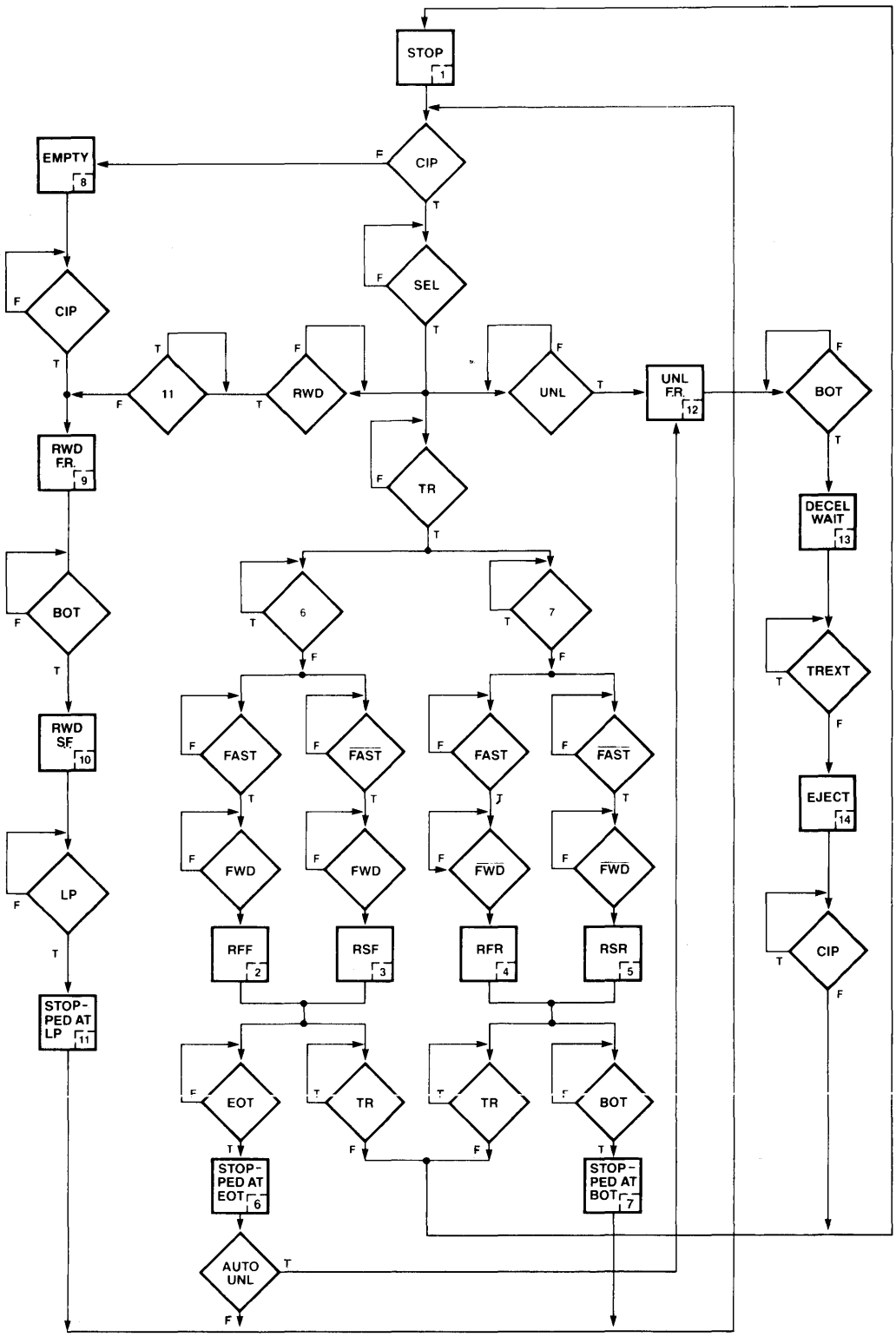


Figure 2.

STATE CODE ASSIGNMENTS

STATE		CODE			
NO.	NAME	Q1	Q2	Q3	Q4
1	STOP	0	0	0	0
2	RFF	1	0	0	0
3	RSF	0	1	0	0
4	RFR	0	0	1	0
5	RSR	0	0	0	1
6	STOPPED AT EOT	1	1	0	0
7	STOPPED AT BOT	0	0	1	1
8	EMPTY	1	0	1	0
9	RWD FR	1	1	1	0
10	LDSF	0	1	1	1
11	STOPPED AT LP	0	1	1	0
12	UNLFR	1	1	0	1
13	DECEL WAIT	1	0	1	1
14	EJECT	1	0	0	1

Figure 3.

and the two AND gates with the assignment tabulated in Figure 5. The decoded outputs provide the various signals needed to operate the tape drive.

Whenever the tape drive power is turned on, or an interlock opened, the tape drive is required to be stopped. This defines state 1, STOP, which has been assigned the code 0000. The state is achieved by using $\overline{\text{INTRDY}}$ to clear the four J-K flip-flops. Once set to STOP, operation at normal write speeds can occur when the following set of conditions are simultaneously satisfied, as diagrammed in Figure 2:

a. Data cartridge is in place: CIP true
b. Tape Drive has been addressed: SEL true
c. Tape has been commanded to run: TR true
d. The logic machine is *not* in state 6: 6 false
e. Tape should move at slow speed: $\overline{\text{FAST}}$ true (=slow)
f. Tape should move forward: FWD true

Then the logic machine will jump from state 1 to 3 (RSF). The preceding conditions are programmed in jump 1-3 of Figure 4. the outputs implemented by the new state are described in Figure 5 under state 3. Reading across the line for state 3 of Figure 5, notice that the servo will be driven (TRS true) in the forward direction (FWDS true) at the slow speed (SLOS true).

After data has been either written or read, the tape drive is commanded to stop (TR false). This allows a jump from state 3 (run-slow-forward) to state 1 (STOP). The jump from state 3 to 1 is charted in Figure 2 and programmed in Figure 4. Figure 5 shows that the only output active in state 1 is Write Inhibit.

By similar agruments, the tape drive can be run either fast or slow in either forward or reverse direction (states 2, 4 and 5).

When the end of tape is reached (EOT true), the tape drive is stopped. This is implemented by a jump from 2-6 or 3-6. Once in state 6, the tape drive can no longer move in the forward direction because of the state 6 false condition preceding states 2 and 3. If auto-unload is true, the drive will automatically rewind (state 12), wait for tape to decelerate (state 13), eject the tape cartridge (state 14) and stop (state 1). If auto-unload is false, the tape drive must wait for either a rewind command (RWD), an unload command (UNL), or a reverse command.

If the tape should be moved in the reverse direction until the beginning of tape is reached, the tape drive is stopped. This is implemented by a jump from 4 to 6 to 7. Once in state 7, the tape drive can no longer move in the reverse direction because of the state 7 false condition preceding states 4 and 5. The tape drive will remain at state 7 (STOPPED AT BOT) until RWD, UNL, or FWD command is given.

If no Data Cartridge is in place (CIP false) when the tape drive is turned on, the logic machine will jump from 1 to 8 (EMPTY). When a cartridge is installed, CIP goes true implementing a jump to 8 to 9 (RWD FR). In state 9, the tape will be rewound in fast reverse until a BOT mark is reached. BOT true implements a jump from 9 to 10 (LDSF). Tape now runs at slow speed in the forward direction until load point (LP) is reached. LP true implements a jump from 10 to 11 (STOPPED AT LP). From state 11, a forward, reverse, or UNL command can be implemented. RWD cannot be implemented from state 11 because of the state 11 condition preceding state 9. This keeps RWD from being needlessly repeated.

The logic machine implements and controls the following tape drive operations and routines:

a. move tape fast forward
b. move tape slow forward
c. move tape fast reverse
d. move tape slow forward
e. bring tape to load point when Data Cartridge installed
f. rewind tape to load point
g. rewind tape to BOT and eject cartridge in response to UNL command
h. rewind tape to BOT and eject cartridge in response to an AUTO UNL true condition.

NOTES:

1) The rewind and unload routines are self-completing and need no further implementation once started.
2) BOT, EOT, LP, are pulses.

3) The ROM output line pairs of FWRS and RVRS, and FASTS and SLOS are required to operate latches which remember tape speed and direction during deceleration.

4) Whenever a Data Cartridge is removed, a CIP false pulse is generated which temporarily resets the logic machine to STOP.

FPLA PROGRAM TABLE FOR IMPLEMENTING ALL STATE JUMPS IN THE FLOW-CHART

STATE JUMP	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0
		H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L
1-2	0	H	–	–	H	–	H	–	H	H	–	–	–	L	L	L	L	A	●	A	●	A	●	●	A
1-3	1	H	–	–	H	–	H	–	L	H	–	–	–	L	L	L	L	A	●	A	●	●	A	A	●
1-4	2	H	–	–	H	–	H	–	H	L	–	–	–	L	L	L	L	A	●	●	A	A	●	A	●
1-5	3	H	–	–	H	–	H	–	L	L	–	–	–	L	L	L	L	●	A	A	●	A	●	A	●
1-8	4	L	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L	A	●	●	A	A	●	●	A
1-9	5	H	–	–	H	H	–	–	–	–	–	–	–	L	L	L	L	A	●	●	A	●	A	●	A
1-12	6	H	–	–	H	–	–	–	–	–	–	–	H	L	L	L	L	●	A	A	●	●	A	●	A
2-1	7	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	A	●	A	●	A	●	A	●
2-6	8	–	–	–	–	–	–	H	–	–	–	–	–	L	L	L	H	A	●	A	●	●	A	●	A
3-1	9	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L	A	●	A	●	A	●	A	●
3-6	10	–	–	–	–	–	–	H	–	–	–	–	–	L	L	H	L	A	●	A	●	●	A	●	A
4-1	11	–	–	–	–	–	L	–	–	–	–	–	–	L	H	L	L	A	●	A	●	A	●	A	●
4-7	12	–	H	–	–	–	–	–	–	–	–	–	–	L	H	L	L	●	A	●	A	A	●	A	●
5-1	13	–	–	–	–	–	L	–	–	–	–	–	–	H	L	L	L	A	●	A	●	A	●	A	●
5-7	14	–	H	–	–	–	–	–	–	–	–	–	–	H	L	L	L	●	A	●	A	A	●	A	●
6-4	15	H	–	–	H	–	H	–	H	L	–	L	–	L	L	H	H	A	●	●	A	A	●	A	●
6-5	16	H	–	–	H	–	H	–	L	L	–	L	–	L	L	H	H	●	A	A	●	A	●	A	●
6-9	17	H	–	–	H	H	–	–	–	–	–	L	–	L	L	H	H	A	●	●	A	●	A	●	A
6-12	18	H	–	–	H	–	–	–	–	–	–	L	H	L	L	H	H	●	A	A	●	●	A	●	A
6-12	19	–	–	–	–	–	–	–	–	–	–	H	–	L	L	H	H	●	A	A	●	●	A	●	A
7-2	20	H	–	–	H	–	H	–	H	H	–	–	–	H	H	L	L	A	●	A	●	A	●	●	A
7-3	21	H	–	–	H	–	H	–	L	H	–	–	–	H	H	L	L	A	●	A	●	●	A	A	●
7-9	22	H	–	–	H	H	–	–	–	–	–	–	–	H	H	L	L	A	●	●	A	●	A	●	A
7-12	23	H	–	–	H	–	–	–	–	–	–	–	H	H	H	L	L	●	A	A	●	●	A	●	A
8-9	24	H	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	A	●	●	A	●	A	●	A
9-10	25	–	H	–	–	–	–	–	–	–	–	–	–	L	H	H	H	●	A	●	A	●	A	A	●
10-11	26	–	–	H	–	–	–	–	–	–	–	–	–	H	H	H	L	A	●	●	A	●	A	A	●
11-2	27	H	–	–	H	–	H	–	H	H	–	–	–	L	H	H	L	A	●	A	●	A	●	●	A
11-3	28	H	–	–	H	–	H	–	L	H	–	–	–	L	H	H	L	A	●	A	●	●	A	A	●
11-4	29	H	–	–	H	–	H	–	H	L	–	–	–	L	H	H	L	A	●	●	A	A	●	A	●
11-5	30	H	–	–	H	–	H	–	L	L	–	–	–	L	H	H	L	●	A	A	●	A	●	A	●
11-12	31	H	–	–	H	–	–	–	–	–	–	–	H	L	H	H	L	●	A	A	●	●	A	●	A
12-13	32	–	H	–	–	–	–	–	–	–	–	–	–	H	L	H	H	●	A	●	A	A	●	●	A
13-14	33	–	–	–	–	–	–	–	–	–	L	–	–	H	H	L	H	●	A	A	●	A	●	●	A
14-1	34	L	–	–	–	–	–	–	–	–	–	–	–	H	L	L	H	A	●	A	●	A	●	A	●
I/O ASSIGNMENT		CIP	BOT	LP	SEL	RWD	TR	EOT	FAST	FWD	TR EXT	AUTO UNL	UNL	O_4	O_3	O_2	O_1	K4	J4	K3	J3	K2	J2	K1	J1

Figure 4.

SERVO COMMANDS GENERATED BY DECODING STATE REGISTER WITH PROM AND GATES

INPUT					OUTPUT									
STATE														
#	CODE				TRS	FWDS	RVRS	FASTS	SLOS	EJECT	WRITE INH	BUSY 1	AT LP	AT EOT
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	1	0	0	0	1	1	0	1	0	0	1	0	0	0
3	0	1	0	0	1	1	0	0	1	0	0	0	0	0
4	0	0	1	0	1	0	1	1	0	0	1	0	0	0
5	0	0	0	1	1	0	1	0	1	0	1	0	0	0
6	1	1	0	0	0	0	0	0	0	0	1	0	0	1
7	0	0	1	1	0	0	0	0	0	0	1	0	0	0
8	1	0	1	0	0	0	0	0	0	0	1	0	0	0
9	1	1	1	0	1	0	1	1	0	0	1	1	0	0
10	0	1	1	1	1	1	0	0	1	0	1	1	0	0
11	0	1	1	0	0	0	0	0	0	0	1	0	1	0
12	1	1	0	1	1	0	1	1	0	0	1	1	0	0
13	1	0	1	1	0	0	0	0	0	0	1	0	0	0
14	1	0	0	1	0	0	0	0	0	1	1	0	0	0
PROM PIN #'s	10	11	12	13	1	2	3	4	5	6	7	8	AND GATE 1	AND GATE 2

Figure 5.

The circuit shown in Figure 1 is a sequential controller which implements all functions needed to control a dot matrix serial printer, and provides a simple, inexpensive, and reliable control circuit useful in many design environments.

The controller consists of an 82S100 FPLA as a program decoder, and two 74175 quad D-type flip-flops as feedback elements.

Parallel ASCII data is presented to the inputs of the FPLA along with an associated Data Strobe. If data detected by the FPLA is a printable character (as opposed to a control function) it will be entered into the buffer memory.

The buffer memory consists of two 3351's configured as an 80 x 9 F1FO, and will hold one full line of 80 characters.

When the memory is full, or if the FPLA detects an ASCII "carriage return," the controller will begin a print cycle, and print out the data stored in memory. The first character in memory will be presented to the inputs of the MK2302P Character Generator. The controller will strobe the Output Enable of the character generator, increment the column counter clock, step the print head, test the Counter Output of the character generator, and shift the next piece of data from the memory if the Counter Output is true. This sequence will continue until the memory is empty, as signalled by the Output Ready line of the second 3351.

The controller will then generate Carriage Return, Line Feed, and Ribbon Advance signals via the 7442 decoder. The print cycle is then complete and the controller awaits new data.

SEQUENTIAL CONTROLLER FOR
SERIAL IMPACT PRINTER

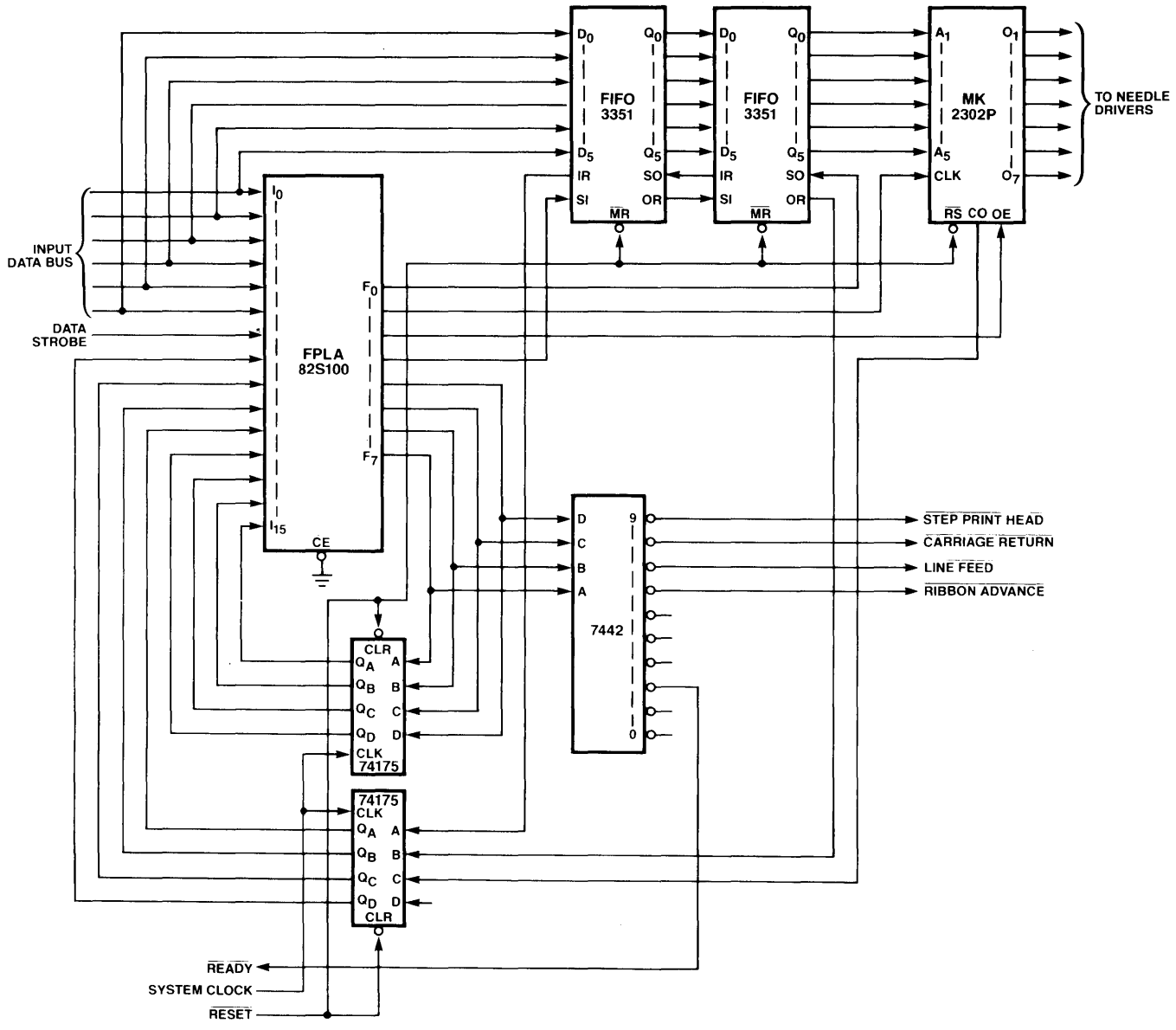


Figure 1.

In industrial or commercial processes it is often necessary to provide many contact closure type control points and/or contact sense points. A simple control system of this type can be implemented using the circuit of Figure 1, and any serial compatible send/receive terminal such as a Teletype or CRT controller.

The circuit has the following features:

A. Up to 56 jumper selectable addressed units may be connected in series to a single control circuit. Address are in octal, excluding 00,11,22,33,44,55,66,77.
B. Up to 16 contact closure output points (TTL level) for each unit. Points A thru P functions ("S"et and "R"eset).
C. Up to 16 contact closure input points (TTL level) for each unit. Points A thru P function ("?" Read).

The serial IN(SI) and serial OUT(SO) drivers are not defined in this circuit. A typical output communications sequence is (#34AS), where:

"#" = Attention character that resets the address function in all units on the serial communications circuit.
"3" = The 1st address character
"4" = The 2nd address character
"A" = The control point
"S" = The control function

When sensing a contact point, replace the "R" or "S" function with a "?" function to read the sense point. The unit responds (in ASCII) with a "1" for a closed contact and a "0" for an open contact point.

Several different modifications of this basic circuit are possible to send BCD or ASCII data using the same or a modified decoding scheme. The decoding function for this circuit is shown in the FPLA program table of Figure 2.

DISTRIBUTED PROCESS CONTROLLER

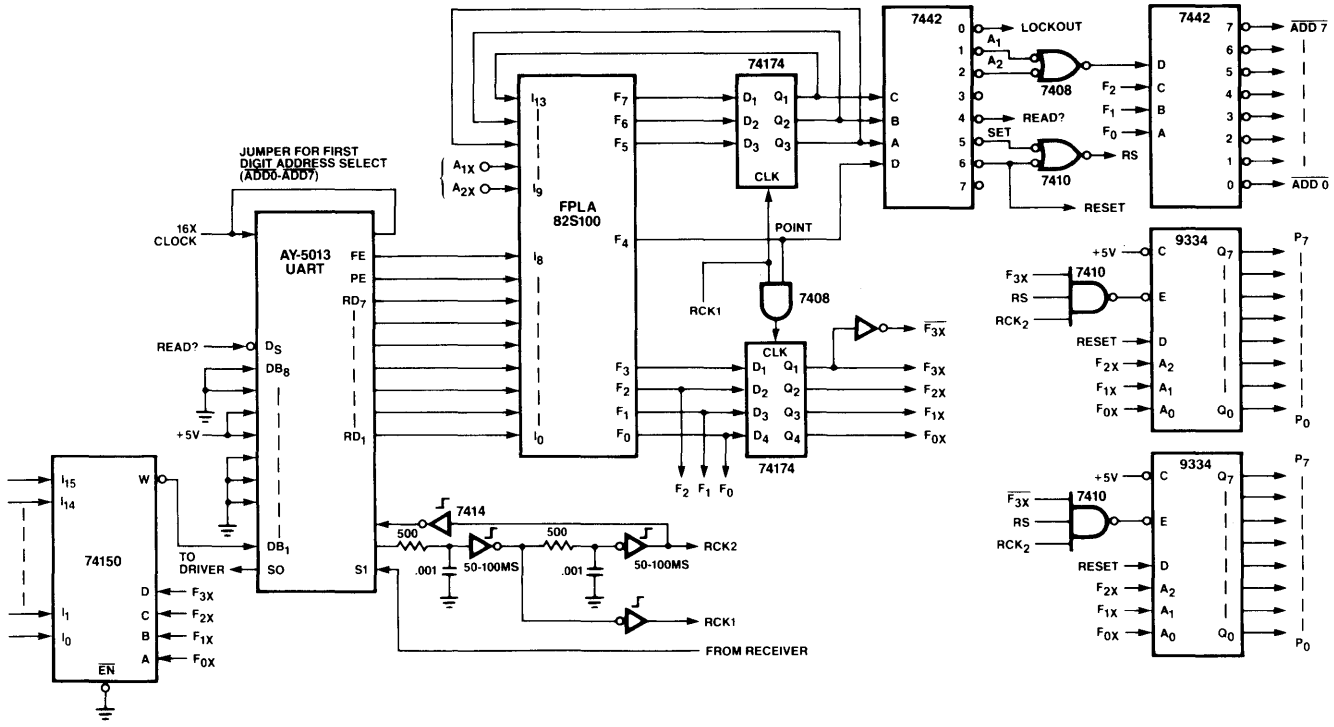


Figure 1.

FPLA PROGRAM TABLE

COMMENT	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: L L L H H H H H / OUTPUT FUNCTION 7	6	5	4	3	2	1	0
#	0	–	–	–	–	–	–	–	L	L	L	H	L	L	L	H	H	A	A	•	•	•	•	•	•
TRANS. ERR.	1	–	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
A1 ERR.	2	–	–	L	L	H	–	L	–	–	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
A2 ERR.	3	–	–	L	H	L	L	–	–	–	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
"S" ET	4	–	–	H	–	–	H	H	L	L	L	H	H	L	L	H	H	•	A	•	•	•	•	•	•
"R" ESET	5	–	–	H	–	–	H	H	L	L	L	H	H	L	L	H	L	•	•	A	•	•	•	•	•
READ "?"	6	–	–	H	–	–	H	H	L	L	H	H	H	H	H	H	H	•	A	A	•	•	•	•	•
A	7	–	–	H	–	–	H	H	L	L	L	L	L	L	L	L	H	•	•	•	A	•	•	•	A
B	8	–	–	H	–	–	H	H	L	L	L	L	L	L	L	H	L	•	•	•	A	•	•	A	•
C	9	–	–	H	–	–	H	H	L	L	L	L	L	L	L	H	H	•	•	•	A	•	•	A	A
D	10	–	–	H	–	–	H	H	L	L	L	L	L	L	H	L	L	•	•	•	A	•	A	•	•
E	11	–	–	H	–	–	H	H	L	L	L	L	L	L	H	L	H	•	•	•	A	•	A	•	A
F	12	–	–	H	–	–	H	H	L	L	L	L	L	L	H	H	L	•	•	•	A	•	A	A	•
G	13	–	–	H	–	–	H	H	L	L	L	L	L	L	H	H	H	•	•	•	A	•	A	A	A
H	14	–	–	H	–	–	H	H	L	L	L	L	L	H	L	L	L	•	•	•	A	A	•	•	•
I	15	–	–	H	–	–	H	H	L	L	L	L	L	H	L	L	H	•	•	•	A	A	•	•	A
J	16	–	–	H	–	–	H	H	L	L	L	L	L	H	L	H	L	•	•	•	A	A	•	A	•
K	17	–	–	H	–	–	H	H	L	L	L	L	L	H	L	H	H	•	•	•	A	A	•	A	A
L	18	–	–	H	–	–	H	H	L	L	L	L	L	H	H	L	L	•	•	•	A	A	A	•	•
M	19	–	–	H	–	–	H	H	L	L	L	L	L	H	H	L	H	•	•	•	A	A	A	•	A
N	20	–	–	H	–	–	H	H	L	L	L	L	L	H	H	H	L	•	•	•	A	A	A	A	•
O	21	–	–	H	–	–	H	H	L	L	L	L	L	H	H	H	H	•	•	•	A	A	A	A	A
P	22	–	–	H	–	–	H	H	L	L	L	H	H	L	L	L	L	•	•	•	A	•	•	•	•
I CHARACTER ADDRESS DECODE	23	–	–	L	L	H	–	H	L	L	L	H	H	L	L	L	L	•	•	•	•	•	•	•	•
	24	–	–	L	L	H	–	H	L	L	L	H	H	L	L	L	H	A	•	A	•	•	•	•	A
	25	–	–	L	L	H	–	H	L	L	L	H	H	L	L	H	L	A	•	A	•	•	•	A	•
	26	–	–	L	L	H	–	H	L	L	L	H	H	L	L	H	H	A	•	A	•	•	•	A	A
	27	–	–	L	L	H	–	H	L	L	L	H	H	L	H	L	L	A	•	A	•	•	A	•	•
	28	–	–	L	L	H	–	H	L	L	L	H	H	L	H	L	H	A	•	A	•	•	A	•	A
	29	–	–	L	L	H	–	H	L	L	L	H	H	L	H	H	L	A	•	A	•	•	A	A	•
	30	–	–	L	L	H	–	H	L	L	L	H	H	L	H	H	H	A	•	A	•	•	A	A	A
II CHARACTER ADDRESS DECODE	31	–	–	L	H	L	H	–	L	L	L	H	H	L	L	L	L	•	A	A	•	•	•	•	•
	32	–	–	L	H	L	H	–	L	L	L	H	H	L	L	L	H	•	A	A	•	•	•	•	A
	33	–	–	L	H	L	H	–	L	L	L	H	H	L	L	H	L	•	A	A	•	•	•	A	•
	34	–	–	L	H	L	H	–	L	L	L	H	H	L	L	H	H	•	A	A	•	•	•	A	A
	35	–	–	L	H	L	H	–	L	L	L	H	H	L	H	L	L	•	A	A	•	•	A	•	•
	36	–	–	L	H	L	H	–	L	L	L	H	H	L	H	L	H	•	A	A	•	•	A	•	A
	37	–	–	L	H	L	H	–	L	L	L	H	H	L	H	H	L	•	A	A	•	•	A	A	•
	38	–	–	L	H	L	H	–	L	L	L	H	H	L	H	H	H	•	A	A	•	•	A	A	A
ILLEGAL CODES	39	–	–	–	–	–	–	–	–	L	L	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	40	–	–	–	–	–	–	–	–	L	L	H	–	–	–	–	–	A	A	A	•	•	•	•	•
	41	–	–	–	–	–	–	–	–	L	H	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	42	–	–	–	–	–	–	–	–	H	H	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	43	–	–	–	–	–	–	–	–	H	H	H	–	–	–	–	–	A	A	A	•	•	•	•	•

Figure 2.

The output states of solenoid-controlled, air-operated process valves, are a function of logical combinations of input variables, which are gated by contacts to signify various process modes, sequence steps, other valve positions, flow, temperature, pressure, level, etc.

Occasionally it is desirable to change the function or equation of process valves. The FPLA provides an elegant yet cost effective solution for reprogramming these valves and/or other process devices. In the circuit of Figure 1, the set of 4N2B couplers senses the status of input contacts and will command the 82S101 to operate the valve when the proper logical combination of inputs occurs. The PS101 will energize the solenoid valve when the output of the FPLA goes LOW.

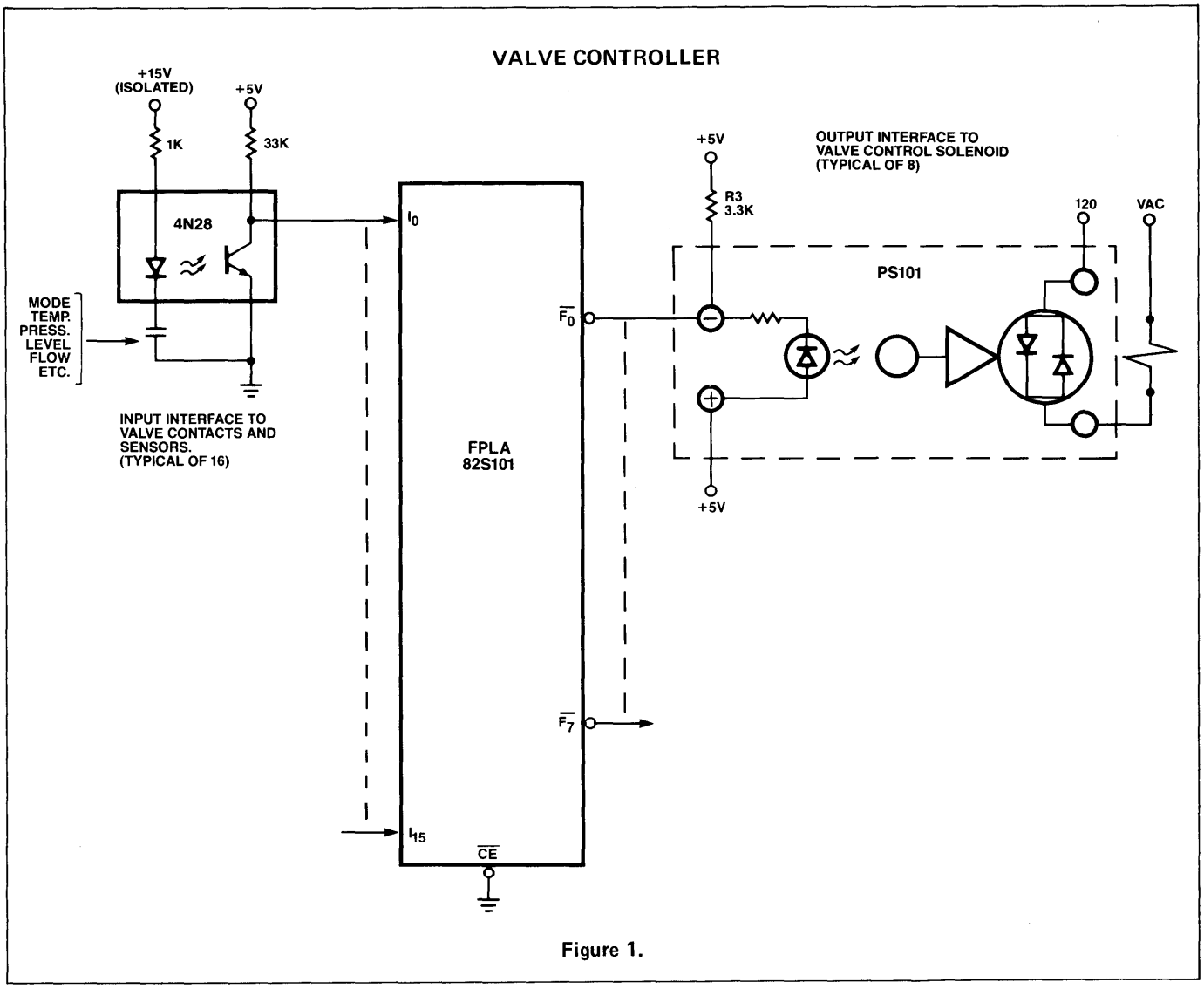


Figure 1.

This controller breaks the time of traffic flow into the four basic parts as shown in Figure 1. Notice that each crossroad is assigned a traffic light during each time frame, and that each time frame has a certain length in seconds. As shown

TIMING OF TRAFFIC LIGHTS

TRAFFIC = 1	GREEN	YELLOW	RED	RED
TRAFFIC = 2	RED	RED	GREEN	YELLOW
TIME	45 SEC.	15 SEC.	45 SEC.	15 SEC.
STATE	00	01	10	11

Figure 1.

in Figure 2, in order for the circuit to recognize the separate states, each state is assigned a state variable to be stored by FFI. The other flip-flops (#2 thru #4) will store the output information for the traffic lights.

Because of the limited number of outputs, the information is time multiplexed out of the FPLA over a 2 bit bus.

The two 7493's, and the 555 timer form a binary counter for counting up to 64 second in 0.5 second intervals.

The FPLA determines the state of the machine by I_{16} and I_{15} and the time elapsed by I_7 thru I_0. When the proper input conditions are met, the 7493 counter is reset, the state in FFI is updated, and the rest of the flip-flops are updated as needed. The time counter continues until a new change in state is indicated by the state of the inputs. At this time the process is repeated.

By programming the FPLA the designer can determine the length of the time segments, and the lights to be activated. Since several inputs are still available there is room to add vehicles and/or people sensors which would instruct the FPLA to change state and continue from there.

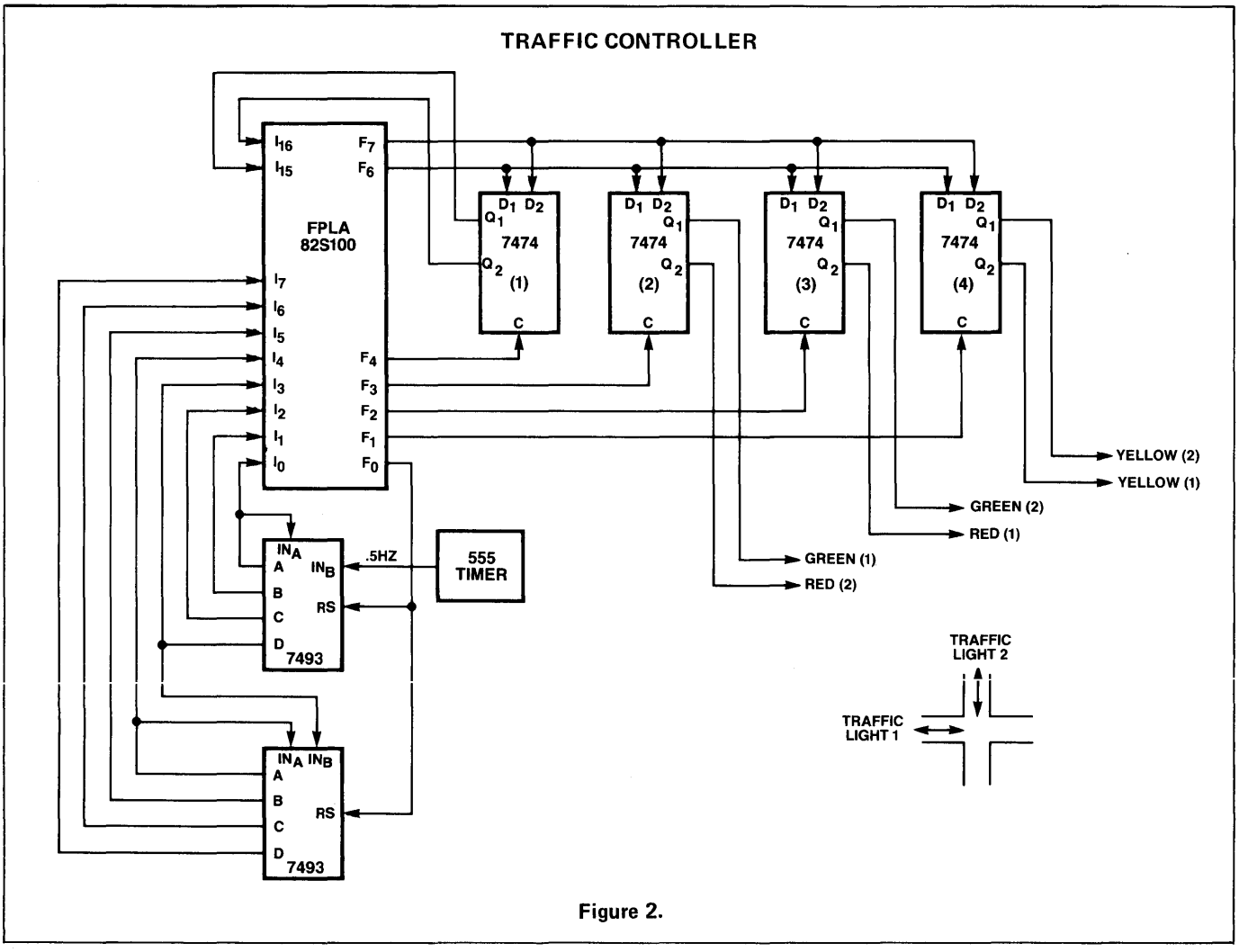


Figure 2.

In the circuit of Figure 1 an 82S101 FPLA is used together with a few other components to build a complete sequential control system on a single 5" x 7" PC card. In this example the circuit is used as a machine controller which recognizes specific starting conditions on its TTL and 120 VAC inputs, and drives the 120 VAC air solenoids in a predetermined sequence, interrupted by time delays as needed. Position feedback from limit switches is used to ensure that each step in the sequence is properly completed, and that the FPLA can detect jams and abort the sequence if any step requires too much time for completion. Logic level outputs are available for such functions as resetting external counters or starting A/D converters.

The system has six inputs and five outputs for interfacing with the machine. In this case, three inputs and two outputs are TTL-compatible, while the remainder are used with 120 VAC. The FPLA analyzes the inputs and internal states of the system, and uses this information to control the desired status of each output, the operation of the time delay circuit, and the advancement of the step counter. Resistor LED indicators can be used for troubleshooting or status checking, with some being driven directly by the FPLA. All 120 VAC interfaces are optically isolated. The 120 VAC inputs come into the system through AC input modules, which provide TTL compatible signals and also drive LED status indicators. These inputs are coupled through debouncing gates before being taken to the FPLA. One of the debouncer sections, with a capacitor tied to its input, provides a startup delay signal, and the debouncer clock is used as the system clock.

An integrated counter/decoder/latch/display provides the sequential step number. The FPLA can either increment or reset the count, and the step number is made available both as a 7 segment display and as BCD data fed back to the FPLA. Time delays are generated by combining a programmable timer/counter with the FPLA. Some of the timer outputs are taken to the FPLA which combines them as required to build time delays of different lengths (50 ms to 12 sec, in this case).

The logic level outputs of the system are taken directly from the FPLA. Outputs to 120 VAC devices are provided by DIP solid state relays driven by the FPLA. LED indicators may be driven by the same FPLA outputs, to show which AC outputs are on. Fuses and varistors for protecting the AC outputs are also mounted on the card.

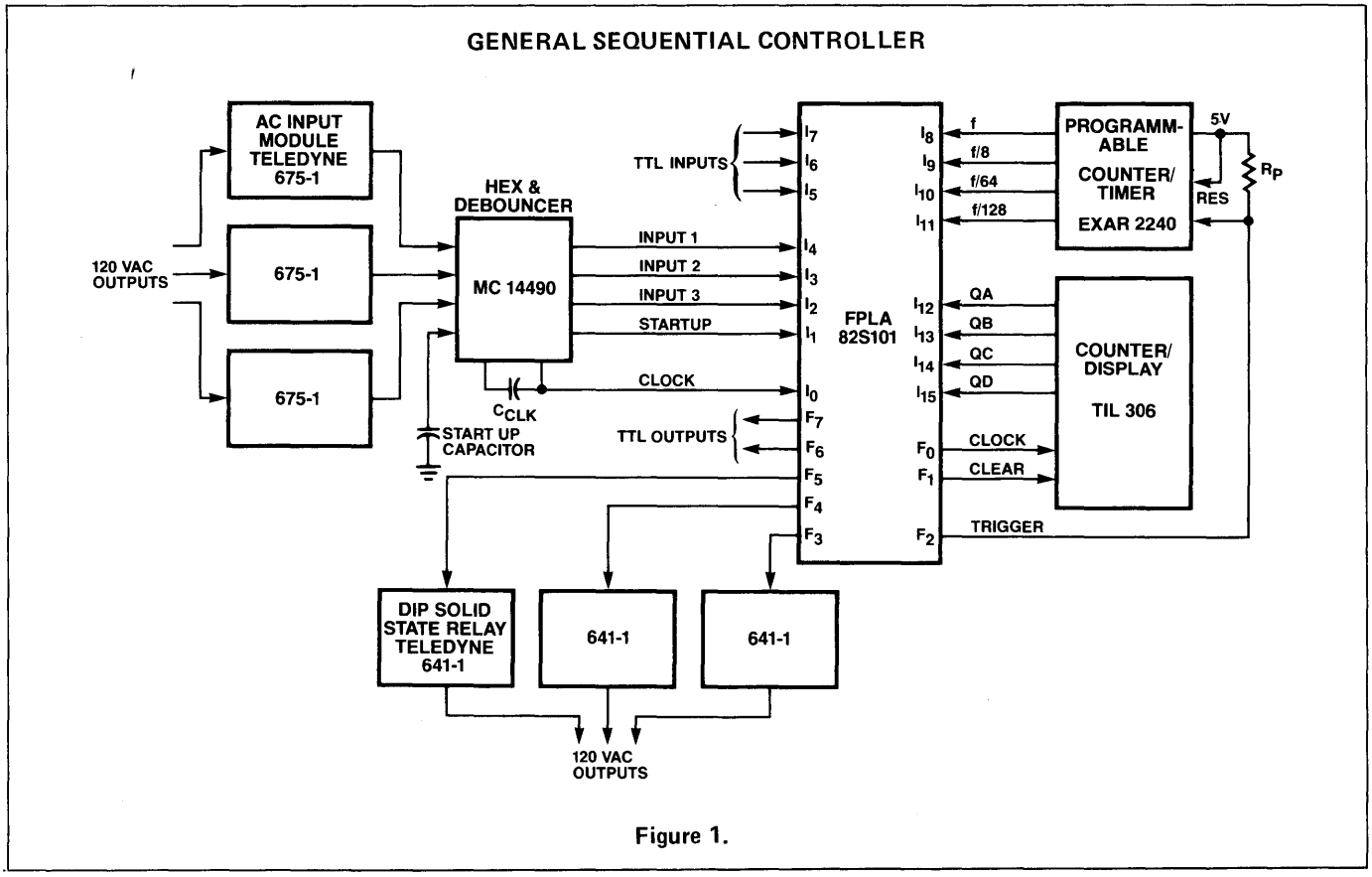


Figure 1.

It is common practice to provide redundant signal paths in todays high reliability military communication systems in order to ensure adequate mission availability. In these systems, it is necessary to monitor the performance of the various subsystems' display status in a concise manner, and provide controls for switching over to standby equipment in either manual or automatic modes. Customarily, a unique System Status and Control Logic Unit (SSCLU) is designed for a system which integrates these control and status functions at a central console. Random T^2L Logic, mounted on a rack full of wire wrap boards, is typically used to implement the SSCLU functions. This approach is inherently costly, inflexible, and unreliable.

The introduction of the Field Programmable Logic Array (FPLA) provides a powerful tool which makes possible the general purpose "n" Expandable SSCLU circuit card set. These two unique circuit card types are all that is normally required to realize an SSCLU. The first circuit card type, shown in Figure 1, is referred to as Subsystem Circuit Card. The second, shown in Figure 2, is designated the Summary Circuit Card. One Subsystem Circuit Card is normally required to monitor and configure each of the subsystems making up a total communication system. Thus, a Satellite Communication Terminal can be divided into the

BLOCK DIAGRAM OF SUBSYSTEM CIRCUIT CARD

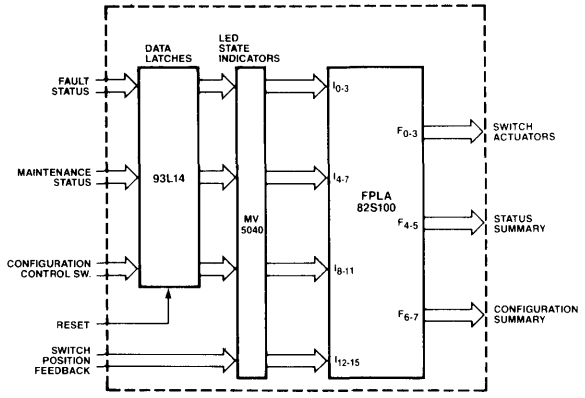


Figure 1.

following dual redundant subsystems: X-Band Downlink, X-Band Uplink, Tracking, and Up/Down Conversion. Each of these subsystems can normally be monitored and controlled by one Subsystem Circuit Card. Two or more simple subsystems could be accommodated by a single Subsystem Circuit Card, or conversely, two cards could possibly be required to service one complex subsystem. The function of the Subsystem Circuit Card is three-fold. It displays the incoming subsystem status and configuration data, reduces the data to develop configuration and status summary messages, and generates control signals to configure the subsystems. The operation of the Subsystem Circuit Card is straightforward. As shown in Figure 1, digital status, configuration and control data, which is normally in the form of contact closures from various subsystem fault sensors, enters the circuit card where it is latched with the exception of the configuration switch feedback signals. The switch feedback is routed directly to the FPLA inputs. The latches are all cleared simultaneously through the reset bus. All inputs to the FPLA have LED state indicators which constantly display the data being fed into it. This feature allows immediate verification of status and facilitates rapid fault isolation. The FPLA scans its input data and generates 3 output data groups based on its micro-program. Its first four outputs activate solid state relays which in turn result in configuration switching of the subsystems equipment. The next two FPLA outputs contain a 2-bit binary code which indicates the subsystem status summary as tabulated in Figure 3. The final two outputs carry a second 2-bit binary code representing the configuration of the subsystem also tabulated in Figure 3.

The SSCLU Summary Circuit Card is illustrated in Figure 2. Its primary purpose is to consolidate and interpret the 2-bit binary status and configuration data from up to 4 Subsystem Circuit Cards. It then generates the outputs based on its micro-program to drive visual and aural system level status and configuration indicators. The inputs to both of its FPLAs are paralleled, and all 16 of them have LED input state indicators. Each FPLA output feeds an open collector lamp driver. Based on the monitoring requirements of the particular system, the outputs of both FPLAs could also be paralleled, resulting in only eight status outputs, but allowing up to 96 product terms to be micro-programmed per Summary Circuit Card.

BLOCK DIAGRAM OF SUMMARY CIRCUIT CARD

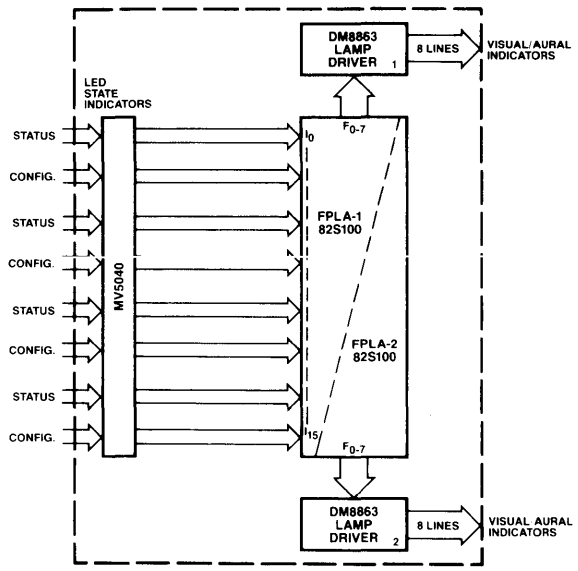


Figure 2.

STATUS AND CONFIGURATION SUMMARY CODE EXAMPLES

	CODE	EQUIPMENT ON LINE	COMMENTS
CONFIGURATION CODE	00	PARAMP. I + IFLA I	NORMAL DOWN LINK I
	11	PARAMP. II + IFLA II	NORMAL DOWN LINK II
	10	PARAMP. I + IFLA II	DOWNLINK I/II CROSS PATCH
	01	PARAMP. II + IFLA I	DOWNLINK I/II CROSS PATCH

	CODE	MEANING
SUMMARY CODE	00	DN LINK FULLY AVAILABLE
	10	DN LINK MINOR FAULT
	01	DN LINK MAJOR FAULT
	11	DN LINK SWITCH CONFIRMATION FAULT

Figure 3.

IGNITION TIMING FOR I/C ENGINE USING MULTI-DIMENSIONAL CURVE FIT 8

The diagram of Figure 1 shows a system for optimizing ignition timing of an internal combustion engine in terms of engine speed, temperature, and manifold vacuum. The greatest contribution to the correct spark advance comes from the speed of the engine. For this measurement two clocks are needed: a constant time base from a crystal oscillator and a full cycle from the output of a finely slotted disc in the distributor.

Next is the vacuum at the intake manifold. This is derived from a pressure transducer and an analog to digital converter. Only six bits of resolution should be necessary. It would help keep the cost down and the conversion time below the 20 microsecond limit that would be encountered at high engine speeds and with a 360° slotted angular clock source.

A third transducer measuring coolant temperature uses comparators to look at only one warm-up setting and one threshold at which the engine has overheated to the extent it should be shut down.

Finally a strobe, PISTON CLK, is provided for system syn-

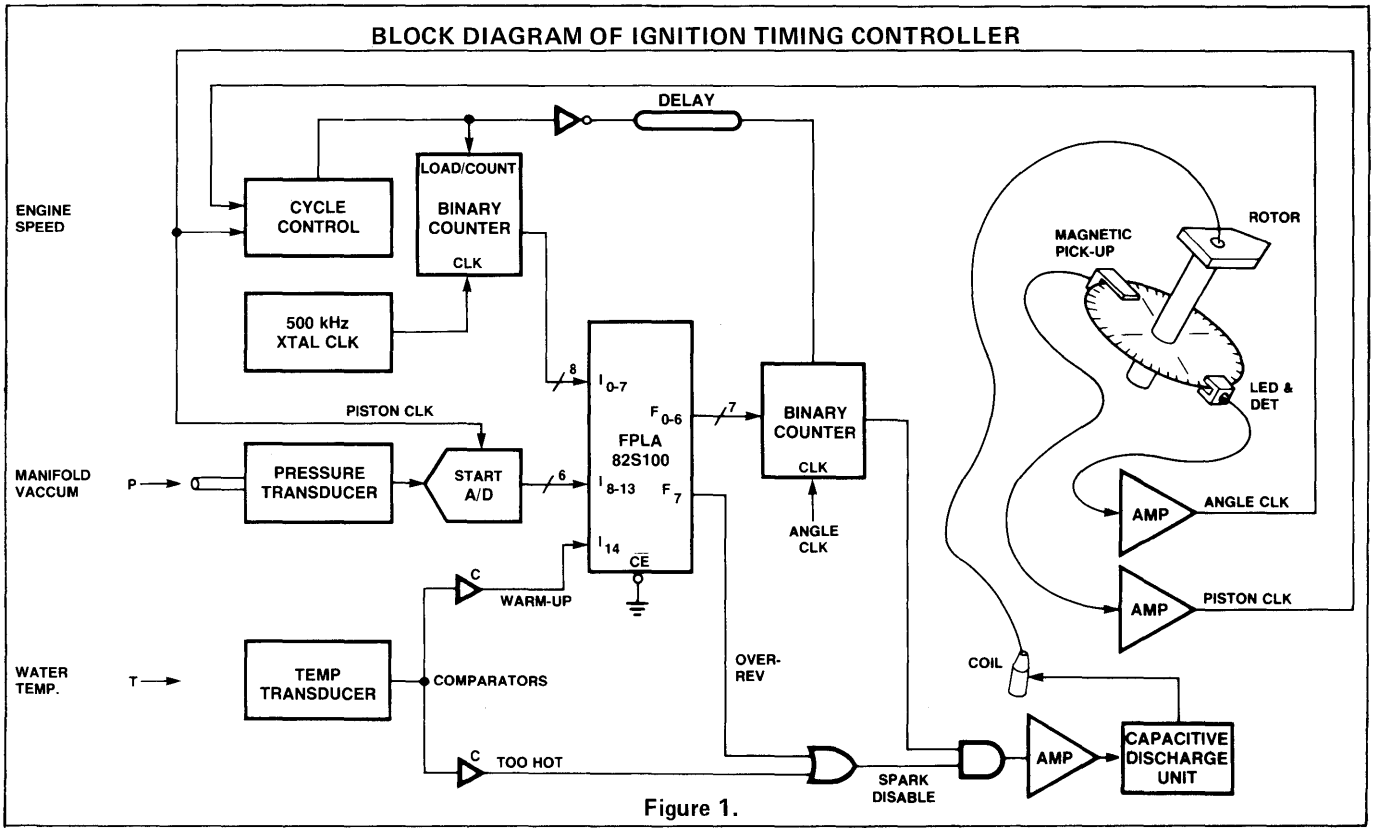


Figure 1.

IGNITION TIMING FOR I/C ENGINE USING MULTI-DIMENSIONAL CURVE FIT 8

chronization by adding a magnetic pick-up to the angular clock disc at a point just ahead of the maximum advance angle. This type of sensor would allow us to program for contingency operation in the event dirt fouled the LED source or detector.

FPLA computed spark advance values are loaded into the output counter via a short delay element as soon as the engine speed has been determined. At the same time, the angular clock input is enabled and the counter is incremented to its maximum value. This will trigger a capacitive discharge unit which in turn ignites the mixture in the combustion chamber. A gate has been added to disable the spark under FPLA determined stress conditions such as excessive engine speed.

Although there are only 48 terms available in the 82S100, this should be more than adequate resolution for an automotive ignition. Effective resolution of input and output values, however, are not limited to one part in 48. Because of functional non-linearities and the involvement of more than one variable input dimension, there may be numerous paths to any of the 48 solutions. An increasing input slope will have a decreasing effect on an output and lower order bits can be programmed to a "Don't Care" state. Furthermore, as one input parameter becomes more dominant in determining output, the significance of the other may be diminished with the same technique.

While working out hypothetical solutions, it became apparent that it is preferable to keep input values in direct proportion to output. This way fewer bits are needed to define a new range where a new set of "Don't Care" positions can be programmed. This rule was violated only in the tachometer circuit in favor of response time. Here, several bit positions were necessary to determine the small time interval that represents an over-reved engine.

USING FPLAs AND SRGs TO STORE HUFFMAN CODES 9

In modern serial data transmission systems, the thruput of a communications channel can be improved by using data compression techniques to reduce data redundancies. One method, developed by Huffman, encodes the set of symbols required to convey various messages into a variable length code set. The frequency of use of each symbol dictates the length of the code that is to be associated with that symbol.

Typically, variable length codes are stored in fixed length read only memory look-up tables with a delimiter bit to allow stripping away the extraneous bits stored with the codes. The original uncompressed symbols are then used to address the look-up table as each symbol occurs in the message. As each code is retrieved from the table, some special equipment must remove extraneous bits and compact the codes into byte size chunks for processing by a device such as a USART.

The circuit described here shows an alternative method of storing, retrieving and compacting variable length Huffman codes. This method uses some of the properties of Huffman codes and the properties of maximal length shift register generators (SRGs) to implement an addressable, variable length, storage device.

A typical but fictitious Huffman code set of 34 codes used to demonstrate the circuit implementation is shown in Figure 1. The codes, as shown, must be transmitted left-most bit first. The reason for this leads to a very important property of Huffman codes: no code may be coded in such a way to appear, digit for digit, as the first part of any other code of greater length. This characteristic allows a receiver

TYPICAL HUFFMAN CODE SET

SYMBOL #	LENGTH	CODE	SYMBOL #	LENGTH	CODE
0	2	00	17	7	0100111
1	3	101	18	7	0111010
2	3	110	19	7	0111011
3	4	0101	20	7	0111111
4	4	0110	21	7	0111110
5	4	1000	22	7	1001000
6	5	10011	23	7	1001001
7	5	11101	24	7	1001010
8	6	010001	25	7	1001011
9	6	010010	26	7	1110000
10	6	011100	27	7	1100001
11	6	011110	28	7	1110000
12	6	111110	29	7	1110001
13	6	111100	30	7	1111111
14	6	111101	31	7	1111110
15	7	0100001	32	8	01000000
16	7	0100110	33	8	01000001

Figure 1.

to uniquely decode an incoming message made up of a continuous stream of serial data by knowing only the code set and the starting point of the message. Two other properties of Huffman codes concern each possible sequence of digits of length one less than the longest code in the set, so that they must 1) be used either as a code, or must have one of its prefixes used as a code and that 2) every binary Huffman code set must have exactly two codes of length L, where L is the length of the longest code in the set.

Linear sequential shift register generators (SRGs) are a special class of non-binary counters whose operation can be described and analyzed by application of linear algebra techniques. These counters are made up of a number of delay stages which, except for the first stage, each receive as their input the output from the previous stage (just as

ordinary shift registers). The input to the first stage is derived from a Modulo-2 addition (EX-or function) of selected outputs of the delay stages. These counters repeatedly cycle through a defined sequence of states that are a function of the feedback used. The SRGs used here have a maximal length sequence of states, and thus generate all 2^k-1 states required for Huffman code sets, where k is the number of delay stages. The "all-zero" state is always missing in these counters since the exclusive-or feedback would cause the counter to remain in this state forever.

Since the longest code in Figure 1 is 8 bits long, an 8 stage maximal length SRG will be used in the generation of the code set, since only the last two codes are this long. The other codes in the set will be made up from combinations of less than 8 digits. Every combination of less than 8 digits (2^n, where n = 1, 2, 3, 4, 5, 6, 7) is available from an 8 stage maximal length SRG.

The maximal length SRG used here is defined by $I(X) = X^8 + X^6 + X^5 + X^4 + 1$, which is the characteristic polynominal of an SRG with the outputs of the 8^{th}, 4^{th}, 3^{rd}, and 2^{nd} stages summed Modulo-2 and fed back to the 0^{th} stage, the input. The connections for any SRG can be found directly from the characteristic polynomial associated with that SRG by subtracting from the order k of the polynomial each power of X present in the polynomial.

The selected SRG will produce all the sequences necessary to make up the code set; so all that is needed is circuitry to allow addressing the SRG to access the codes. FPLAs are ideally suited for this application since they can be used as programmable address decoders. This can be accomplished by using code addresses in combination with the outputs of each delay stage of the SRG as inputs to the FPLA.

In the circuit diagram of Figure 2, the SRG (8,4,3,2,0) is made up of U3, U4, U8-A, U8-B, and U8-C and is normally sequencing through its 255 states. An input code address latch is provided by U1, which could be an output port of a

VARIABLE LENGTH STORAGE SYSTEM FOR A TYPICAL HUFFMAN CODE SET

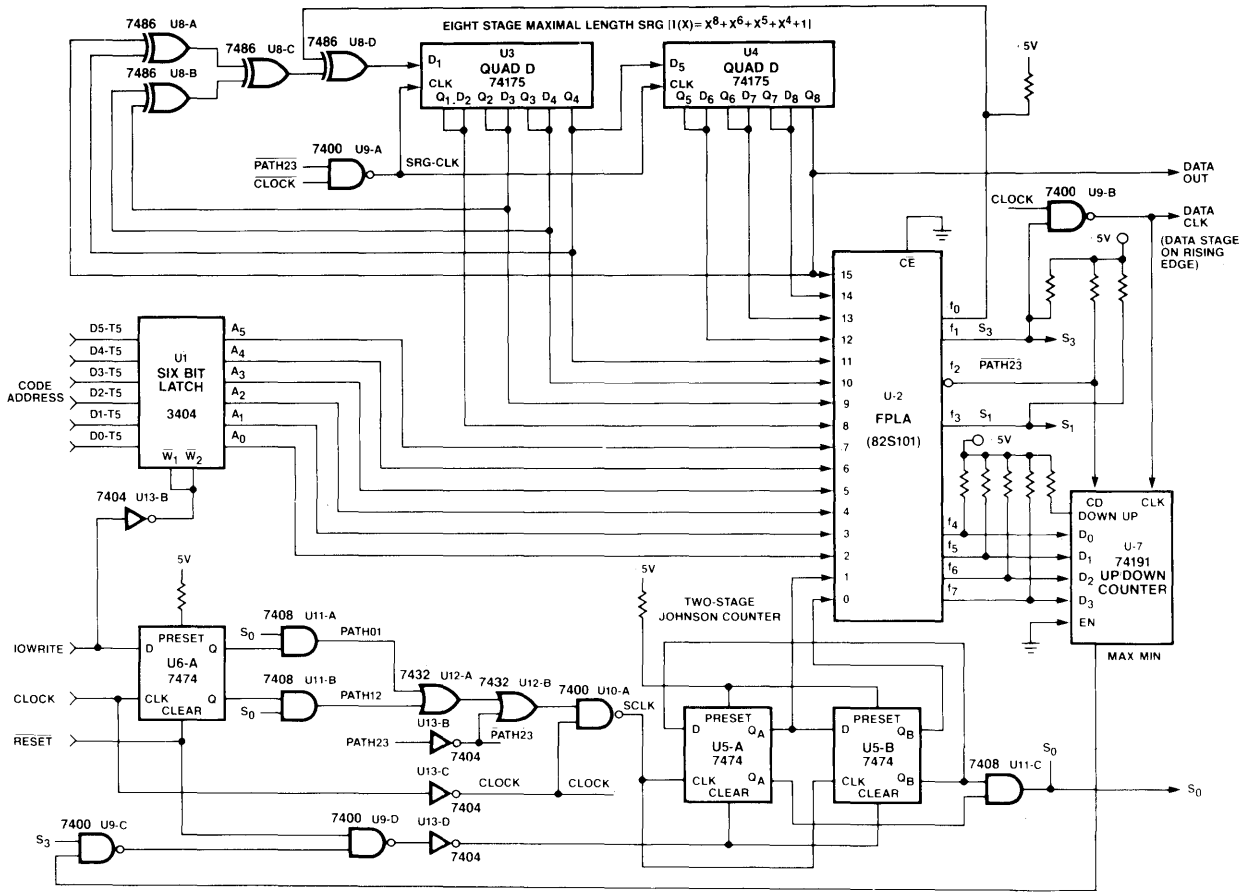


Figure 2.

processor controlled system where the circuitry described here serves as a peripheral. An indication of the latch availability is provided with the output S_0. The FPLA is used for address decoding and some state control functions. The D flip-flop, U6-A, serves to re-clock the "IOWRITE" signal for use by the state controller, which is implemented with a 2-stage Johnson counter by U-5. The up/down counter U-7 is used in conjunction with the FPLA to provide the proper number of data out clocks for each code addressed. With reference to Figure 3, initially the system is in state S_0, and is waiting to be addressed. The S_0 output could be used in either an interrupt driven or a polled system to determine when the circuit is ready for the next code address. When the 6 bit input address latch is serviced the "IOWRITE" signal goes HIGH and is latched by U6-A. This causes the state controller to advance to S_1 on the next system clock. State S_1 is a do nothing state that is just waiting for "IOWRITE" to return back to a logic "0" condition, indicating that the address in the input latch is not stable. When "IOWRITE" goes to logic "0", $\bar{Q}$ of U6-A will go HIGH on the next system clock, enabling the state counter to advance to S_2 on the next clock to find the code that has been addressed.

Since the SRG is free running through its 255 states, the Q_8 through Q_1 inputs to the FPLA are continuously changing. The FPLA is programmed in such a way that it looks for a match between the desired code (the input latch address) and the condition of the last L bits of the SRG, where L is the length of the desired code. For example, as tabulated in Figure 4, the address of code #0 is $\bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4\bar{A}_5$ and code #0 is 00, so the FPLA contains the Product Term ($\bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4\bar{A}_5\bar{Q}_8\bar{Q}_7\bar{Q}_A\bar{Q}_B$). The Q_A and Q_B factors serve to ensure that this condition is met only in S_2. When a match is found, the FPLA outputs $\overline{Path23}$ and a code length word of f_4, f_5, f_6, and f_7. The signal $\overline{Path23}$ does three things: 1) loads the code length word into the up/down counter, 2) disables the clock to the SRG so the desired code is held in the SRG, 3) enables the state counter to advance to S_3. With the system in S_3, the data clock is enabled and the SRG clock is re-enabled. The addressed code can now be collected by a device such as a high speed synchronous receiver tied to the data and data clock lines as the data is shifted out of the SRG in S_3. Notice that the data clock is also connected to the clock input of the up/down counter. This will allow shifting out the L bits corresponding to the length of the code addressed. When all bits of the code have been shifted out the max/min signal from the up/down counter will go HIGH and reset the state counter back to S_0 to wait for the next code address.

The access time for the codes is directly proportional to the code length since the combination of bits to make the shorter codes occur more often in the 255 bit long SRG sequence.

Longer codes could be accommodated with the use of longer SRGs. Longer code sets could also be implemented using this method by using more FPLAs.

STATE CONTROLLER TIMING DIAGRAM

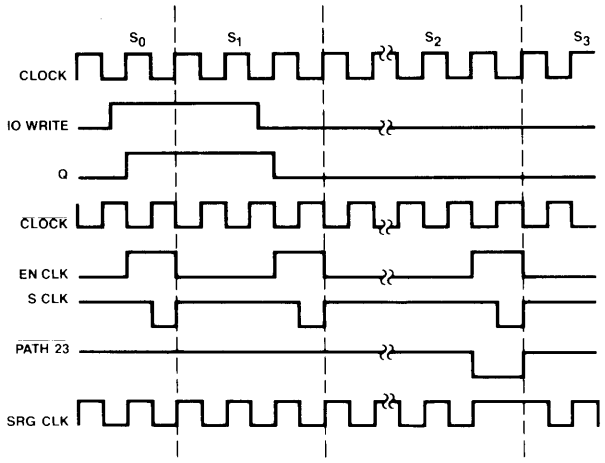


Figure 3.

FPLA PROGRAM TABLE FOR HUFFMAN CODE SET AND STATE CONTROL FUNCTIONS

PRODUCT TERM																	ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	L	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	L	L	–	–	–	–	–	–	L	L	L	L	L	L	H	H	•	•	A	•	•	A	•	•
1	H	L	H	–	–	–	–	–	L	L	L	L	L	H	H	H	•	•	A	A	•	A	•	•
2	H	H	L	–	–	–	–	–	L	L	L	L	H	L	H	H	•	•	A	A	•	A	•	•
3	L	H	L	H	–	–	–	–	L	L	L	L	H	H	H	H	•	A	•	•	•	A	•	•
4	L	H	H	L	–	–	–	–	L	L	L	H	L	L	H	H	•	A	•	•	•	A	•	•
5	H	L	L	L	–	–	–	–	L	L	L	H	L	H	H	H	•	A	•	•	•	A	•	•
6	H	L	L	H	H	–	–	–	L	L	L	H	H	L	H	H	•	A	•	A	•	A	•	•
7	H	H	H	L	H	–	–	–	L	L	L	H	H	H	H	H	•	A	•	A	•	A	•	•
8	L	H	L	L	L	H	–	–	L	L	H	L	L	L	H	H	•	A	A	•	•	A	•	•
9	L	H	L	L	H	L	–	–	L	L	H	L	L	H	H	H	•	A	A	•	•	A	•	•
10	L	H	H	H	L	L	–	–	L	L	H	L	H	L	H	H	•	A	A	•	•	A	•	•
11	L	H	H	H	H	L	–	–	L	L	H	L	H	H	H	H	•	A	A	•	•	A	•	•
12	H	H	H	H	H	L	–	–	L	L	H	H	L	L	H	H	•	A	A	•	•	A	•	•
13	H	H	H	H	L	L	–	–	L	L	H	H	L	H	H	H	•	A	A	•	•	A	•	•
14	H	H	H	H	L	H	–	–	L	L	H	H	H	L	H	H	•	A	A	•	•	A	•	•
15	L	H	L	L	L	L	H	–	L	L	H	H	H	H	H	H	•	A	A	A	•	A	•	•
16	L	H	L	L	H	H	L	–	L	H	L	L	L	L	H	H	•	A	A	A	•	A	•	•
17	L	H	L	L	H	H	H	–	L	H	L	L	L	H	H	H	•	A	A	A	•	A	•	•
18	L	H	H	H	L	H	L	–	L	H	L	L	H	L	H	H	•	A	A	A	•	A	•	•
19	L	H	H	H	L	H	H	–	L	H	L	L	H	H	H	H	•	A	A	A	•	A	•	•
20	L	H	H	H	H	H	H	–	L	H	L	H	L	L	H	H	•	A	A	A	•	A	•	•
21	L	H	H	H	H	H	L	–	L	H	L	H	L	H	H	H	•	A	A	A	•	A	•	•
22	H	L	L	H	L	L	L	–	L	H	L	H	H	L	H	H	•	A	A	A	•	A	•	•
23	H	L	L	H	L	L	H	–	L	H	L	H	H	H	H	H	•	A	A	A	•	A	•	•
24	H	L	L	H	L	H	L	–	L	H	H	L	L	L	H	H	•	A	A	A	•	A	•	•
25	H	L	L	H	L	H	H	–	L	H	H	L	L	H	H	H	•	A	A	A	•	A	•	•
26	H	H	H	L	L	L	L	–	L	H	H	L	H	L	H	H	•	A	A	A	•	A	•	•
27	H	H	H	L	L	L	H	–	L	H	H	L	H	H	H	H	•	A	A	A	•	A	•	•
28	H	H	H	L	L	L	L	–	L	H	H	H	L	L	H	H	•	A	A	A	•	A	•	•
29	H	H	H	L	L	L	H	–	L	H	H	H	L	H	H	H	•	A	A	A	•	A	•	•
30	H	H	H	H	H	H	H	–	L	H	H	H	H	L	H	H	•	A	A	A	•	A	•	•
31	H	H	H	H	H	H	L	–	L	H	H	H	H	H	H	H	•	A	A	A	•	A	•	•
32	L	H	L	L	L	L	L	L	H	L	L	L	L	L	H	H	A	•	•	•	•	A	•	•
33	L	H	L	L	L	L	L	H	H	L	L	L	L	H	H	H	A	•	•	•	•	A	•	•
34	L	L	L	L	L	L	L	L	L	–	–	–	–	–	–	–	•	•	•	•	•	•	•	A
35	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	L	•	•	•	•	A	•	•	•
36	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	•	•	•	•	•	•	A	•
INPUT ASSIGNMENT	Q_8 ———————— Q_1								A_5 —————— A_0						Q_A	Q_B								

Figure 4.

Judicious partitioning of a set of input variables allows an FPLA to easily generate a 4-bit output N, where N equals the number of one's in the 8-bit input I_N. The FPLA Program Table implementing this function is shown in Figure 1, in which positive logic is assumed. The correct output appears after two passes have occurred in the array. A first pass is used to calculate two interim values R_L and R_H. They are partial expressions of the number of one's over two 4-bit sections of input I_N called I_L and I_H respectively. R_L and R_H are fed back to inputs 12 through 15 in the input section, and together with I_N generate the final result N in a second array pass through product terms 20-33.

FINAL FPLA PROGRAM TABLE FOR N-OUT OF -8

DECODE FUNCTION. R_H, R_L ARE FOLDBACK CONNECTIONS

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	L	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	L	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
1	H	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
2	H	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
3	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
4	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
5	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
6	L	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	•	•	•	•
7	L	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	•	•	•	•
8	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	•	•	•	•
9	H	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	•	•	•	•
10	–	–	–	–	L	H	H	H	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
11	–	–	–	–	H	L	H	H	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
12	–	–	–	–	H	H	L	H	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
13	–	–	–	–	L	L	L	L	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
14	–	–	–	–	H	H	H	H	–	–	–	–	–	–	–	–	•	•	A	•	•	•	•	•
15	–	–	–	–	L	L	L	H	–	–	–	–	–	–	–	–	•	•	A	•	•	•	•	•
16	–	–	–	–	L	L	L	H	–	–	–	–	–	–	–	–	•	•	A	A	•	•	•	•
17	–	–	–	–	L	L	H	L	–	–	–	–	–	–	–	–	•	•	A	A	•	•	•	•
18	–	–	–	–	L	H	L	L	–	–	–	–	–	–	–	–	•	•	A	A	•	•	•	•
19	–	–	–	–	H	L	L	L	–	–	–	–	–	–	–	–	•	•	A	A	•	•	•	•
20	L	L	L	L	L	L	L	L	–	–	–	–	–	–	–	–	•	•	•	•	•	A	•	•
21	L	L	L	L	–	–	–	–	–	–	–	–	H	H	–	–	•	•	•	•	•	A	•	•
22	–	–	–	–	L	L	L	L	–	–	–	–	–	–	H	H	•	•	•	•	•	A	•	•
23	–	–	–	–	–	–	–	–	–	–	–	–	–	H	–	L	•	•	•	•	•	•	•	A
24	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	H	•	•	•	•	•	•	•	A
25	L	L	L	L	–	–	–	–	–	–	–	–	L	–	–	–	•	•	•	•	•	A	A	•
26	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	•	•	•	•	•	A	A	•
27	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	–	•	•	•	•	•	A	A	•
28	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	•	•	•	•	•	A	A	•
29	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	L	•	•	•	•	•	A	A	•
30	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	–	•	•	•	•	•	•	A	•
31	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	•	•	•	•	•	•	A	•
32	H	H	H	H	–	–	–	–	–	–	–	–	L	–	–	–	•	•	•	•	•	•	A	•
33	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	–	•	•	•	•	A	A	•	•

COMPRESSED

I_H I_L I_N R_H R_L N

Figure 1.

Minimization of product terms has been achieved by careful selection of the definition of R_L and R_H.

A 2-bit number can denote four different cases, yet the number of one's in I_L (or I_H) can range from zero to four (5 cases) with the following frequencies of occurence: 1, 4, 6, 4, 1. It thus makes sense to define R_L (and R_H) in such a way that:

a) the default code 00 is used to express the case of the highest occurrence.
b) the two cases of lowest occurrence are chosen to be the cases expressed by a common code.

As a result, the following definition can be adopted for both design cases:

$$R_L = \begin{cases} 00 & \text{when there are 2 one's in } I_L \\ 01 & \text{when there are 3 one's in } I_L \\ 11 & \text{when there is 1 one in } I_L \\ 10 & \text{for either no one's or all one's in } I_L \end{cases}$$

The same definition is adopted for R_H in relation to I_H. Notice that the righthand bit of R is a "1" for an odd number of one's and a "0" for an even number. Based on this definition a first design pass resulted in 44 product terms, with the first 20 shown in Figure 1. The other 24 product terms are shown in Figure 2. Addition of a product term for the case of all zero's in I_N, and simultaneous inversion of active level for output 5 transforms the table in Figure 2 to that in Figure 3, indicating only 20 product terms are required for array pass 2. Visual inspection of

PART OF FPLA TABLE FROM INITIAL DESIGN PASS.

NOTE F_2 ASSIGNED ACTIVE-HIGH

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0																								
1																								
2																								
20	L	L	L	L	–	–	–	–	–	–	–	–	H	H	–	–	•	•	•	•	•	•	•	A
21	–	–	–	–	L	L	L	L	–	–	–	–	–	–	H	H	•	•	•	•	•	•	•	A
22	L	L	L	L	–	–	–	–	–	–	–	–	L	L	–	–	•	•	•	•	•	•	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	•	•	•	•	•	•	A	•
24	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	L	•	•	•	•	•	•	A	•
25	L	L	L	L	–	–	–	–	–	–	–	–	L	H	–	–	•	•	•	•	•	•	A	A
26	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	•	•	•	•	•	•	A	A
27	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	L	•	•	•	•	•	•	A	A
28	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	H	•	•	•	•	•	•	A	A
29	L	L	L	L	H	H	H	H	–	–	–	–	–	–	–	–	•	•	•	•	•	A	•	•
30	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	•	•	•	•	•	A	•	•
31	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L	•	•	•	•	•	A	•	•
32	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H	•	•	•	•	•	A	•	•
33	H	H	H	H	L	L	L	L	–	–	–	–	–	–	–	–	•	•	•	•	•	A	•	•
34	–	–	–	–	H	H	H	H	–	–	–	–	–	–	H	H	•	•	•	•	•	A	•	A
35	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	L	•	•	•	•	•	A	•	A
36	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	•	•	•	•	•	A	•	A
37	H	H	H	H	–	–	–	–	–	–	–	–	H	H	–	–	•	•	•	•	•	A	•	A
38	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	L	•	•	•	•	•	A	A	•
39	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	•	•	•	•	•	A	A	•
40	H	H	H	H	–	–	–	–	–	–	–	–	L	L	–	–	•	•	•	•	•	A	A	•
41	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	H	•	•	•	•	•	A	A	A
42	H	H	H	H	–	–	–	–	–	–	–	–	L	H	–	–	•	•	•	•	•	A	A	A
43	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	–	•	•	•	•	A	•	•	•

Figure 2.

Figure 3 for commonalities resulting in "don't care" conditions allows a further reduction to only 14 product terms as shown in the bottom part of the table in Figure 1. A variant of this method yields the function incorporated in the table of Figure 4. Here, a single output labeled "yes" is activated when the number of one's in input I_N equals the binary number of 4-bit control input M. The function could have been obtained using an FPLA programmed as in Figure 1 followed by an MSI compare module (SN7485). However, this would have increased the total circuit delay. As Figure 4 shows, it is still possible to perform the entire function within one FPLA with two array passes.

PART OF FPLA TABLE AFTER SECOND PASS.

NOTE F_2 CHANGED TO ACTIVE – LOW

	PRODUCT TERM															
	INPUT VARIABLE															
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																
1																
2																
20	L	L	L	L	–	–	–	–	–	–	–	–	H	H	–	–
21	–	–	–	–	L	L	L	L	–	–	–	–	–	–	H	H
22	L	L	L	L	–	–	–	–	–	–	–	–	L	L	–	–
23	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H
24	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	L
25	L	L	L	L	–	–	–	–	–	–	–	–	L	H	–	–
26	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H
27	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	L
28	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	H
29	L	L	L	L	H	H	H	H	–	–	–	–	–	–	–	–
30	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H
31	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L
32	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H
33	H	H	H	H	L	L	L	L	–	–	–	–	–	–	–	–
34	–	–	–	–	H	H	H	H	–	–	–	–	–	–	H	H
35	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	L
36	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H
37	H	H	H	H	–	–	–	–	–	–	–	–	H	H	–	–
38	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	L
39	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H
40	H	H	H	H	–	–	–	–	–	–	–	–	L	L	–	–
41	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	H
42	H	H	H	H	–	–	–	–	–	–	–	–	L	H	–	–
43	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	–
44	L	L	L	L	L	L	L	L	–	–	–	–	–	–	–	–

	ACTIVE LEVEL							
	H	H	H	H	H	L	H	H
	OUTPUT FUNCTION							
	7	6	5	4	3	2	1	0
0								
1								
2								
20	•	•	•	•	•	A	•	A
21	•	•	•	•	•	A	•	A
22	•	•	•	•	•	A	A	•
23	•	•	•	•	•	A	A	•
24	•	•	•	•	•	A	A	•
25	•	•	•	•	•	A	A	A
26	•	•	•	•	•	A	A	A
27	•	•	•	•	•	A	A	A
28	•	•	•	•	•	A	A	A
29	•	•	•	•	•	•	•	•
30	•	•	•	•	•	•	•	•
31	•	•	•	•	•	•	•	•
32	•	•	•	•	•	•	•	•
33	•	•	•	•	•	•	•	•
34	•	•	•	•	•	•	•	A
35	•	•	•	•	•	•	•	A
36	•	•	•	•	•	•	•	A
37	•	•	•	•	•	•	•	A
38	•	•	•	•	•	•	A	•
39	•	•	•	•	•	•	A	•
40	•	•	•	•	•	•	A	•
41	•	•	•	•	•	•	A	A
42	•	•	•	•	•	•	A	A
43	•	•	•	•	A	A	•	•
44	•	•	•	•	•	A	•	•

Figure 3.

FPLA PROGRAM TABLE FOR FUNCTION "YES" = "1" WHEN NUMBER OF 1's IN I_N AND M ARE THE SAME

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H H H H H H H H / OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	L	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
1	H	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
2	H	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
3	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
4	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•
5	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•
6	L	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
7	L	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
8	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
9	H	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
10	–	–	–	–	L	H	H	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
11	–	–	–	–	H	L	H	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
12	–	–	–	–	H	H	L	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
13	–	–	–	–	H	H	H	L	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
14	–	–	–	–	L	L	L	L	–	–	–	–	–	–	–	–	•	•	A	•	A	A	A	•
15	–	–	–	–	H	H	H	H	–	–	–	–	–	–	–	–	•	•	A	•	A	A	A	•
16	–	–	–	–	L	L	L	H	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
17	–	–	–	–	H	L	H	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
18	–	–	–	–	L	H	L	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
19	–	–	–	–	H	L	L	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
20	L	L	L	L	L	L	L	L	L	L	L	L	–	–	–	–	•	•	•	•	A	A	A	A
21	L	L	L	L	–	–	–	–	L	L	L	H	H	H	–	–	•	•	•	•	A	A	A	A
22	–	–	–	–	L	L	L	L	L	L	L	H	–	–	–	–	•	•	•	•	A	A	A	A
23	L	L	L	L	–	–	–	–	L	L	H	L	L	L	–	–	•	•	•	•	A	A	A	A
24	–	–	–	–	–	–	–	–	L	L	H	L	H	H	H	H	•	•	•	•	A	A	A	A
25	–	–	–	–	L	L	L	L	L	L	H	L	–	–	L	L	•	•	•	•	A	A	A	A
26	L	L	L	L	–	–	–	–	L	L	H	H	L	H	–	–	•	•	•	•	A	A	A	A
27	–	–	–	–	–	–	–	–	L	L	H	H	L	L	H	H	•	•	•	•	A	A	A	A
28	–	–	–	–	–	–	–	–	L	L	H	H	H	H	L	L	•	•	•	•	A	A	A	A
29	–	–	–	–	L	L	L	L	L	L	H	H	–	–	L	H	•	•	•	•	A	A	A	A
30	L	L	L	L	H	H	H	H	L	H	L	L	–	–	–	–	•	•	•	•	A	A	A	A
31	–	–	–	–	–	–	–	–	L	H	L	L	L	H	H	H	•	•	•	•	A	A	A	A
32	–	–	–	–	–	–	–	–	L	H	L	L	L	L	L	L	•	•	•	•	A	A	A	A
33	–	–	–	–	–	–	–	–	L	H	L	L	H	H	L	H	•	•	•	•	A	A	A	A
34	H	H	H	H	L	L	L	L	L	H	L	L	–	–	–	–	•	•	•	•	A	A	A	A
35	–	–	–	–	H	H	H	H	L	H	L	H	–	–	H	H	•	•	•	•	A	A	A	A
36	–	–	–	–	–	–	–	–	L	H	L	H	L	H	L	L	•	•	•	•	A	A	A	A
37	–	–	–	–	–	–	–	–	L	H	L	H	L	L	L	H	•	•	•	•	A	A	A	A
38	H	H	H	H	–	–	–	–	L	H	L	H	H	H	–	–	•	•	•	•	A	A	A	A
39	–	–	–	–	H	H	H	H	L	H	H	L	–	–	L	L	•	•	•	•	A	A	A	A
40	–	–	–	–	–	–	–	–	L	H	H	L	L	H	L	H	•	•	•	•	A	A	A	A
41	H	H	H	H	–	–	–	–	L	H	H	L	L	L	–	–	•	•	•	•	A	A	A	A
42	–	–	–	–	H	H	H	H	L	H	H	H	–	–	L	H	•	•	•	•	A	A	A	A
43	H	H	H	H	–	–	–	–	L	H	H	H	L	H	–	–	•	•	•	•	A	A	A	A
44	H	H	H	H	H	H	H	H	H	L	L	L	–	–	–	–	•	•	•	•	A	A	A	A

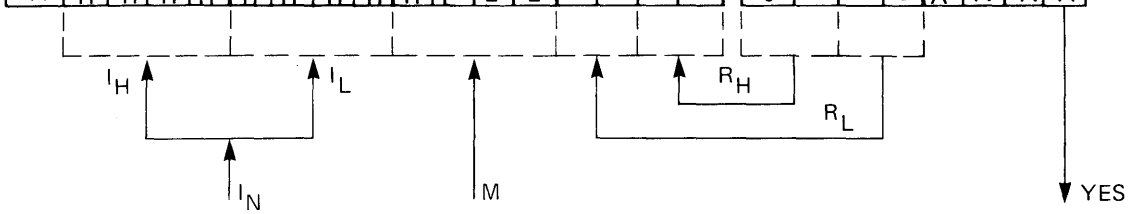


Figure 4.

A shift/logic unit (SLU) with FPLA's can provide high-speed field and bit manipulations for data packing in fast I/O controllers, and minicomputers used in number crunching applications. The block diagram of Figure 1 illustrates the SLU environment, consisting of a multi-port register file, such as Signetics' 82S112, and a control sequencer which could also be designed with additional FPLA's.

Any two registers, not necessarily different, can be read on the A and B ports and supplied as left and right operands to the SLU. The result, Z, can be returned to any register in the register file.

The SLU is composed of 4 FPLA pairs programmed as SLU1 and SLU2, for a total of 8 FPLA's. These are connected as shown in Figure 2 to implement the overall SLU instruction set tabulated in Figure 3.

The SLU and register file are 16 bits wide, and are both controlled by an 8-bit word from the control sequencer. The control word, mapped in Figure 4, commands the SLU to perform three basic operations which are selected by mode bits M0 and M1. These are:

SYSTEM BLOCK DIAGRAM

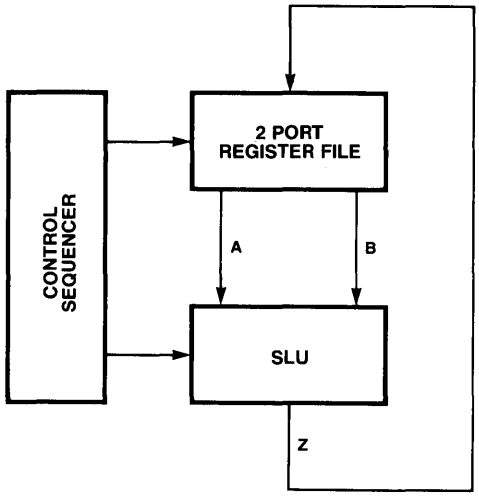


Figure 1.

SLU DIAGRAM SHOWING I/O CONNECTIONS FOR EACH FPLA PAIR

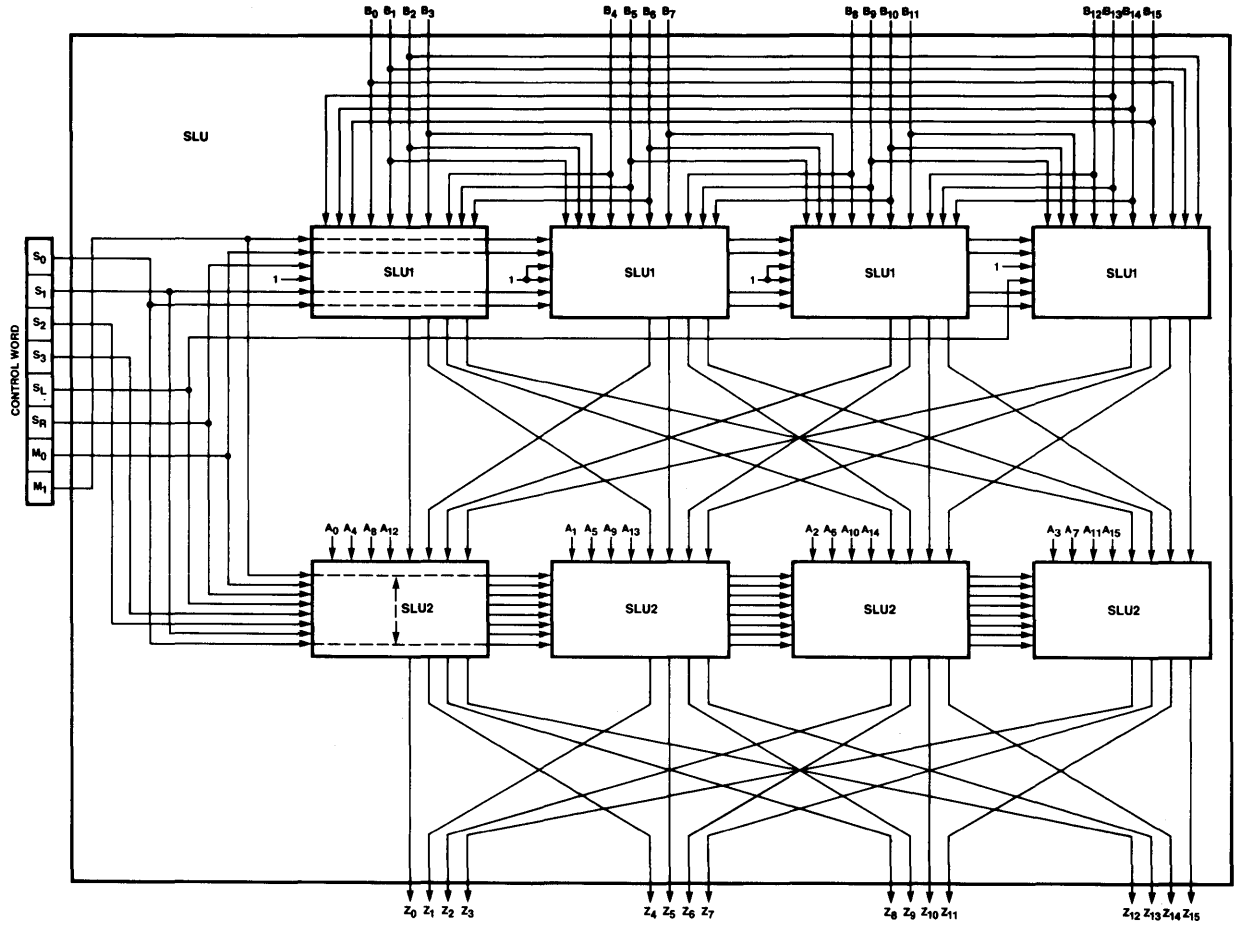


Figure 2.

SLU INSTRUCTION SET

Mnemonic	M_1	M_0	S_R	S_L	S_3	S_2	S_1	S_0	Boolean Operation	Comment
ZERO	0	1	X	X	0	0	0	0	0	Logical zero
NOR	0	1	X	X	0	0	0	1	$\bar{A}\bar{B}$	NOR
NOR NOT	0	1	X	X	0	0	1	0	$\bar{A}B$	NOR NOT
NLID	0	1	X	X	0	0	1	1	$\bar{A}$	Left identity complement
AND NOT	0	1	X	X	0	1	0	0	$A\bar{B}$	AND NOT
NRID	0	1	X	X	0	1	0	1	$\bar{B}$	Right identity complement
EXOR	0	1	X	X	0	1	1	0	$A \oplus B$	Exclusive or
NAND	0	1	X	X	0	1	1	1	$\bar{A} + \bar{B}$	NAND
AND	0	1	X	X	1	0	0	0	AB	AND
EQV	0	1	X	X	1	0	0	1	$A \odot B$	Equivalence
RID	0	1	X	X	1	0	1	0	B	right identity
NAND NOT	0	1	X	X	1	0	1	1	$\bar{A} + B$	NAND NOT
LID	0	1	X	X	1	1	0	0	A	left identity
OR NOT	0	1	X	X	1	1	0	1	$A + \bar{B}$	OR NOT
OR	0	1	X	X	1	1	1	0	$A + B$	OR
ONE	0	1	X	X	1	1	1	1	1	Logical one
SOFF RN	1	0	0	1	(	N_2^*		)		Shift B right end-off N places
SCIR RN	1	0	1	1	(	N_2		)		Shift B right circular N places
SOFF LN	1	1	1	0	(	N_2		)		Shift B left end-off N places
SCIR LN	1	1	1	1	(	N_2		)		Shift B left circular N places
TEST	0	0	X	X	(	N_2		)		Logical zero if B = 0 Logical one if B ≠ 0

*N_2 = binary number specified by $S_{0\sim3}$.

Figure 3.

MAP OF SLU CONTROL WORD

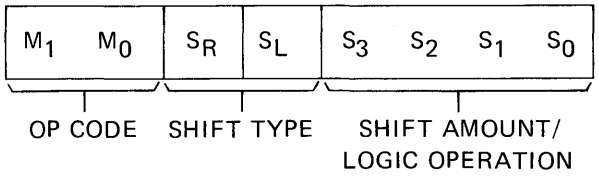


Figure 4.

A. Shift left or right, circular or end-off any number of places.
B. Execute any of 16 boolean operations for 2 variables.
C. Test for zero.

Shift-type bits, S_R and S_L, control execution of circular or end-off shift. These bits are ignored in other than shift operation. Select bits S_{0-3} supply the shift amount (0 thru 15) in shift modes, or determine one of 16 boolean functions in logic mode. The SLU instruction set has been partitioned in two distinct logic equation sets incorporated in two FPLA types designated respectively SLU1 and SLU2. The

I/O assignment for each FPLA type is defined in Figure 5. The program table for each FPLA can be derived by implementing the logic equation sets tabulated Figures 6 and 7 respectively. Both figures express in a short-hand notation (similar to a multiplication table) the logic equations for SLU1 and SLU2, involving their inputs, outputs, and number of Product Terms necessary to implement each function group. Unlisted input combinations are not allowed, and can be treated as Don't Care during minimization. For example, the logic equation for SLU1 output X_0 is obtained by minimizing with a Karnaugh map all logic products tabulated in Figure 6. These are formed by plotting all entries in column X_0 of the FPLA outputs in appropriate squares corresponding to the logic value of the FPLA inputs. This procedure is illustrated in the map of Figure 8, in which all unlisted input combinations have been assigned a Don't Care because of input constraints.

After minimizing all adjacent squares, output X_0 is given by the sum of 12 product terms as follows:

$$X_0 = B_0(\overline{M}_1 M_0) + B_0(M_1\overline{S}_1\overline{S}_0) + B_{-1}(M_1\overline{M}_0 S_R\overline{S}_1 S_0) + B_{-2}(M_1\overline{M}_0 S_R S_1\overline{S}_0) + B_{-3}(M_1\overline{M}_0 S_R S_1 S_0) + B_1(M_1 M_0\overline{S}_1 S_0) + B_2(M_1 M_0 S_1\overline{S}_0) + B_3(M_1 M_0 S_1 S_0) + (B_0+B_1+B_2+B_3)\overline{M}_1\overline{M}_0$$

Outputs $X_{1\text{-}3}$ are treated the same way, and we obtain a total of 36 product terms which are tabulated in the FPLA Program Table for SLU1 in Figure 9. The Program Table for SLU2 is derived by following a similar procedure.

I/O ASSIGNMENT OF FPLA PAIR SLU1 and SLU2

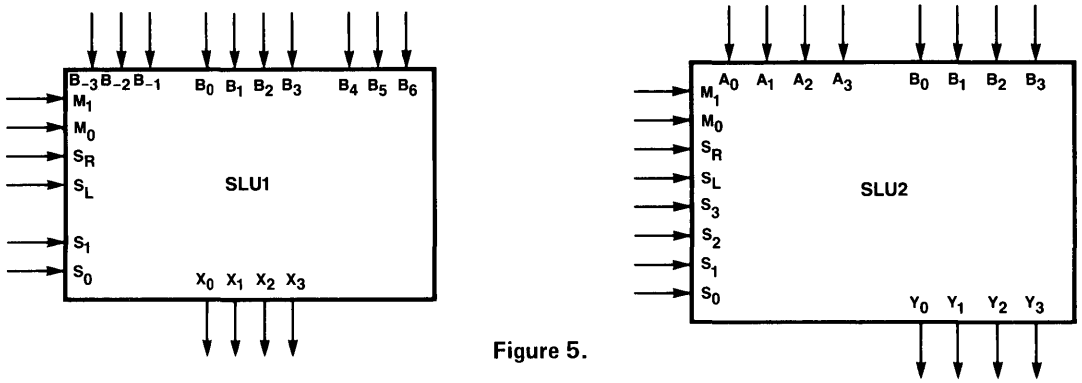


Figure 5.

SLU1 FUNCTION TABLE. Note that B_0–B_6 and B_{-1}→B_{-3} are also FPLA inputs

FPLA INPUTS						FPLA OUTPUTS				P-TERMS REQUIRED
M_1	M_0	S_R	S_L	S_1	S_0	X_0	X_1	X_2	X_3	
0	1	X	X	X	X	B_0	B_1	B_2	B_3	4
1	0	0	1	0	0	B_0	B_1	B_2	B_3	16
1	0	0	1	0	1	0	B_0	B_1	B_2	
1	0	0	1	1	0	0	0	B_0	B_1	
1	0	0	1	1	1	0	0	0	B_0	
1	0	1	1	0	0	B_0	B_1	B_2	B_3	
1	0	1	1	0	1	B_{-1}	B_0	B_1	B_2	
1	0	1	1	1	0	B_{-2}	B_{-1}	B_0	B_1	
1	0	1	1	1	1	B_{-3}	B_{-2}	B_{-1}	B_0	
1	1	1	0	0	0	B_0	B_1	B_2	B_3	12
1	1	1	0	0	1	B_1	B_2	B_3	0	
1	1	1	0	1	0	B_2	B_3	0	0	
1	1	1	0	1	1	B_3	0	0	0	
1	1	1	1	0	0	B_0	B_1	B_2	B_3	
1	1	1	1	0	1	B_1	B_2	B_3	B_4	
1	1	1	1	1	0	B_2	B_3	B_4	B_5	
1	1	1	1	1	1	B_3	B_4	B_5	B_6	
0	0	X	X	X	X	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	4

Figure 6.

SLU2 TRUTH TABLE. Note that A_{0-3} and B_{0-3} are also FPLA inputs

FPLA INPUTS								FPLA OUTPUTS				P-TERMS
M_1	M_0	S_R	S_L	S_3	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	REQUIRED
0	1	X	X	0	0	0	0	0	0	0	0	
0	1	X	X	0	0	0	1	$\bar{A}_0\bar{B}_0$	$\bar{A}_1\bar{B}_1$	$\bar{A}_2\bar{B}_2$	$\bar{A}_3\bar{B}_3$	
0	1	X	X	0	0	1	0	$\bar{A}_0B_0$	$\bar{A}_1B_1$	$\bar{A}_2B_2$	$\bar{A}_3B_3$	
0	1	X	X	0	0	1	1	$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	
0	1	X	X	0	1	0	0	$A_0\bar{B}_0$	$A_1\bar{B}_1$	$A_2\bar{B}_2$	$A_3\bar{B}_3$	
0	1	X	X	0	1	0	1	$\bar{B}_0$	$\bar{B}_1$	$\bar{B}_2$	$\bar{B}_3$	
0	1	X	X	0	1	1	0	$A_0\bar{B}_0+\bar{A}_0B_0$	$A_1\bar{B}_1+\bar{A}_1B_1$	$A_2\bar{B}_2+\bar{A}_2B_2$	$A_3\bar{B}_3+\bar{A}_3B_3$	
0	1	X	X	0	1	1	1	$\bar{A}_0+\bar{B}_0$	$\bar{A}_1+\bar{B}_1$	$\bar{A}_2+\bar{B}_2$	$\bar{A}_3+\bar{B}_3$	16
0	1	X	X	1	0	0	0	A_0B_0	A_1B_1	A_2B_2	A_3B_3	
0	1	X	X	1	0	0	1	$A_0B_0+\bar{A}_0\bar{B}_0$	$AA_1B_1+\bar{A}_1\bar{B}_1$	$A_2B_2+\bar{A}_2\bar{B}_2$	$AA_3B_3+\bar{A}_3\bar{B}_3$	
0	1	X	X	1	0	1	0	B_0	B_1	B_2	B_3	
0	1	X	X	1	0	1	1	$\bar{A}_0+B_0$	$\bar{A}_1+B_1$	$\bar{A}_2+B_2$	$\bar{A}_3+B_3$	
0	1	X	X	1	1	0	0	A_0	A_1	A_2	A_3	
0	1	X	X	1	1	0	1	$A_0+\bar{B}_0$	$A_1+\bar{B}_1$	$A_2+\bar{B}_2$	$A_3+\bar{B}_3$	
0	1	X	X	1	1	1	0	A_0+B_0	A_1+B_1	A_2+B_2	A_3+B_3	
0	1	X	X	1	1	1	1	1	1	1	1	
1	0	0	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	0	0	1	0	1	X	X	0	B_0	B_1	B_2	
1	0	0	1	1	0	X	X	0	0	B_0	B_1	
1	0	0	1	1	1	X	X	0	0	0	B_0	
1	0	1	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	0	1	1	0	1	X	X	B_3	B_0	B_1	B_2	28
1	0	1	1	1	0	X	X	B_2	B_3	B_0	B_1	
1	0	1	1	1	1	X	X	B_1	B_2	B_3	B_0	
1	1	1	0	0	0	X	X	B_0	B_1	B_2	B_3	
1	1	1	0	0	1	X	X	B_1	B_2	B_3	0	
1	1	1	0	1	0	X	X	B_2	B_3	0	0	
1	1	1	0	1	1	X	X	B_3	0	0	0	
1	1	1	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	1	1	1	0	1	X	X	B_1	B_2	B_3	B_0	
1	1	1	1	1	0	X	X	B_2	B_3	B_0	B_1	
1	1	1	1	1	1	X	X	B_3	B_0	B_1	B_2	
0	0	X	X	X	X	X	X	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	4

Figure 7.

KARNAUGH MAP FOR OUTPUT X_0 OF SLU1, WHERE $C = B_0 + B_1 + B_2 + B_3$, and (X) = DON'T CARE

			$M_0 = 0$				$M_0 = 1$			
		S_1, S_0	00	01	11	10	00	01	11	10
$M_1 = 0$	S_R, S_L	00	C	C	C	C	B_0	B_0	B_0	B_0
		01	C	C	C	C	B_0	B_0	B_0	B_0
		11	C	C	C	C	B_0	B_0	B_0	B_0
		10	C	C	C	C	B_0	B_0	B_0	B_0
$M_1 = 0$		00	X	X	X	X	X	X	X	X
		01	B_0	0	0	0	X	X	X	X
		11	B_0	B-1	B-3	B-2	B_0	B_1	B_3	B_2
		10	X	X	X	X	B_0	B_1	B_3	B_2

Figure 8.

FPLA PROGRAM TABLE FOR SLU1

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	L	H	–	–	–	–	–	–	–	H	–	–	–	–	–	–	A	•	•	•	A	A	A	A
1	L	H	–	–	–	–	–	–	–	–	H	–	–	–	–	–	•	A	•	•	A	A	A	A
2	L	H	–	–	–	–	–	–	–	–	–	H	–	–	–	–	•	•	A	•	A	A	A	A
3	L	H	–	–	–	–	–	–	–	–	–	–	H	–	–	–	•	•	•	A	A	A	A	A
4	H	–	–	–	L	L	–	–	–	H	–	–	–	–	–	–	A	•	•	•	A	A	A	A
5	H	–	–	–	L	L	–	–	–	–	H	–	–	–	–	–	•	A	•	•	A	A	A	A
6	H	–	–	–	L	L	–	–	–	–	–	H	–	–	–	–	•	•	A	•	A	A	A	A
7	H	–	–	–	L	L	–	–	–	–	–	–	H	–	–	–	•	•	•	A	A	A	A	A
8	H	L	H	–	L	H	–	–	H	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
9	H	L	–	–	L	H	–	–	–	H	–	–	–	–	–	–	•	A	•	•	A	A	A	A
10	H	L	–	–	L	H	–	–	–	–	H	–	–	–	–	–	•	•	A	•	A	A	A	A
11	H	L	–	–	L	H	–	–	–	–	–	H	–	–	–	–	•	•	•	A	A	A	A	A
12	H	L	H	–	H	L	–	H	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
13	H	L	H	–	H	L	–	–	H	–	–	–	–	–	–	–	•	A	•	•	A	A	A	A
14	H	L	–	–	H	L	–	–	–	H	–	–	–	–	–	–	•	•	A	•	A	A	A	A
15	H	L	–	–	H	L	–	–	–	–	H	–	–	–	–	–	•	•	•	A	A	A	A	A
16	H	L	H	–	H	H	H	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
17	H	L	H	–	H	H	–	H	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	A
18	H	L	H	–	H	H	–	–	H	–	–	–	–	–	–	–	•	•	A	•	A	A	A	A
19	H	L	–	–	H	H	–	–	–	H	–	–	–	–	–	–	•	•	•	A	A	A	A	A
20	H	H	–	–	L	H	–	–	–	–	H	–	–	–	–	–	A	A	•	•	A	A	A	A
21	H	H	–	–	L	H	–	–	–	–	–	H	–	–	–	–	•	A	•	•	A	A	A	A
22	H	H	–	–	L	H	–	–	–	–	–	–	H	–	–	–	•	•	A	•	A	A	A	A
23	H	H	–	H	L	H	–	–	–	–	–	–	–	H	–	–	•	•	•	A	A	A	A	A
24	H	H	–	–	H	L	–	–	–	–	–	H	–	–	–	–	A	•	•	•	A	A	A	A
25	H	H	–	–	H	L	–	–	–	–	–	–	H	–	–	–	•	A	•	•	A	A	A	A
26	H	H	–	H	H	L	–	–	–	–	–	–	–	H	–	–	•	•	A	•	A	A	A	A
27	H	H	–	H	H	L	–	–	–	–	–	–	–	–	H	–	•	•	•	A	A	A	A	A
28	H	H	–	–	H	H	–	–	–	–	–	–	H	–	–	–	A	•	•	•	A	A	A	A
29	H	H	–	H	H	H	–	–	–	–	–	–	–	H	–	–	•	A	•	•	A	A	A	A
30	H	H	–	H	H	H	–	–	–	–	–	–	–	–	H	–	•	•	A	•	A	A	A	A
31	H	H	–	H	H	H	–	–	–	–	–	–	–	–	–	H	•	•	•	A	A	A	A	A
32	L	L	–	–	–	–	–	–	–	H	–	–	–	–	–	–	A	A	A	A	A	A	A	A
33	L	L	–	–	–	–	–	–	–	–	H	–	–	–	–	–	A	A	A	A	A	A	A	A
34	L	L	–	–	–	–	–	–	–	–	–	H	–	–	–	–	A	A	A	A	A	A	A	A
35	L	L	–	–	–	–	–	–	–	–	–	–	H	–	–	–	A	A	A	A	A	A	A	A
I/O ASSIGNMENT	M_1	M_0	S_R	S_L	S_1	S_0	B_{-3}	B_{-2}	B_{-1}	B_0	B_1	B_2	B_3	B_4	B_5	B_6	X_0	X_1	X_2	X_3	unused			

Figure 9.

An existing sequential circuit can be easily augmented or modified by using an FPLA in conjunction with a 2:1 multiplexer, and thus drastically cut costs associated with a new design.

Given a sequential circuit with the function illustrated in the map of Figure 1a, it becomes a trivial task to alter and increase its state table as in Figure 1b, by adding an FPLA and a 2:1 multiplexer to the initial circuit, as shown in Figure 2. With reference to Figure 1, the (X,Y) combinations corresponding to the circled next states in augmented state table (B) are not programmed in the FPLA; they are provided rather by the existing circuit (A). During these combinations, FPLA output k = "0" by default. The additional states of B, as well as the unmatched portion of (A) are provided by proper programming of the FPLA, at which time k = "1". The function of output k is easily implemented with virtually no dedicated P-terms, except in the case where a next state supplied by the FPLA has an all "0" coding.

D-type flip-flops will be needed to implement the augmented portion of the state table (B). However, as FPLA technology improves they can be included within the FPLA; in addition the FPLA can have more input variable (X) lines than sequential circuit (A).

A MODEL FOR MODIFYING AND/OR AUGMENTING AN EXISTING SEQUENTIAL CIRCUIT

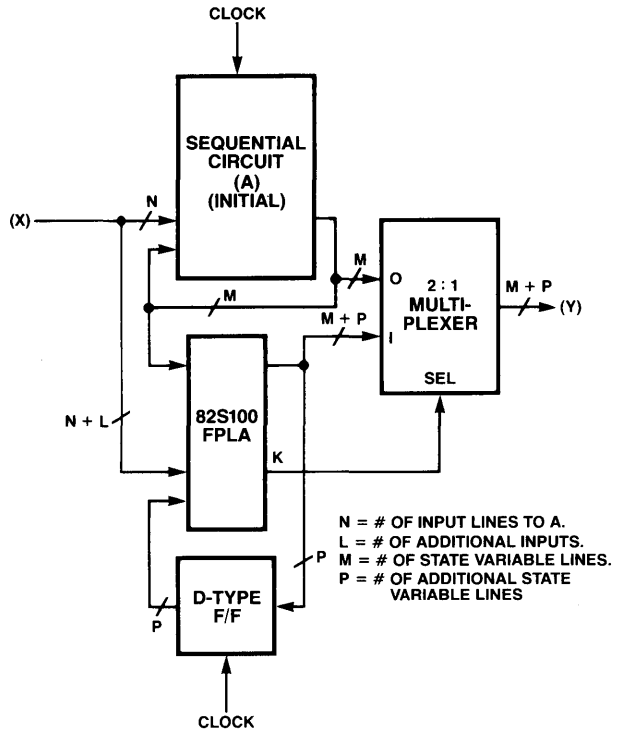


Figure 2.

SEQUENTIAL CIRCUIT MODIFICATION. TRANSITION MAPS WITH STABLE STATE ENTRIES AS A FUNCTION OF INPUT AND PRESENT STATE. CIRCLED ENTRIES DENOTE MATCHED AREAS.

Y \ X	00	01	10	11
A	(A)	D	C	(B)
B	(B)	(C)	(D)	(A)
C	(C)	(A)	B	D
D	(A)	C	(D)	B

a. Sequential circuit (A)

Change to

Y \ X	000	001	010	011	100	101
A	(A)	C	D	(B)	E	F
B	(B)	(C)	(D)	(A)	D	A
C	(C)	(A)	D	B	C	A
D	(A)	B	(D)	D	F	E
E	B	E	D	B	D	E
F	D	C	F	D	F	D

b. Modified sequential circuit

Figure 1.

Two or more dissimilar pieces of digitally tuned communications equipment can be operated with a single bandswitch using an FPLA code converter. For example, two transceivers may be simultaneously bandswitched using the code from one transceiver. In another case, a power amplifier requiring a cyclic bandswitching code may be operated from a transceiver supplying a BCD code.

In an actual case a single FPLA is capable of replacing 10 IC's and 36 pull-up resistors.

The circuit in Figure 1 illustrates how to obtain a 5-wire code, such as used for positioning a power amplifier tuning turret, from a transceiver.

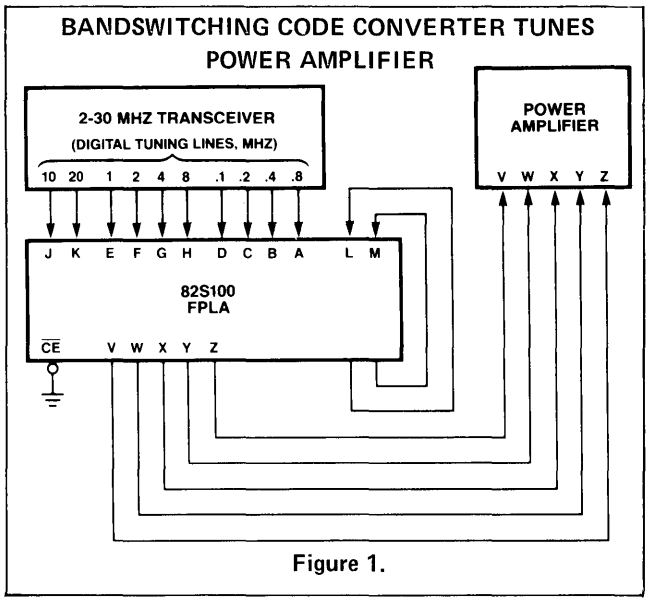


Figure 1.

The power amplifier's bandswitching code is tabulated in Figure 2. The 100KHz BCD tuning lines can be encoded in the FPLA for the X.0 to X.4 MHz and X.5 to X.9 MHz ranges according to the table in Figure 3. Two additional functions, L and M, are fed back into the FPLA to be encoded with the other BCD lines, and are defined as follows:

X.0 to X.4 MHz : $L = \overline{AB} + \overline{BCD}$

X.5 to X.9 MHz : $M = A + BC + BD$

If the other relevant BCD tuning lines are labelled E, F, G, H, J, K, for 1, 2, 3, 4, 8, 10, and 20 MHz lines respectively, a Boolean expression including the two 100KHz functions L and M can be written and encoded into the FPLA for each of the 5 desired output functions. For example, line Y of the 5-wire code is represented:

$$Y = \overline{E}FG\overline{HJK}M + EFG\overline{HJK}L + EFG\overline{HJK}M + \overline{EF}G\overline{HJK} + \overline{E}FG\overline{HJK} + EFG\overline{HJK} + \overline{EF}GH\overline{JK} + E\overline{FGH}\overline{JK} + \overline{E}FG\overline{H}\overline{JK} + EFG\overline{H}\overline{JK} + \overline{EF}GH\overline{JK} + \overline{EFGHJK} + \overline{E}FG\overline{HJK} + EFG\overline{HJK} + \overline{F}HG\overline{HJK}$$

The entire equation set can be programmed in a single FPLA using 12 inputs, 7 outputs, and 35 Product Terms.

AMPLIFIER INPUT CODE ASSIGNMENT

BAND	FREQ. (MHZ)	V W X Y Z	BAND	FREQ. (MHZ)	V W X Y Z
1	2.0 – 2.4	0 0 0 0 1	16	15 – 15.9	0 0 0 1 0
2	2.5 – 2.9	0 0 0 1 1	17	16 – 16.9	0 0 1 0 1
3	3.0 – 3.4	0 0 1 1 1	18	17 – 17.9	0 1 0 1 1
4	3.5 – 3.9	0 1 1 1 1	19	18 – 18.9	1 0 1 1 0
5	4 – 4.9	1 1 1 1 0	20	19 – 19.9	0 1 1 0 1
6	5 – 5.9	1 1 1 0 1	21	20 – 20.9	1 1 0 1 0
7	6 – 6.9	1 1 0 1 1	22	21 – 21.9	1 0 1 0 1
8	7 – 7.9	1 0 1 1 1	23	22 – 22.9	0 1 0 1 0
9	8 – 8.9	0 1 1 1 0	24	23 – 23.9	1 0 1 0 0
10	9 – 9.9	1 1 1 0 0	25	24 – 24.9	0 1 0 0 1
11	10 – 10.9	1 1 0 0 1	26	25 – 25.9	1 0 0 1 1
12	11 – 11.9	1 0 0 1 0	27	26 – 26.9	0 0 1 1 0
13	12 – 12.9	0 0 1 0 0	28	27 – 27.9	0 1 1 0 0
14	13 – 13.9	0 1 0 0 0	29	28 – 28.9	1 1 0 0 0
15	14 – 14.9	1 0 0 0 1	30	29 – 29.9	1 0 0 0 0

Figure 2.

FPLA ENCODING OF kHz TUNING LINES

800 KHz (A)	400 KHz (B)	200 KHz (C)	100 KHz (D)	RANGE (MHZ)
0	0	0	0	X.0
0	0	0	1	X.1
0	0	1	0	X.2
0	0	1	1	X.3
0	1	0	0	X.4
0	1	0	1	X.5
0	1	1	0	X.6
0	1	1	1	X.7
1	0	0	0	X.8
1	0	0	1	X.9

Figure 3.

RF-CHANNEL CONTROL SELECTOR 14

The circuit shown in Figure 1 is used in the output scheduling portion of a digital controller subsystem that generates RF signals for testing receiver antennas in a multipurpose electromagnetic environment simulator.

In this system a channel match signal is used to determine available channels that can produce a signal. One channel match, or all eight channel matches can occur simultaneously. Therefore, the channel match signals and a 5-bit assignment code are used to address an FPLA programmed to resolve a preassigned priority.

The assignment code is used to control the selection of an RF channel in which a signal can be generated. The assignment parameter or code is intended to ensure that a particular signal will always appear on a specific channel when desired. Three levels of assignment priorities are used.

LEVEL 1 – No assignment; selects first available channel, beginning with lowest number channels.

LEVEL 2 – Preferential assignment; selects first available channel starting with the channel given in assignment code.

LEVEL 3 – Mandatory assignment – Selects only the channel given in assignment parameter.

The assignment parameter is a 5-bit code which is used as tabulated in Figure 2. The circuit in Figure 1 implements a matrix function for priority Levels 1 and 2. These are resident in 2 Tri-state FPLA's programmed respectively with sub-tables A and B, shown respectively in Figures 3 and 4. Both FPLAs are operated in parallel and controlled by I_{10}, the ASGN 2 signal, or by the M ASGN bit via their $\overline{CE}$ input.

Priority Level 3 is a mandatory assignment and the FPLA's are disabled. The assignment code (AS0-AS3) is decoded and, provided the channel match for the decoded channel is present, the EN CH signal is then generated.

CIRCUIT DIAGRAM OF RF CHANNEL SELECTOR

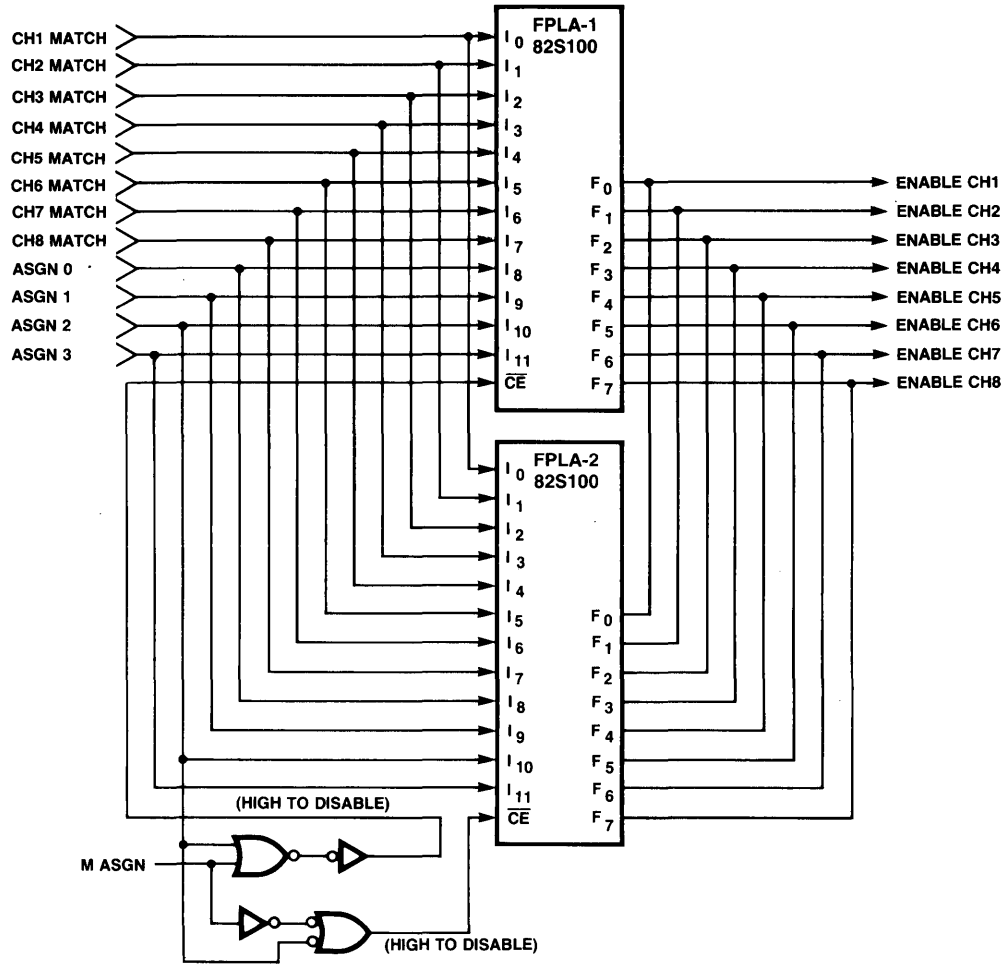


Figure 1.

CHANNEL CODE ASSIGNMENT

CASE	(msb) M ASGN*	AS3	AS2	AS1	(lsb) AS0	CHANNEL
1	O	0	0	0	0	Any
2	M	0	0	0	1	1
3	M	0	0	1	0	2
4	M	0	0	1	1	3
5	M	0	1	0	0	4
6	M	0	1	0	1	5
7	M	0	1	1	0	6
8	M	0	1	1	1	7
9	M	1	0	0	0	8

*M ASGN when a Logic "0" is a preferential assignment, and when a Logic "1" is a mandatory assignment.

Figure 2.

SUBTABLE A PROGRAMMED IN FPLA-1

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	L	L	L	–	–	–	–	–	–	–	–	H	•	•	•	•	•	•	•	A
1	–	–	–	–	L	L	L	–	–	–	–	–	–	–	H	L	•	•	•	•	•	•	A	•
2	–	–	–	–	L	L	L	–	–	–	–	–	–	H	L	L	•	•	•	•	•	A	•	•
3	–	–	–	–	L	L	L	–	–	–	–	–	H	L	L	L	•	•	•	•	A	•	•	•
4	–	–	–	–	L	L	L	–	–	–	–	H	L	L	L	L	•	•	•	A	•	•	•	•
5	–	–	–	–	L	L	L	–	–	–	H	L	L	L	L	L	•	•	A	•	•	•	•	•
6	–	–	–	–	L	L	L	–	–	H	L	L	L	L	L	L	•	A	•	•	•	•	•	•
7	–	–	–	–	L	L	L	–	H	L	L	L	L	L	L	L	A	•	•	•	•	•	•	•
8	–	–	–	–	L	L	H	L	–	–	–	–	–	–	H	–	•	•	•	•	•	•	A	•
9	–	–	–	–	L	L	H	L	–	–	–	–	–	H	L	–	•	•	•	•	•	A	•	•
10	–	–	–	–	L	L	H	L	–	–	–	–	H	L	L	–	•	•	•	•	A	•	•	•
11	–	–	–	–	L	L	H	L	–	–	–	H	L	L	L	–	•	•	•	A	•	•	•	•
12	–	–	–	–	L	L	H	L	–	–	H	L	L	L	L	–	•	•	A	•	•	•	•	•
13	–	–	–	–	L	L	H	L	–	H	L	L	L	L	L	–	•	A	•	•	•	•	•	•
14	–	–	–	–	L	L	H	L	H	L	L	L	L	L	L	–	A	•	•	•	•	•	•	•
15	–	–	–	–	L	L	H	L	L	L	L	L	L	L	L	H	•	•	•	•	•	•	•	A
16	–	–	–	–	L	L	H	H	–	–	–	–	–	H	–	–	•	•	•	•	•	A	•	•
17	–	–	–	–	L	L	H	H	–	–	–	–	H	L	–	–	•	•	•	•	A	•	•	•
18	–	–	–	–	L	L	H	H	–	–	–	H	L	L	–	–	•	•	•	A	•	•	•	•
19	–	–	–	–	L	L	H	H	–	–	H	L	L	L	–	–	•	•	A	•	•	•	•	•
20	–	–	–	–	L	L	H	H	–	H	L	L	L	L	–	–	•	A	•	•	•	•	•	•
21	–	–	–	–	L	L	H	H	H	L	L	L	L	L	–	–	A	•	•	•	•	•	•	•
22	–	–	–	–	L	L	H	H	L	L	L	L	L	L	L	H	•	•	•	•	•	•	•	A
23	–	–	–	–	L	L	H	H	L	L	L	L	L	L	H	L	•	•	•	•	•	•	A	•
24	–	–	–	–	H	L	L	L	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
25	–	–	–	–	H	L	L	L	L	–	–	–	–	–	–	H	•	•	•	•	•	•	•	A
26	–	–	–	–	H	L	L	L	L	–	–	–	–	–	H	L	•	•	•	•	•	•	A	•
27	–	–	–	–	H	L	L	L	L	–	–	–	–	H	L	L	•	•	•	•	•	A	•	•
28	–	–	–	–	H	L	L	L	L	–	–	–	H	L	L	L	•	•	•	•	A	•	•	•
29	–	–	–	–	H	L	L	L	L	–	–	H	L	L	L	L	•	•	•	A	•	•	•	•
30	–	–	–	–	H	L	L	L	L	–	H	L	L	L	L	L	•	•	A	•	•	•	•	•
31	–	–	–	–	H	L	L	L	L	H	L	L	L	L	L	L	•	A	•	•	•	•	•	•
I/O ASSIGNMENT	UNUSED				ASGN 3	ASGN 2	ASGN 1	ASGN 0	CH 8 MATCH	CH 7 MATCH	CH 6 MATCH	CH 5 MATCH	CH 4 MATCH	CH 3 MATCH	CH 2 MATCH	CH 1 MATCH	CH 8 ENB	CH 7 ENB	CH 6 ENB	CH 5 ENB	CH 4 ENB	CH 3 ENB	CH 2 ENB	CH 1 ENB

Figure 3.

SUBTABLE B STORED IN FPLA-2

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION: 7	6	5	4	3	2	1	0
0	–	–	–	–	L	H	L	L	–	–	–	–	H	–	–	–	•	•	•	•	A	•	•	•
1	–	–	–	–	L	H	L	L	–	–	–	H	L	–	–	–	•	•	•	A	•	•	•	•
2	–	–	–	–	L	H	L	L	–	–	H	L	L	–	–	–	•	•	A	•	•	•	•	•
3	–	–	–	–	L	H	L	L	–	H	L	L	L	–	–	–	•	A	•	•	•	•	•	•
4	–	–	–	–	L	H	L	L	H	L	L	L	L	–	–	–	A	•	•	•	•	•	•	•
5	–	–	–	–	L	H	L	L	L	L	L	L	L	–	–	H	•	•	•	•	•	•	•	A
6	–	–	–	–	L	H	L	L	L	L	L	L	L	–	H	L	•	•	•	•	•	•	A	•
7	–	–	–	–	L	H	L	L	L	L	L	L	L	H	L	L	•	•	•	•	•	A	•	•
8	–	–	–	–	L	H	L	H	–	–	–	H	–	–	–	–	•	•	•	A	•	•	•	•
9	–	–	–	–	L	H	L	H	–	–	H	L	–	–	–	–	•	•	A	•	•	•	•	•
10	–	–	–	–	L	H	L	H	–	H	L	L	–	–	–	–	•	A	•	•	•	•	•	•
11	–	–	–	–	L	H	L	H	H	L	L	L	–	–	–	–	A	•	•	•	•	•	•	•
12	–	–	–	–	L	H	L	H	L	L	L	L	–	–	–	H	•	•	•	•	•	•	•	A
13	–	–	–	–	L	H	L	H	L	L	L	L	–	–	H	L	•	•	•	•	•	•	A	•
14	–	–	–	–	L	H	L	H	L	L	L	L	–	H	L	L	•	•	•	•	•	A	•	•
15	–	–	–	–	L	H	L	H	L	L	L	L	H	L	L	L	•	•	•	•	A	•	•	•
16	–	–	–	–	L	H	H	L	–	–	H	–	–	–	–	–	•	•	A	•	•	•	•	•
17	–	–	–	–	L	H	H	L	–	H	L	–	–	–	–	–	•	A	•	•	•	•	•	•
18	–	–	–	–	L	H	H	L	H	L	L	–	–	–	–	–	A	•	•	•	•	•	•	•
19	–	–	–	–	L	H	H	L	L	L	L	–	–	–	–	H	•	•	•	•	•	•	•	A
20	–	–	–	–	L	H	H	L	L	L	L	–	–	–	H	L	•	•	•	•	•	•	A	•
21	–	–	–	–	L	H	H	L	L	L	L	–	–	H	L	L	•	•	•	•	•	A	•	•
22	–	–	–	–	L	H	H	L	L	L	L	–	H	L	L	L	•	•	•	•	A	•	•	•
23	–	–	–	–	L	H	H	L	L	L	L	H	L	L	L	L	•	•	•	A	•	•	•	•
24	–	–	–	–	L	H	H	H	–	H	–	–	–	–	–	–	•	A	•	•	•	•	•	•
25	–	–	–	–	L	H	H	H	H	L	–	–	–	–	–	–	A	•	•	•	•	•	•	•
26	–	–	–	–	L	H	H	H	L	L	–	–	–	–	–	H	•	•	•	•	•	•	•	A
27	–	–	–	–	L	H	H	H	L	L	–	–	–	–	H	L	•	•	•	•	•	•	A	•
28	–	–	–	–	L	H	H	H	L	L	–	–	–	H	L	L	•	•	•	•	•	A	•	•
29	–	–	–	–	L	H	H	H	L	L	–	–	H	L	L	L	•	•	•	•	A	•	•	•
30	–	–	–	–	L	H	H	H	L	L	–	H	L	L	L	L	•	•	•	A	•	•	•	•
31	–	–	–	–	L	H	H	H	L	L	H	L	L	L	L	L	•	•	A	•	•	•	•	•
I/O ASSIGNMENT	UNUSED				ASGN 3	ASGN 2	ASGN 1	ASGN 0	CH 8 MATCH	CH 7 MATCH	CH 6 MATCH	CH 5 MATCH	CH 4 MATCH	CH 3 MATCH	CH 2 MATCH	CH 1 MATCH	CH 8 ENB	CH 7 ENB	CH 6 ENB	CH 5 ENB	CH 4 ENB	CH 3 ENB	CH 2 ENB	CH 1 ENB

Figure 4.

The circuit shown in Figure 1 filters asynchronous serial data as it enter an asynchronous receiver. These devices, often used to deserialize Teletype signals, typically sample each bit of a character only once, in the middle of the bit time.

CIRCUIT DIAGRAM OF SERIAL DATA FILTER

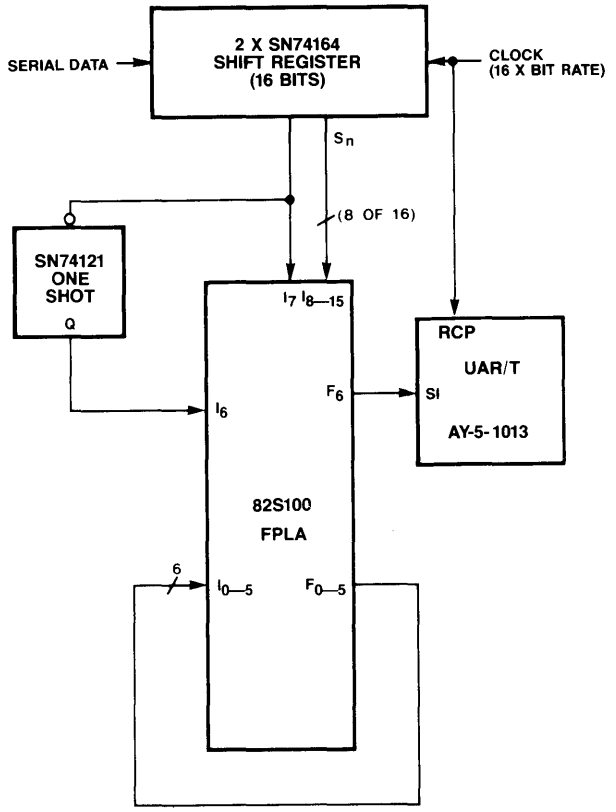


Figure 1.

If a noise pulse, perhaps caused by dirty Teletype distributor, occurs at a sample time, an error results. Low-pass filtering may remove the noise, but may also cause the receiver to start late and erratically since the leading edge of the start bit, used for a time reference is blurred.

A better solution to this problem can be obtained by using an FPLA to implement a 5-out-of 9 majority function from 9 samples of each bit stored in a 16-bit shift register. Since asynchronous receivers typically require a clock at 16 times the bit rate, this is used for the shift register also. For a clean start, an additional input is used to select between a single sample and the majority function. It is controlled by a one-shot which, when fired by the start bit, substitutes the majority function for the middle sample during the rest of the character.

Since a straightforward implementation of the majority function would require 6435 product terms, a 2 level approach is used. The inputs are grouped into triplets (I_{7-9}, I_{10-12}, and I_{13-15},). The I-out-of-3 functions (F_0, F_1, and F_2, respectively) and 2-out-of-3 functions (F_3, F_4, and F_5, respectively) are generated for each triplet. These are fed back to unused inputs and the complete function obtained from output F_6. The complete equation set is tabulated in Fig. 2, which indicates that there are 16 inputs, 29 product terms, and 7 outputs. These are programmed in the FPLA Program Table of Fig. 3.

Other shift registers and receivers will work just as well as those shown. And, if a smaller number of samples is used, say for 4-out-of-8, set $I_2 = I_{15}$ = "1" in the FPLA equations.

The samples should be distributed nearly evenly across the shift register, but may be adjusted to accommodate known signal characteristics such as uncertain bit-boundary locations. The concept can be extended to other systems, such as communication equipment for telephone links etc., even to include nonuniform input weighting to solve a particular noise problem by modifying the FPLA program.

LOGIC EQUATION SET RESIDENT IN THE FPLA

$$I_0 = F_0 = \overline{\bar{I}_7 \bar{I}_8 \bar{I}_9} = I_7 + I_8 + I_9$$

$$I_1 = F_1 = \overline{\bar{I}_{10} \bar{I}_{11} \bar{I}_{12}} = I_{10} + I_{11} + I_{12}$$

$$I_2 = F_2 = \overline{\bar{I}_{13} \bar{I}_{14} \bar{I}_{15}} = I_{13} + I_{14} + I_{15}$$

$$I_3 = F_3 = I_7 I_8 + I_7 I_9 + I_8 I_9$$

$$I_4 = F_4 = I_{10} I_{11} + I_{10} I_{12} + I_{11} I_{12}$$

$$I_5 = F_5 = I_{13} I_{14} + I_{13} I_{15} + I_{14} I_{15}$$

$$F_6 = I_6 I_0 I_1 I_{13} I_{14} I_{15} + I_6 I_0 I_4 I_5 + I_6 I_0 I_{10} I_{11} I_{12} I_2 + I_6 I_3 I_{13} I_{14} I_{15} + I_6 I_3 I_1 I_5 + I_6 I_3 I_4 I_2 + I_6 I_3 I_{10} I_{11} I_{12} + I_6 I_7 I_8 I_9 I_5 + I_6 I_7 I_8 I_9 I_1 I_2 + I_6 I_7 I_8 I_9 I_4 + \bar{I}_6 I_7$$

Figure 2.

FPLA PROGRAM TABLE FOR 5-OUT-OF-9 MAJORITY FUNCTION
FOR 9-BIT WORD SAMPLE OF INPUT DATA

PRODUCT TERM																	ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
1	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
2	–	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
3	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
4	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
5	–	–	–	H	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
6	–	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
7	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
8	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
9	–	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
10	–	–	–	–	–	–	H	–	H	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
11	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
12	–	–	–	–	H	H	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
13	–	–	–	H	–	H	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
14	–	–	–	H	H	–	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
15	–	H	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
16	H	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
17	H	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
18	H	H	H	–	–	–	–	–	–	H	–	–	–	–	H	H	A	A	•	•	•	•	•	•
19	–	–	–	–	–	–	–	–	–	H	H	H	–	–	–	H	A	A	•	•	•	•	•	•
20	–	–	–	H	H	H	–	–	–	H	–	–	–	H	–	H	A	A	•	•	•	•	•	•
21	H	H	H	–	–	–	–	–	–	H	–	–	H	–	–	–	A	A	•	•	•	•	•	•
22	–	–	–	–	–	–	–	–	–	H	H	–	H	–	H	–	A	A	•	•	•	•	•	•
23	–	–	–	–	–	–	–	–	–	H	–	H	H	H	–	–	A	A	•	•	•	•	•	•
24	–	–	–	H	H	H	–	–	–	H	–	–	H	–	–	–	A	A	•	•	•	•	•	•
25	–	–	–	–	–	–	H	H	H	H	H	–	–	–	–	–	A	A	•	•	•	•	•	•
26	–	–	–	–	–	–	H	H	H	H	–	–	–	H	H	–	A	A	•	•	•	•	•	•
27	–	–	–	–	–	–	H	H	H	H	–	H	–	–	–	–	A	A	•	•	•	•	•	•
28	–	–	–	–	–	–	–	–	H	L	–	–	–	–	–	–	A	A	•	•	•	•	•	•
I/O ASSIGNMENT	S_{15}	S_{13}	S_{11}	S_9	S_7	S_5	S_3	S_0	S_8	Q	F_5	F_4	F_3	F_2	F_1	F_0	unused	SI	I_5	I_4	I_3	I_2	I_1	I_0

Figure 3.

The circuit shown in Figure 1 generates an output pulse on one of eight channels whenever the duration of an input pulse falls within the preselected window corresponding to that channel.

The input pulse is applied to input I_0, of the 82S100. It also drives network R_1 C_1, whose exponentially rising and falling output voltage is squared up by G1 to look like a 1/4 μS delayed, inverted version of the input pulse, and is applied to input I_1 of the FPLA. The FPLA forms the logical product $I_0 \cdot \overline{I_1}$ (see timing diagram, Figure 2) as part of each of its 48 product terms, to mark the trailing edge of the input pulse. Simultaneously, the input pulse turns on crystal controlled multivibrator G2-G3, whose pulse output is accumulated by counters IC2 and IC3. Thus as long as the input pulse is high, FPLA inputs I_2 through I_9 will indicate elapsed microseconds. The FPLA is programmed with successive counts and selector switch settings as shown in the Program Table of Figure 3, so that when the trailing edge signal $\overline{I_0} \cdot \overline{I_1}$ is coincident with a specific count and switch setting, the coincidence pulse will appear on the appropriate output line. Network R_2 C_2 differentiates the trailing edge of the delayed input pulse to provide a reset pulse for the counter. The MSB on counter IC3 is inverted by G4 to cage multivibrator G2-G3 whenever the counter overranges (i.e., goes to its upper half-range). This avoids the ambiguity created when long extraneous noise pulses drive the multivibrator for several full counting cycles (256n counts) past any specific desired count. If the pulse transmission line is known to be free of noise pulses, this circuit can be deleted, doubling the count capacity of the counter. With the given program, the FPLA will detect input pulses in the pulse width windows 1 to 2μS, 2 to 3μS, 3 to 4μS, ..., 48 to 49μS. By programming the FPLA so that count bits I_2 through I_9 range from 49 (LLHHLLLH) through 96 (LHHLLLLL) the windows 49 to 50 μS, ..., 96 to 97 μS can be detected instead. Thus with the overrange lock-out by G_4, up to 127 separate pulse widths can be detected by some change in the programming; without the overrange lock-out implemented this number increases to 255.

This circuit can be used to control multi-function machines in several locations throughout a small plant by using the power mains as a pulse transmission line. The system is relatively noise-immune because of the narrow tolerances on the control pulse-widths. The output pulses F_0 through F_7 can be used to control latching solid state A.C. relays, SCRs for motor phase-control, stepper motors, and can even relay audio information by toggling a flip-flop whose filtered, buffered output is applied to a speaker.

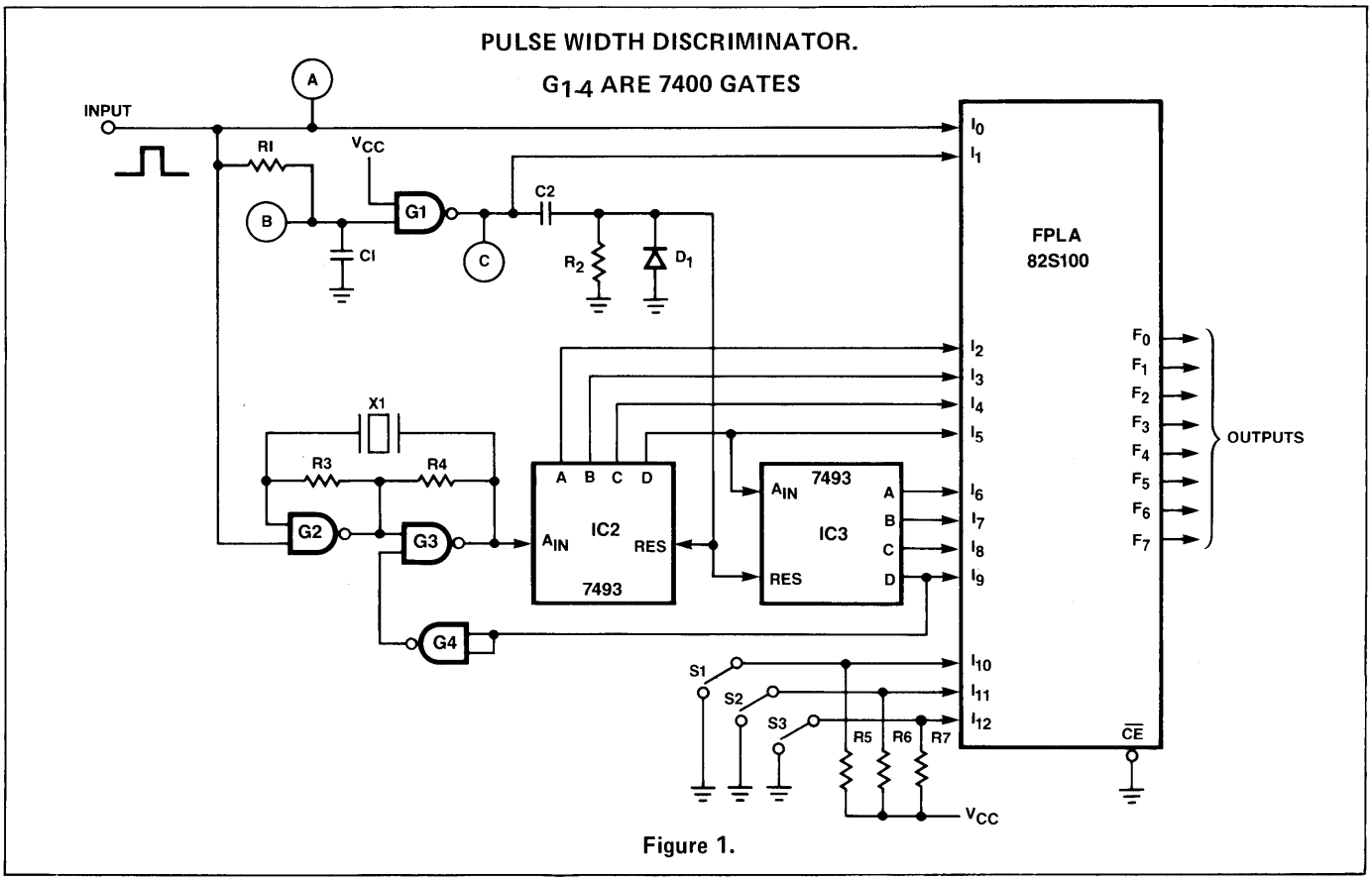


Figure 1.

PARTIAL TIMING DIAGRAM

NO.	PRODUCT TERM / INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL H H H H H H H H / OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	L	L	L	L	L	L	L	L	L	L	H	L	L	A	•	•	•	•	•	•	•
1	–	–	–	L	L	L	L	L	L	L	L	L	H	L	L	L	•	A	•	•	•	•	•	•
2	–	–	–	L	L	L	L	L	L	L	L	H	H	L	L	L	•	•	A	•	•	•	•	•
3	–	–	–	L	L	L	L	L	L	L	L	H	L	L	L	L	•	•	•	A	•	•	•	•
4	–	–	–	L	L	L	L	L	L	L	L	H	L	H	L	L	•	•	•	•	A	•	•	•
5	–	–	–	L	L	L	L	L	L	L	L	H	H	L	L	L	•	•	•	•	•	A	•	•
6	–	–	–	L	L	L	L	L	L	L	L	H	H	H	L	L	•	•	•	•	•	•	A	•
7	–	–	–	L	L	L	L	L	L	L	H	L	L	L	L	L	•	•	•	•	•	•	•	A
8	–	–	–	L	L	H	L	L	L	L	H	L	L	H	L	L	A	•	•	•	•	•	•	•
9	–	–	–	L	L	H	L	L	L	L	H	L	H	L	L	L	•	A	•	•	•	•	•	•
10	–	–	–	L	L	H	L	L	L	L	H	L	H	H	L	L	•	•	A	•	•	•	•	•
11	–	–	–	L	L	H	L	L	L	L	H	H	L	L	L	L	•	•	•	A	•	•	•	•
12	–	–	–	L	L	H	L	L	L	L	H	H	L	H	L	L	•	•	•	•	A	•	•	•
13	–	–	–	L	L	H	L	L	L	L	H	H	H	L	L	L	•	•	•	•	•	A	•	•
14	–	–	–	L	L	H	L	L	L	L	H	H	H	H	L	L	•	•	•	•	•	•	A	•
15	–	–	–	L	L	H	L	L	L	L	H	L	L	L	L	L	•	•	•	•	•	•	•	A
16	–	–	–	L	H	L	L	L	L	H	L	L	L	H	L	L	A	•	•	•	•	•	•	•
17	–	–	–	L	H	L	L	L	L	H	L	L	H	L	L	L	•	A	•	•	•	•	•	•
18	–	–	–	L	H	L	L	L	L	H	L	L	H	H	L	L	•	•	A	•	•	•	•	•
19	–	–	–	L	H	L	L	L	L	H	L	H	L	L	L	L	•	•	•	A	•	•	•	•
20	–	–	–	L	H	L	L	L	L	H	L	H	L	H	L	L	•	•	•	•	A	•	•	•
21	–	–	–	L	H	L	L	L	L	H	L	H	H	L	L	L	•	•	•	•	•	A	•	•
22	–	–	–	L	H	L	L	L	L	H	L	H	H	H	L	L	•	•	•	•	•	•	A	•
23	–	–	–	L	H	L	L	L	L	H	H	L	L	L	L	L	•	•	•	•	•	•	•	A
24	–	–	–	L	H	H	L	L	L	H	H	L	L	H	L	L	A	•	•	•	•	•	•	•
25	–	–	–	L	H	H	L	L	L	H	H	L	H	L	L	L	•	A	•	•	•	•	•	•
26	–	–	–	L	H	H	L	L	L	H	H	L	H	H	L	L	•	•	A	•	•	•	•	•
27	–	–	–	L	H	H	L	L	L	H	H	H	L	L	L	L	•	•	•	A	•	•	•	•
28	–	–	–	L	H	H	L	L	L	H	H	H	L	H	L	L	•	•	•	•	A	•	•	•
29	–	–	–	L	H	H	L	L	L	H	H	H	H	L	L	L	•	•	•	•	•	A	•	•
30	–	–	–	L	H	H	L	L	L	H	H	H	H	H	L	L	•	•	•	•	•	•	A	•
31	–	–	–	L	H	H	L	L	H	L	L	L	L	L	L	L	•	•	•	•	•	•	•	A
32	–	–	–	H	L	L	L	L	H	L	L	L	L	H	L	L	A	•	•	•	•	•	•	•
33	–	–	–	H	L	L	L	L	H	L	L	L	H	L	L	L	•	A	•	•	•	•	•	•
34	–	–	–	H	L	L	L	L	H	L	L	L	H	H	L	L	•	•	A	•	•	•	•	•
35	–	–	–	H	L	L	L	L	H	L	L	H	L	L	L	L	•	•	•	A	•	•	•	•
36	–	–	–	H	L	L	L	L	H	L	L	H	L	H	L	L	•	•	•	•	A	•	•	•
37	–	–	–	H	L	L	L	L	H	L	L	H	H	L	L	L	•	•	•	•	•	A	•	•
38	–	–	–	H	L	L	L	L	H	L	L	H	H	H	L	L	•	•	•	•	•	•	A	•
39	–	–	–	H	L	L	L	L	H	L	H	L	L	L	L	L	•	•	•	•	•	•	•	A
40	–	–	–	H	L	H	L	L	H	L	H	L	L	H	L	L	A	•	•	•	•	•	•	•
41	–	–	–	H	L	H	L	L	H	L	H	L	H	L	L	L	•	A	•	•	•	•	•	•
42	–	–	–	H	L	H	L	L	H	L	H	L	H	H	L	L	•	•	A	•	•	•	•	•
43	–	–	–	H	L	H	L	L	H	L	H	H	L	L	L	L	•	•	•	A	•	•	•	•
44	–	–	–	H	L	H	L	L	H	L	H	H	L	H	L	L	•	•	•	•	A	•	•	•
45	–	–	–	H	L	H	L	L	H	L	H	H	H	L	L	L	•	•	•	•	•	A	•	•
46	–	–	–	H	L	H	L	L	H	L	H	H	H	H	L	L	•	•	•	•	•	•	A	•
47	–	–	–	H	L	H	L	L	H	H	L	L	L	L	L	L	•	•	•	•	•	•	•	A

Figure 2.

FPLA PROGRAM FOR 1μS PULSE WINDOWS UP TO 50μS.

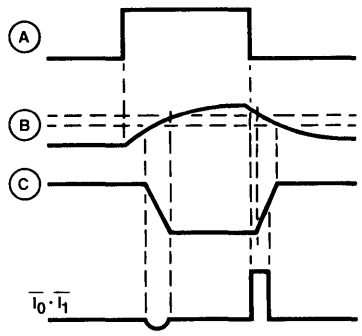


Figure 3.

TIMING DECODER & SEQUENCE CONTROLLER FOR DATA ACQUISITION SYSTEM

The functional diagram of Figure 1 shows a data acquisition peripheral which samples four analog signals and sequentially supplies digitized data to a minicomputer. Computer commands vary sample timing. Four digitized samples are sent to the computer via interrupts every one millisecond.

The system operates with a basic repetition interval of 1Ms. Sample acquisition time begins at the start of each 1Msec interval, and samples are held at various times after the beginning of the interval. Sample No. 1 is always held at 200 μsec into the interval. Sample No. 2 follows No. 1 by 2.5 μsec. The timing of Samples 3 and 4 are varied before and after sample 1 in 50 μsec steps by a 3-bit command from the computer. Sample 4 follows Sample 3 by 2.5 μsec. The timing commands and the resulting Sample 3 hold times (from start of 1Ms) are:

Command	0	1	2	3	4	5	6	
Sample 3 Time	50	100	150	200	250	300	350	. . . (μsec)

As shown in the circuit diagram of Figure 2, the counter outputs and the command bits are decoded in FPLA No. 1 to provide turn-off signals to the Sample & Hold gate flip-flops. A 16-state sequential controller operates the multiplexers, A/D converter and interfaces with the computer interrupt system. The controller is implemented in FPLA No. 2 and is expanded into the remaining parts of FPLA No. 1 because more than 8 outputs are required. Both FPLA Program Tables and I/O assignment are tabulated in Figures 4 and 5, respectively.

The Sequence Controller decodes counter outputs at the end of the 1 Msec interval and resets the counter synchronously. The Sample gate flip-flops are also turned on. The controller then waits until Sample 1 is held and starts the A/D Converter pulsing an output for one cycle. When the converter is no longer busy, the controller sets the interrupt output active until the computer recognizes it, and responds by reading the A/D output data. The multiplexer is advanced to Sample 2 and the process is repeated. The A/D conversion takes longer than 2.5 μsec, so Sample 2 can be converted immediately. Conversion of Sample 2 may finish before Sample 3 is held in the latest cases, so the sequencer may have to wait before converting. After the conversion and interrupt sequence is completed for all four samples, the sequencer waits for the end of the 1 Msec interval. One of the multiplexer switches is active at all times. A flowchart

of the control sequence is shown in Figure 3. The system interface is through the following signals:

Clock: 2 MHz TTL square wave.

Analog Inputs: F_C = 100 KHz, 30 KHz Bandwidth Amplitude: ± 1 volt.

Data from Computer: 3 bits of command data and a strobe. The strobe is pulsed to load the command register after the computer has responded to all four interrupts in an interval.

Data to Computer: The A/D Converter provides 10 bits of parallel data in TTL compatible levels.

Interrupt: Low TTL level active while the controller has valid data for the computer to read.

Response to Interrupt: A 2 μsec active-low pulse after the computer has read the A/D data.

FUNCTIONAL DIAGRAM OF DATA ACQUISITION SYSTEM

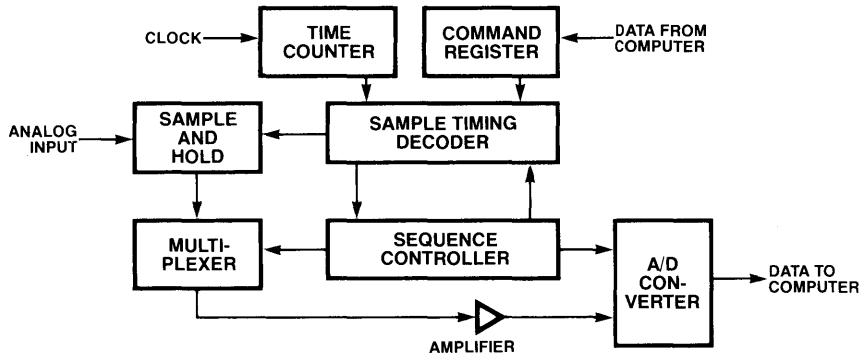


Figure 1.

CIRCUIT DIAGRAM OF DAS CONTROLLER

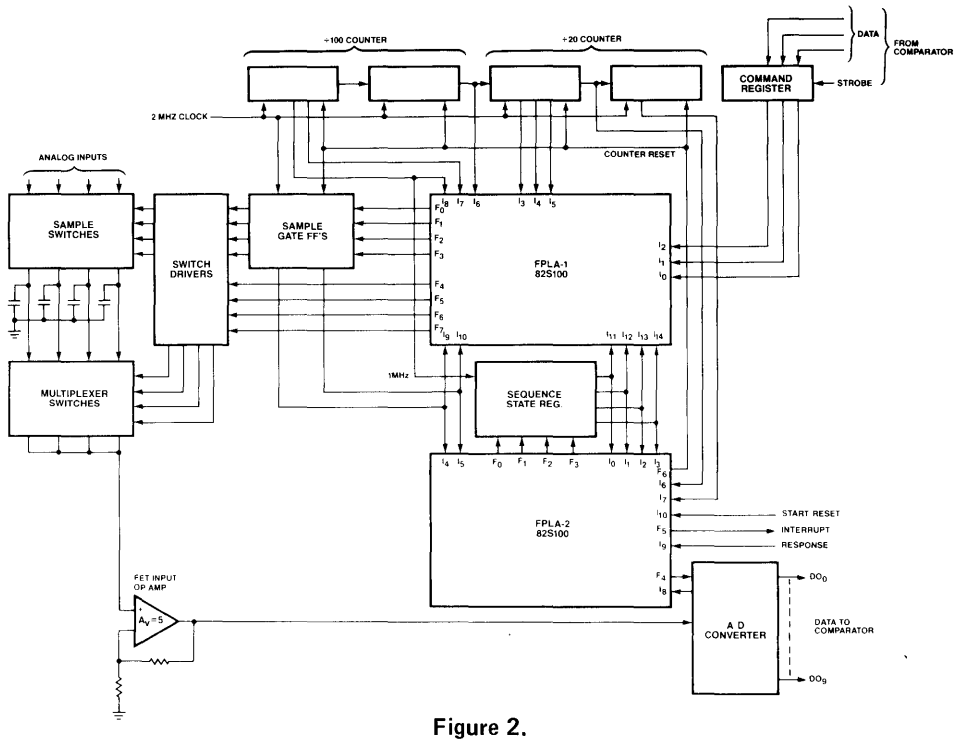


Figure 2.

FLOWCHART OF A/D CONVERSION AND DATA INTERFACE SEQUENCE

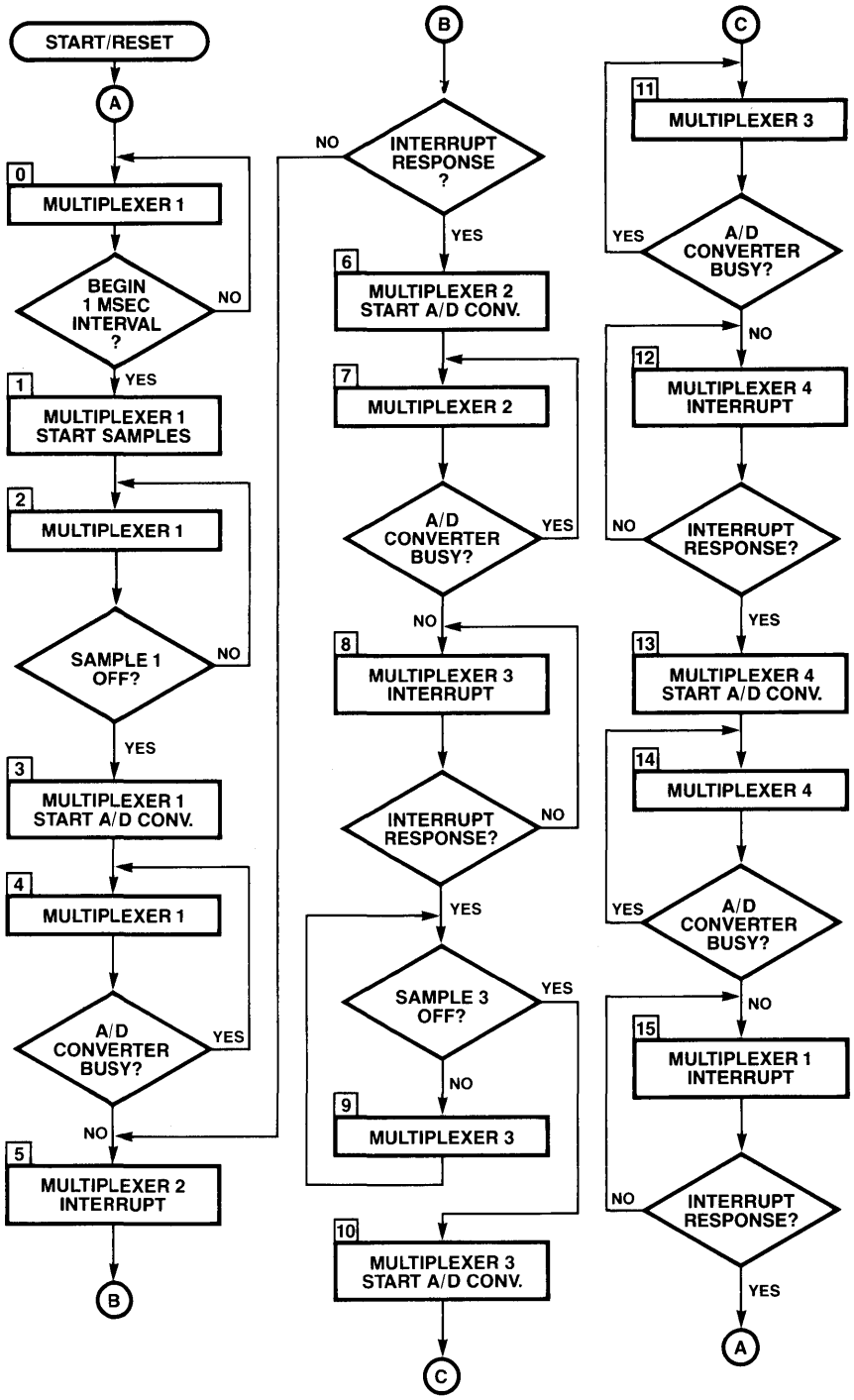


Figure 3.

PROGRAM TABLE FOR FPLA 1

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: L	L	L	L	H	H	H	H
																	OUTPUT FUNCTION: 7	6	5	4	3	2	1	0
0	–	L	L	–	–	–	–	–	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
1	–	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
2	–	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	•	•	•	A	•	•	•	•
3	–	L	H	L	H	–	–	–	–	–	–	–	–	–	–	–	•	•	A	•	•	•	•	•
4	–	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	•	A	•	•	•	•	•
5	–	H	L	–	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
6	–	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
7	–	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
8	–	–	–	–	–	–	L	H	H	–	–	–	–	–	–	–	•	•	•	•	•	•	A	•
9	–	–	–	–	–	L	–	H	H	–	–	–	–	–	–	–	•	•	•	•	A	•	•	•
10	–	–	–	–	–	–	–	–	–	H	L	H	H	–	–	–	•	•	•	•	•	•	•	A
11	–	–	–	–	–	–	–	–	–	H	L	L	L	L	L	L	•	•	•	•	•	A	•	•
12	–	–	–	–	–	–	–	–	–	H	L	L	H	L	L	H	•	•	•	•	•	A	•	•
13	–	–	–	–	–	–	–	–	–	H	L	H	L	L	H	L	•	•	•	•	•	A	•	•
14	–	–	–	–	–	–	–	–	–	H	L	H	H	L	H	H	•	•	•	•	•	A	•	•
15	–	–	–	–	–	–	–	–	–	H	H	L	L	H	L	L	•	•	•	•	•	A	•	•
16	–	–	–	–	–	–	–	–	–	H	H	L	H	H	L	H	•	•	•	•	•	A	•	•
17	–	–	–	–	–	–	–	–	–	H	H	H	L	H	H	L	•	•	•	•	•	A	•	•

I/O ASSIGNMENT	0-2: Command Register bits. 3-5: Count of 50 μsec intervals from ÷ 20 counter. 6: .5 μsec pulse every 50 μsec. 7: 2 μsec output from ÷ 100 counter. 8: .5 μsec output from ÷ 100 counter. This is also the signal used to clock the sequence state register. 9-10: Sample 1 and 3 gates. 11-14: Four sequence state bits.	0-3: Four sample Gate turn-off signals. 4-7: Four multiplexer control signals.

Figure 4.

PROGRAM TABLE FOR FPLA-2

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	L	L	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–	A	A	•	•	•	•	•	A
1	–	–	–	–	–	L	–	–	H	H	–	–	L	L	L	L	A	A	•	•	•	•	•	A
2	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	A	A	•	•	•	•	A	•
3	–	–	–	–	–	L	–	–	–	–	–	H	L	L	H	L	A	•	•	•	•	•	A	•
4	–	–	–	–	–	L	–	–	–	–	–	L	L	L	H	L	A	•	•	A	•	•	A	A
5	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	H	A	•	•	A	•	A	•	•
6	–	–	–	–	–	L	–	H	–	–	–	–	L	H	L	L	A	•	•	•	•	A	•	•
7	–	–	–	–	–	L	–	L	–	–	–	–	L	H	L	L	A	•	A	•	•	A	•	A
8	–	–	–	–	–	L	H	–	–	–	–	–	L	H	L	H	A	•	A	•	•	A	•	A
9	–	–	–	–	–	L	L	L	–	–	–	–	L	H	L	H	A	•	•	A	•	A	A	•
10	–	–	–	–	–	L	–	–	–	–	–	–	L	H	H	L	A	•	•	A	•	A	A	A
11	–	–	–	–	–	L	–	H	–	–	–	–	L	H	H	H	A	•	•	•	•	A	A	A
12	–	–	–	–	–	L	–	L	–	–	–	–	L	H	H	H	A	•	A	•	A	•	•	•
13	–	–	–	–	–	L	H	–	–	–	–	–	H	L	L	L	A	•	A	•	A	•	•	•
14	–	–	–	–	–	L	L	–	–	–	H	–	H	L	L	L	A	•	•	•	A	•	•	A
15	–	–	–	–	–	L	L	–	–	–	L	–	H	L	L	L	A	•	•	A	A	•	A	•
16	–	–	–	–	–	L	–	–	–	–	H	–	H	L	L	H	A	•	•	•	A	•	•	A
17	–	–	–	–	–	L	–	–	–	–	L	–	H	L	L	H	A	•	•	A	A	•	A	•
18	–	–	–	–	–	L	–	–	–	–	–	–	H	L	H	L	A	•	•	A	A	•	A	A
19	–	–	–	–	–	L	–	H	–	–	–	–	H	L	H	H	A	•	•	•	A	•	A	A
20	–	–	–	–	–	L	–	L	–	–	–	–	H	L	H	H	A	•	A	•	A	A	•	•
21	–	–	–	–	–	L	H	–	–	–	–	–	H	H	L	L	A	•	A	•	A	A	•	•
22	–	–	–	–	–	L	L	–	–	–	–	–	H	H	L	L	A	•	•	A	A	A	•	A
23	–	–	–	–	–	L	–	–	–	–	–	–	H	H	L	H	A	•	•	A	A	A	A	•
24	–	–	–	–	–	L	–	H	–	–	–	–	H	H	H	L	A	•	•	•	A	A	A	•
25	–	–	–	–	–	L	–	L	–	–	–	–	H	H	H	L	A	•	A	•	A	A	A	A
26	–	–	–	–	–	L	H	–	–	–	–	–	H	H	H	H	A	•	A	•	A	A	A	A

I/O ASSIGNMENT	0-3: Four sequence state bits. 4-5: Sample 1 and 3 gates. 6-7: ÷ 20 counter outputs to decode 1 Msec. 8: A/D Converter busy. 9: Computer interrupt response. 10: Start/Reset signal from computer.	0-3: Four sequence state bits for next step. 4: A/D Converter start pulse-high logic level for 1 μsec. 5: Interrupt to computer. 6: Reset signal to counter and Sample Gates

Figure 5.

One application where the FPLA proves ideal is in the control for time-division multiplexing a data channel among a number of data sources according to a fixed schedule, or a small number of selectable fixed schedules.

In the system shown in Figure 1 there are two identical instruments providing data words upon demand. The proportion of the channel capacity allocated to each data source depends on the frequency with which that information is needed for the eventual data evaluation process. In this case there are three different formats which may be selected, depending on the currently available channel capacity, which requires skewing the ideal data word mix to neatly fit the aggregate bit rate. A further variation allows the choice of ignoring one of the two instruments, and devote the entire data output to the other instrument to allow higher resolution measurements to be performed. These functions are summarized in the Major Frame logic truth-table of Figure 2, and incorporated in two FPLAs programmed as in Figures 3 and 4.

Each half-second increment in the format represents one complete measurement cycle for the instrument. Additional FPLAs can be used to implement a Minor Frame logic function for decoding time increments within that half-second interval to flag the start of new operating modes (zero-check, calibrate, gain-range, sweep, etc.), initiate old converter sequences, etc.

The outputs of the FPLAs for both Major and Minor logic decoding functions are logically ANDed together to produce the actual enable signals which connect a given data source to the output bus. This output bus then feeds the input of a first-in/first-out memory to smooth the data flow for application to the uniform-rate synchronous data channel. Applied to an actual working design, the two FPLAs for the Major Frame logic replaced 11 MSI and SSI logic packages.

BLOCK DIAGRAM OF SINGLE TDM DATA CHANNEL

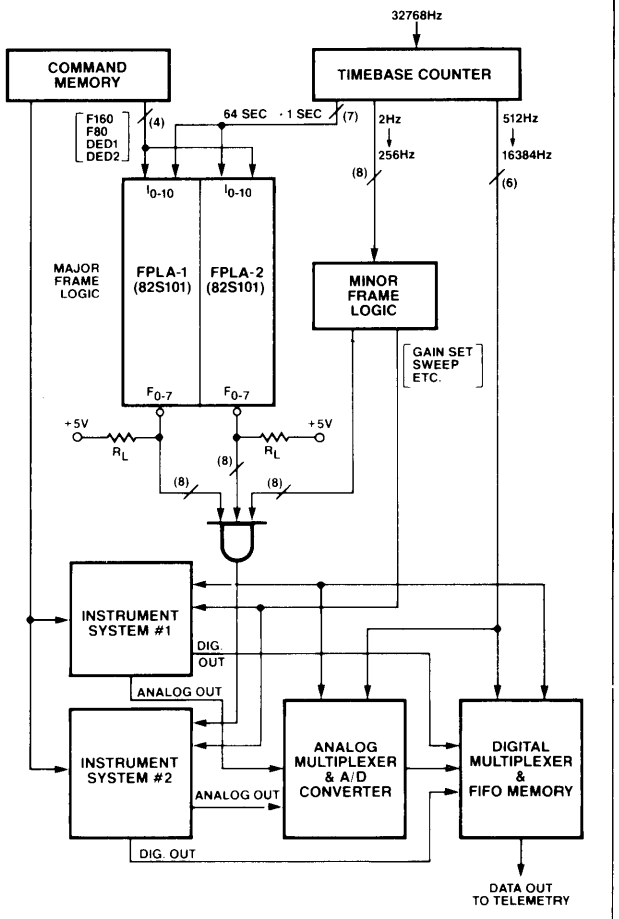


Figure 1.

FUNCTIONAL TRUTH TABLE
FOR MAJOR FRAME LOGIC

F160	F80	Function
0	0	128 BPS Format
0	1	80 BPS Format
1	0	160 BPS Format
1	1	not allowed

DED 1	DED 2	Function
0	0	Timeshared Format
0	1	TM dedicated to System 2
1	0	TM dedicated to System 1
1	1	not allowed

Figure 2.

PROGRAM TABLE FOR FPLA 1 IN MAJOR FRAME LOGIC

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																L	L	L	H	L	L	L	L
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	L	L	L	L	–	–	–	–	H	L	L	•	•	•	A	•	•	•	A
1	–	–	–	–	–	L	L	L	L	L	–	–	L	L	H	L	•	•	•	A	•	•	•	A
2	–	–	–	–	–	L	L	L	L	H	–	–	H	L	H	L	•	•	•	A	•	•	•	A
3	–	–	–	–	–	L	L	–	–	–	–	–	–	H	H	L	•	•	•	A	•	•	A	•
4	–	–	–	–	–	L	L	L	L	–	–	–	L	L	H	L	•	•	•	A	•	•	A	•
5	–	–	–	–	–	L	L	L	H	–	–	–	H	L	H	L	•	•	•	A	•	•	A	•
6	–	–	–	–	–	L	–	–	–	–	–	–	–	H	H	L	•	•	•	A	•	A	•	•
7	–	–	–	–	–	L	L	–	–	–	–	–	L	L	H	L	•	•	•	A	•	A	•	•
8	–	–	–	–	–	L	H	–	–	–	–	–	H	L	H	L	•	•	•	A	•	A	•	•
9	–	–	–	–	–	L	L	L	L	L	L	L	–	H	–	L	•	•	•	A	A	•	•	•
10	–	–	–	–	–	L	L	L	L	L	L	L	L	L	–	L	•	•	•	A	A	•	•	•
11	–	–	–	–	–	–	L	L	L	L	L	H	–	H	–	L	•	•	•	A	•	•	•	•
12	–	–	–	–	–	–	L	L	L	L	L	H	L	L	–	L	•	•	•	A	•	•	•	•
13	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L	•	•	A	A	•	•	•	•
14	–	–	–	–	–	–	L	–	–	–	–	–	H	–	–	L	•	A	•	A	•	•	•	•
15	–	–	–	–	–	–	H	–	–	–	–	–	L	–	–	L	•	A	•	A	•	•	•	•
16	–	–	–	–	–	–	L	L	L	L	L	L	–	–	–	L	A	•	•	A	•	•	•	•
17	–	–	–	–	–	L	L	L	–	–	–	–	–	H	L	–	•	•	•	A	•	•	•	A
18	–	–	–	–	–	L	L	L	L	–	–	–	L	L	L	H	•	•	•	A	•	•	•	A
19	–	–	–	–	–	L	L	L	H	–	–	–	H	L	L	H	•	•	•	A	•	•	•	A
20	–	–	–	–	–	L	L	–	–	–	–	–	–	H	L	–	•	•	•	A	•	•	A	•
21	–	–	–	–	–	L	L	L	–	–	–	L	–	L	L	H	•	•	•	A	•	•	A	•
22	–	–	–	–	–	L	L	H	–	–	–	H	–	L	L	H	•	•	•	A	•	•	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	•	•	•	A	•	A	•	•
24	–	–	–	–	–	L	L	L	L	L	L	–	–	H	L	H	•	•	•	A	A	•	•	•
25	–	–	–	–	–	L	L	L	L	L	L	L	–	L	L	H	•	•	•	A	A	•	•	•
26	–	–	–	–	–	L	L	L	L	L	L	H	–	L	L	H	•	•	•	A	•	•	•	•
27	–	–	–	–	–	L	–	–	–	–	–	–	–	H	–	–	•	•	A	A	•	•	•	•
28	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	•	•	A	A	•	•	•	•
29	–	–	–	–	–	–	L	–	–	–	–	H	–	–	L	H	•	A	•	A	•	•	•	•
30	–	–	–	–	–	–	H	–	–	–	–	L	–	–	L	H	•	A	•	A	•	•	•	•
31	–	–	–	–	–	–	L	L	L	L	L	–	–	–	L	H	A	•	•	A	•	•	•	•
32	–	–	–	–	–	L	L	L	L	–	–	L	–	L	L	L	•	•	•	A	•	•	•	A
33	–	–	–	–	–	L	L	L	H	–	–	H	–	L	L	L	•	•	•	A	•	•	•	A
34	–	–	–	–	–	L	L	L	–	–	–	–	L	L	L	L	•	•	•	A	•	•	A	•
35	–	–	–	–	–	L	L	H	–	–	–	–	H	L	L	L	•	•	•	A	•	•	A	•
36	–	–	–	–	–	L	–	–	–	–	–	L	–	L	L	L	•	•	A	A	•	•	•	•
I/O ASSIGNMENT	UNUSED					DED 2	1 Sec	2 Sec	4 Sec	8 Sec	16 Sec	32 Sec	64 Sec	DED 1	F80	F160	HEN	CNARR	SC1SAM	unused	SC1EN	DV1EN	NI1EN	VO1EN

Figure 3.

PROGRAM TABLE FOR FPLA 2 IN MAJOR FRAME LOGIC

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	L	L	H	L	L	L
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	L	L	L	L	–	–	–	–	H	L	L	A	A	•	•	A	•	•	A
1	–	–	–	–	–	L	L	L	L	L	–	–	L	L	H	L	A	A	•	•	A	•	•	A
2	–	–	–	–	–	L	L	L	L	H	–	–	H	L	H	L	A	A	•	•	A	•	•	A
3	–	–	–	–	–	L	L	L	–	–	–	–	–	H	H	L	A	A	•	•	A	•	A	•
4	–	–	–	–	–	L	L	L	L	–	–	–	L	L	H	L	A	A	•	•	A	•	A	•
5	–	–	–	–	–	L	L	L	H	–	–	–	H	L	H	L	A	A	•	•	A	•	A	•
6	–	–	–	–	–	L	–	–	–	–	–	–	–	H	H	L	A	A	•	•	A	A	•	•
7	–	–	–	–	–	L	L	–	–	–	–	–	L	L	H	L	A	A	•	•	A	A	•	•
8	–	–	–	–	–	L	H	–	–	–	–	–	H	L	H	L	A	A	•	•	A	A	•	•
9	–	–	–	–	–	L	L	L	L	L	L	L	–	H	–	L	A	A	•	•	A	•	•	•
10	–	–	–	–	–	L	L	L	L	L	L	L	L	L	–	L	A	A	•	•	A	•	•	•
11	–	–	–	–	–	–	L	L	L	L	L	H	–	H	–	L	A	A	•	A	A	•	•	•
12	–	–	–	–	–	–	L	L	L	L	L	H	L	L	–	L	A	A	•	A	A	•	•	•
13	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L	A	A	A	•	A	•	•	•
14	–	–	–	–	–	–	–	–	–	–	–	–	H	–	–	L	A	A	•	•	A	•	•	•
15	–	–	–	–	–	H	–	–	–	–	–	–	L	–	–	L	A	A	•	•	A	•	•	•
16	–	–	–	–	–	L	L	L	L	L	L	L	–	–	–	L	A	A	•	•	A	•	•	•
17	–	–	–	–	–	L	L	L	–	–	–	–	–	H	L	–	A	A	•	•	A	•	•	A
18	–	–	–	–	–	L	L	L	L	–	–	–	L	L	L	H	A	A	•	•	A	•	•	A
19	–	–	–	–	–	L	L	L	H	–	–	–	H	L	L	H	A	A	•	•	A	•	•	A
20	–	–	–	–	–	L	L	–	–	–	–	–	–	H	L	–	A	A	•	•	A	•	A	•
21	–	–	–	–	–	L	L	L	–	–	–	L	–	L	L	H	A	A	•	•	A	•	A	•
22	–	–	–	–	–	L	L	H	–	–	–	H	–	L	L	H	A	A	•	•	A	•	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	A	A	•	•	A	A	•	•
24	–	–	–	–	–	L	L	L	L	L	L	–	–	H	L	H	A	A	•	A	A	•	•	•
25	–	–	–	–	–	L	L	L	L	L	L	L	–	L	L	H	A	A	•	•	A	•	•	•
26	–	–	–	–	–	L	L	L	L	L	L	H	–	L	L	H	A	A	•	A	A	•	•	•
27	–	–	–	–	–	L	–	–	–	–	–	–	–	H	–	–	A	A	A	•	A	•	•	•
28	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	A	A	A	•	A	•	•	•
29	–	–	–	–	–	–	L	–	–	–	–	H	–	–	L	H	A	A	•	•	A	•	•	•
30	–	–	–	–	–	–	H	–	–	–	–	L	–	–	L	H	A	A	•	•	A	•	•	•
31	–	–	–	–	–	–	L	L	L	L	L	–	–	–	L	H	A	A	•	•	A	•	•	•
32	–	–	–	–	–	L	L	L	L	–	–	L	–	L	L	L	A	A	•	•	A	•	•	A
33	–	–	–	–	–	L	L	L	H	–	–	H	–	L	L	L	A	A	•	•	A	•	•	A
34	–	–	–	–	–	L	L	L	–	–	–	–	L	L	L	L	A	A	•	•	A	•	A	•
35	–	–	–	–	–	L	L	H	–	–	–	–	H	L	L	L	A	A	•	•	A	•	A	•
36	–	–	–	–	–	L	–	–	–	–	–	L	–	L	L	L	A	A	A	•	A	•	•	•
I/O ASSIGNMENT	UNUSED					DED 1	1 Sec	2 Sec	4 Sec	8 Sec	16 Sec	$\overline{32}$ Sec	$\overline{64}$ Sec	DED 2	F 80	F 160	unused		SC2 SAM	SC2 EN	unused	DV2 EN	NI2 EN	VO2 EN

Figure 4.

Until now designers of ATC (Air Traffic Control) transponder encoders have had little choice other than stacking up numerous stages of shift registers and banks of gates, or go the route of custom LSI. The former choice involves close to two dozen chips, while the second entails a large initial expense for masks. The design of Figure 1 uses an FPLA to solve the problem of converting large amounts (25 lines total) of parallel data into unique combinations of serial bit streams using only six commercially available chips. While the full capability of the FPLA's is not utilized, the design represents considerable savings over previous options, and with only slight modifications can be used in other parallel to serial data conversion applications.

With reference to Figure 1, the start command resets A4. The outputs of A4 start the external clock and allow counters A2 and A5 to toggle. A2 and A5 count until maximum count (19) is decoded by A1, which in turn sets A4, thereby stopping the cycle and resetting the counters. During the first 16 clock pulses, either the Mode-A or Mode-C programming is outputted from A3 or A6 respectively, depending on the state of the MODE CONTROL as gated through A7, A8 and A9. The special Mode-A IDENT pulse is decoded on the eighteenth (18th) clock pulse and enabled all through gates A10 and A11. A12 gates the output with the clock to eliminate ripple-through spikes and race problems.

ATC TRANSPONDER ENCODER CIRCUIT DIAGRAM

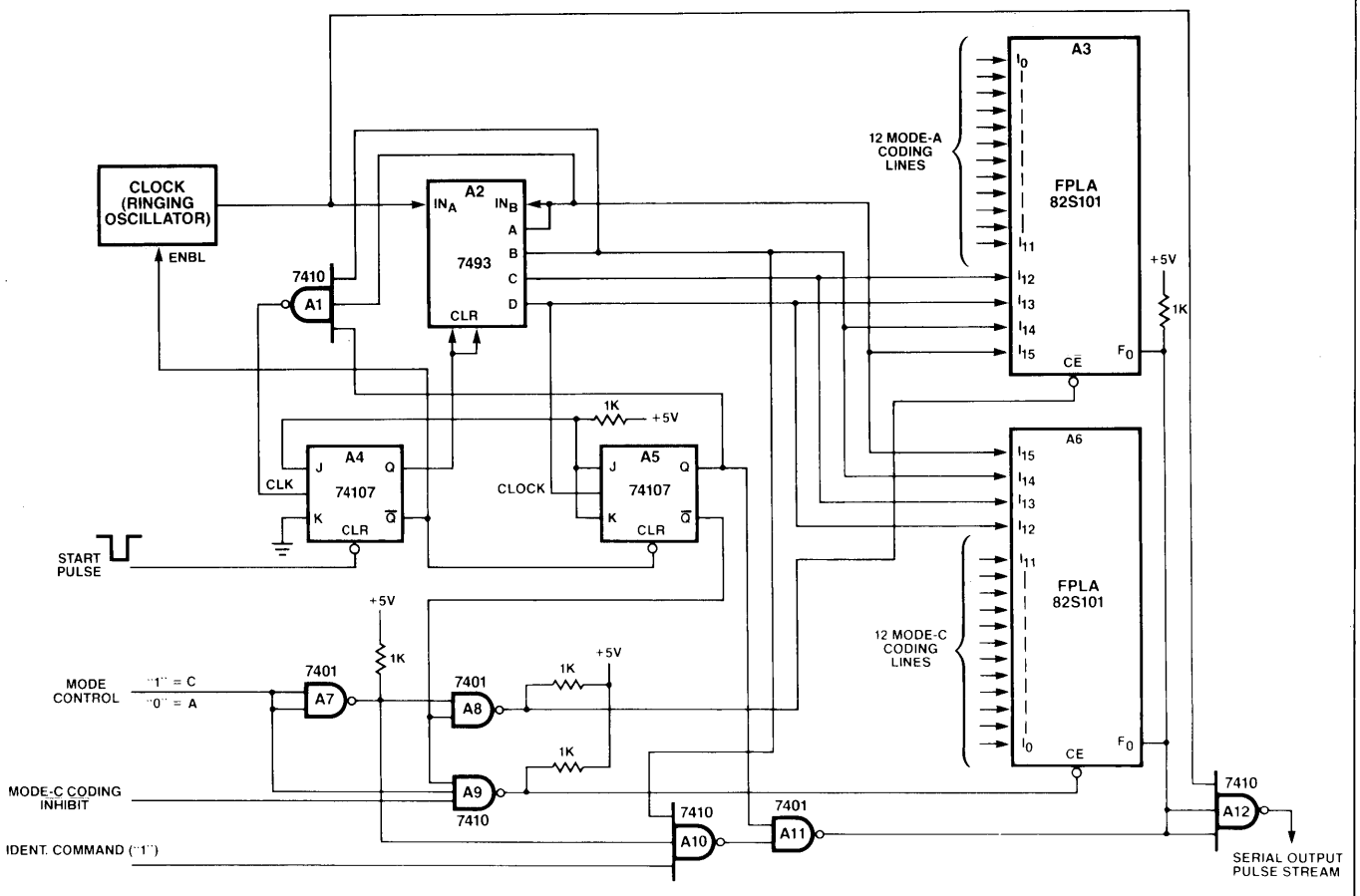


Figure 1.

In many applications it is necessary to have multi channel two way radiotelephone for mobile communications. However, in emergency situations it is important to be able to change channels rather quickly, especially when the transceiver is mounted in a vehicle where the driver cannot look at the radio channel selector switch and the driveway at the same time. This situation is analogous to the car radio stations. Channel selection can be made easier by incorporating in the transceiver a Frequency Synthesizer with a Field Programmable Logic Array as a Channel Memory, in which the channels are programmed digitally and are selected by a keyboard that has the channels designations JJ, JL, YL, . . . etc.

The block diagram of Figure 1 shows a Phase-Locked Channel Selector Synthesizer using a Hughes HCTR0320 Frequency Synthesizer integrated circuit operating at low frequencies (1-3 MHz), together with a Voltage Controlled Oscillator (VCO) and a low pass filter for closing the loop.

The HCTR0320 chip has a programmable counter/divider, with BCD and binary inputs, which sets the output frequency to an exact multiple of the reference frequency. The reference frequency is generated by a stable crystal controlled oscillator and divided down by a prescaler (counter-divider) for supplying the 30 KHz channel spacing necessary for the VHF band.

An 82S101 FPLA programmed with different settings of the programmable counter fixes the channel frequencies, and is addressed by a keyboard that selects a channel by pressing one of its keys. Part of the binary output of this FPLA is used to set the programmable counter in the HCTR0320 chip. A second 82S101 FPLA is used as an optional binary to BCD converter for addressing the BCD inputs of the programmable counter in the Frequency Synthesizer chip.

An interface is placed between the keyboard and the FPLA Channel Memory, to provide contact debounce and for changing transmit and receive mode. This permits the use of one crystal for generating both transmitting frequencies and

FREQUENCY SYNTHESIZED MOBILE FM-VHF TRANCEIVER
USING DIGITAL MEMORY FOR QUICK CHANNEL SELECTION

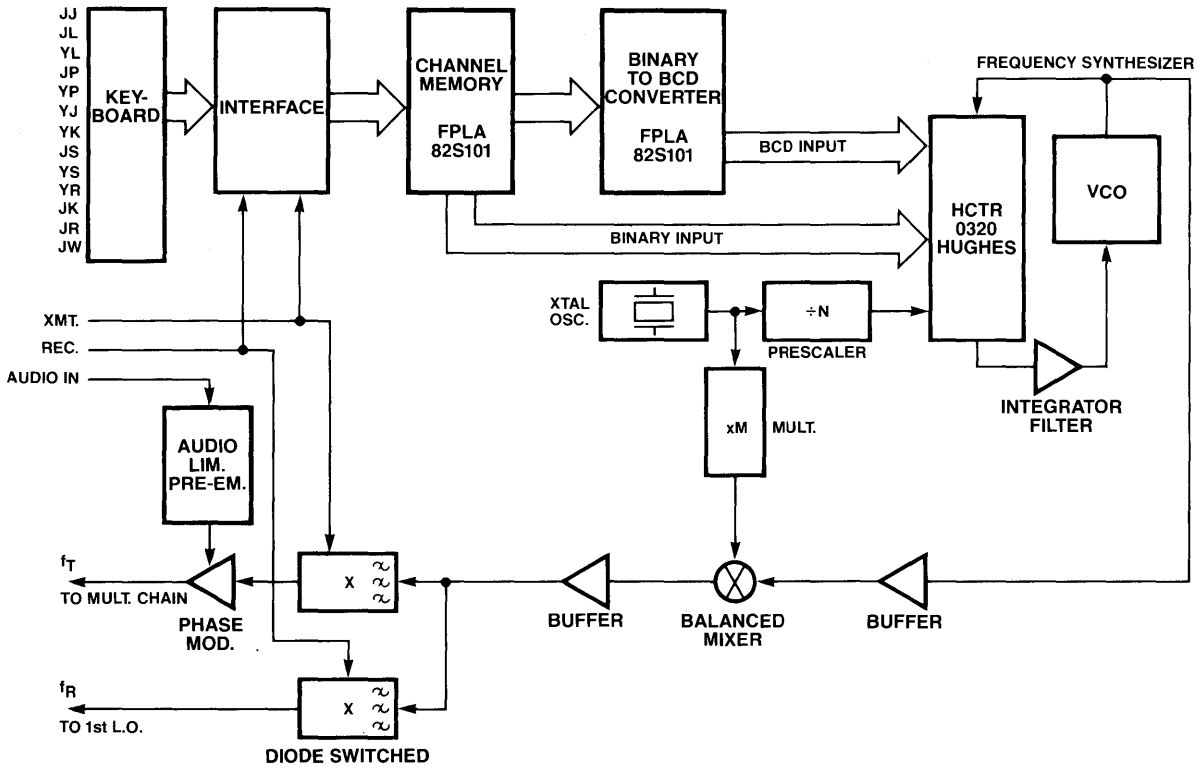


Figure 1.

the local oscillator frequencies for the receiver mixer. This can be done by mixing the signal from the crystal oscillator reference, previously multiplied, with the selected channel frequency from the synthesizer. A balanced mixer with diode switched bandpass filters, one for transmit and the other for receive is used. These are selected by a PTT switch or other suitable means.

In the transmit mode, the synthesized channel frequency goes thru a phase modulator that receives the audio signal from a limiter/pre-emphasis network. The output of the phase modulator is fed to a standard multiplier chain.

In the receive mode, the frequency generated f_R is used as a local oscillator for the receiver mixer ($f_R = F_T + f_{IF}$).

As the HCTR0320 can generate 1021 different frequencies/channels (in the programmable counter $3 \leqslant N \leqslant 1023$), it is possible to choose the Frequency Synthesizer output with its 13 channels centered in a convenient range as:

$$(3) \times 30\ KHz \leqslant f_{VCO} \leqslant 30\ KHz \times (1023)$$
$$90\ KHz \leqslant f_{VCO} \leqslant 30{,}690\ KHz$$

The choice of this center frequency is determined by the VCO available, quality of the mixer and the crystal frequency.

As noted, the second FPLA is optional, depending if the BCD inputs to the programmable counter are used or not, as this determines the high frequency operation of the Frequency Synthesizer. Alternately, these BCD inputs could be strapped such that the operating frequency of the VCO is around some desirable value.

The proposed Frequency Synthesizer with Channel Memory could form the basic building block for any two way radio telephone, as it has the definitive advantages of single crystal operation, and quick selection of the desired communication channel. The number of channels can be extended up to the limit set by the frequency synthesizer and the FPLA capacity.

The circuit of Figure 2 uses FPLA's in a programmable divider for a frequency synthesizer, as part of a 23 channel CB transceiver using a 10.7 MHz IF frequency. There are spare resources in the two FPLA's to go to the new 40 channels, once the frequencies of the additional channels are established by the FCC.

The block diagram of the frequency synthesizer is shown in Figure 1.

The scheme utilizes a divide by 20/21 prescaler such that the total divide ratio is N = (Px20) +Q. Table 1 gives the values of P and Q for both transmit and receive. FPLA #1 decodes when the counters count down to Q+1, and F_1 gives a HIGH output which resets the R-S NOR flip-flop causing the prescaler to divide by 21. When the counter reaches State 1, The F_0 output goes HIGH, providing a HIGH input for the D shift register which drives LOAD. On the next rising edge of the clock, LOAD goes LOW and presets the counter to P_1. Also at this time the ÷21 control is switched back to a divide by 20 mode.

Both Channel Select and Divide Control functions are programmed in the FPLA's as shown in the Program Tables of Figure 3 and 4.

SYNTHESIZER BLOCK DIAGRAM

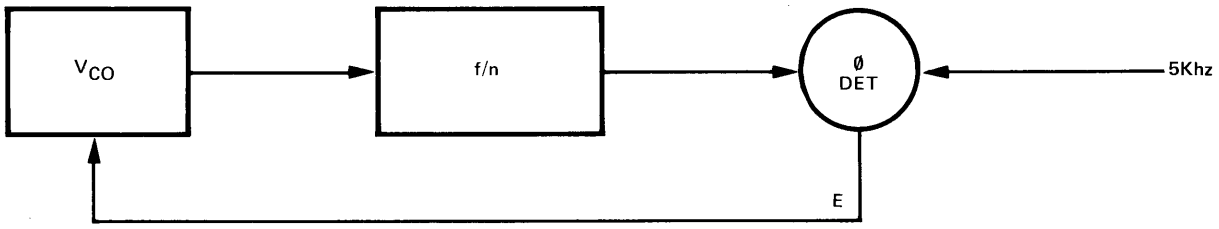


Figure 1.

PROGRAMMABLE FREQUENCY DIVIDER

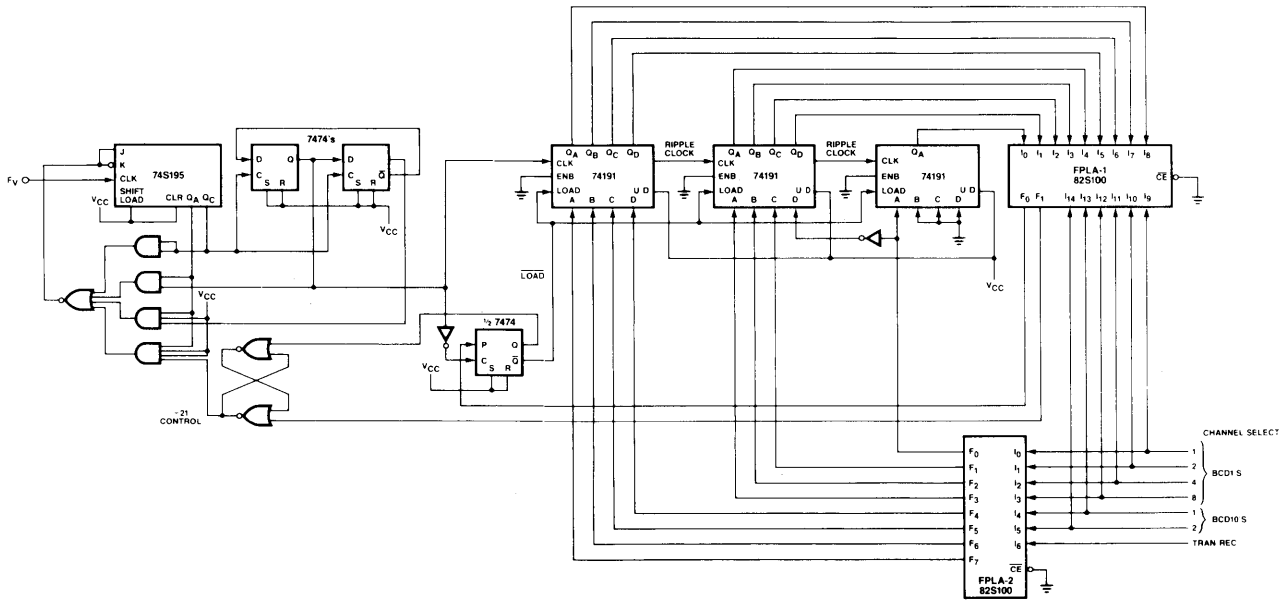


Figure 2.

P AND Q VALUES FOR TRANSMIT AND RECEIVE

Channel #	Frequency	Divide Ratio	P ÷ 20	Q ÷ 21	Frequency	Divide Ratio	P ÷ 20	Q ÷ 21
1	26.965	5393	269	13	16.265	3253	162	13
2	26.975	5395	269	15	16.275	3255	162	15
3	26.985	5397	269	17	16.285	3257	162	17
4	27.005	5401	270	1	16.305	3261	163	1
5	27.015	5403	270	3	16.315	3263	163	3
6	27.025	5405	270	5	16.325	3265	163	5
7	27.035	5407	270	7	16.335	3267	163	7
8	27.055	5411	270	11	16.355	3271	163	11
9	27.065	5413	270	13	16.365	3273	163	13
10	27.075	5415	270	15	16.375	3275	163	15
11	27.085	5417	270	17	16.385	3277	163	17
12	27.105	5421	271	1	16.405	3281	164	1
13	27.115	5423	271	3	16.415	3283	164	3
14	27.125	5425	271	5	16.425	3285	164	5
15	27.135	5427	271	7	16.435	3287	164	7
16	27.155	5431	271	11	16.455	3291	164	11
17	27.165	5433	271	13	16.465	3293	164	13
18	27.175	5435	271	15	16.475	3295	164	15
19	27.185	5432	271	17	16.485	3297	164	17
20	27.205	5441	272	1	16.505	3301	165	1
21	27.215	5443	272	3	16.515	3303	165	3
22	27.225	5445	272	5	16.525	3305	165	5
23	27.255	5451	272	11	16.555	3311	165	11

Table 1.

FPLA-2 PROGRAM TABLE FOR CHANNEL SELECT FUNCTIONS

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	L	L	L	L	L	L	H	•	•	A	•	•	•	A	•
1	–	–	–	–	–	–	–	–	–	L	L	L	L	H	H	–	•	•	A	•	•	•	A	•
2	–	–	–	–	–	–	–	–	–	L	L	L	L	H	–	–	•	•	A	•	•	•	A	A
3	–	–	–	–	–	–	–	–	–	L	L	L	H	L	L	–	•	•	A	•	•	•	A	A
4	–	–	–	–	–	–	–	–	–	L	L	H	L	L	L	–	•	•	A	•	•	•	A	A
5	–	–	–	–	–	–	–	–	–	L	L	H	L	L	H	–	•	•	A	•	•	A	•	•
6	–	–	–	–	–	–	–	–	–	L	L	H	L	H	–	–	•	•	A	•	•	A	•	•
7	–	–	–	–	–	–	–	–	–	L	L	H	H	L	L	–	•	•	A	•	•	A	•	•
8	–	–	–	–	–	–	–	–	–	L	H	L	L	L	–	–	•	•	A	•	•	A	•	A
9	–	–	–	–	–	–	–	–	–	H	L	L	L	L	L	H	A	•	•	•	A	A	•	A
10	–	–	–	–	–	–	–	–	–	H	L	L	L	L	H	–	A	•	•	•	A	A	•	A
11	–	–	–	–	–	–	–	–	–	H	L	L	L	H	–	–	A	•	•	•	A	A	A	•
12	–	–	–	–	–	–	–	–	–	H	L	L	H	L	L	–	A	•	•	•	A	A	A	•
13	–	–	–	–	–	–	–	–	–	H	L	H	L	L	L	–	A	•	•	•	A	A	A	•
14	–	–	–	–	–	–	–	–	–	H	L	H	L	L	H	–	A	•	•	•	A	A	A	A
15	–	–	–	–	–	–	–	–	–	H	L	H	L	H	–	–	A	•	•	•	A	A	A	A
16	–	–	–	–	–	–	–	–	–	H	L	H	H	L	L	–	A	•	•	•	A	A	A	A
17	–	–	–	–	–	–	–	–	–	H	H	L	L	L	–	–	A	•	•	A	•	•	•	•

Figure 3.

FPLA-1 PROGRAM TABLE FOR DIVIDE CONTROL FUNCTION.

SINCE F_{2-7} ARE UNUSED, ALL LINKS IN THAT AREA OF THE DEVICE HAVE BEEN LEFT INTACT

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	L	L	L	L	L	H	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
1	–	L	L	L	L	H	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
2	–	L	L	L	L	H	H	L	H	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
3	–	L	L	L	H	L	L	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
4	–	L	L	L	H	L	H	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
5	–	L	L	L	H	H	L	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
6	–	L	L	L	H	H	H	L	L	L	H	L	L	L	L	L	A	A	A	A	A	A	A	•
7	–	L	L	H	L	L	L	L	L	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
8	–	L	L	H	L	L	H	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
9	–	L	H	L	L	L	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
10	–	L	H	L	L	L	H	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
11	–	L	H	L	L	H	L	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
12	–	L	H	L	L	H	H	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
13	–	L	H	L	H	L	L	L	L	L	H	L	L	L	L	L	A	A	A	A	A	A	A	•
14	–	L	H	L	H	L	H	L	L	H	L	H	L	L	L	L	A	A	A	A	A	A	A	•
15	–	L	H	L	H	H	L	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
16	–	L	H	H	L	L	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
17	–	L	H	H	L	L	H	H	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
18	–	H	L	L	L	L	L	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
19	–	H	L	L	L	L	H	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
20	–	H	L	L	L	H	L	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
21	–	H	L	L	L	H	H	L	L	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
22	–	–	–	–	–	–	–	–	L	L	L	L	L	L	L	–	A	A	A	A	A	A	•	A

UNUSED

Figure 4.

PROGRAMMABLE FREQUENCY SYNTHESIZER 22

The circuit shown in Figure 1 is a keyer and/or V.C.A., as part of a synthesizer. The FPLA is used as a memory to store the shape and amplitude of the envelope for a given selection of the switches. The programmable timer operates in the astable mode with "C_X" and "R_X" determining the frequency of the multivibrator. The FPLA compares the state of the timer/counter with the selection of the switches. Its output goes to the appropriate transmission gate(s) which, in conjunction with the precision resistors, determine the gain of the Op-Amp and the final output frequency envelope. It should be noted that the resistors used must be *precision* or the envelope will lack uniformity. This circuit is very practical in forming waveforms that would be virtually impossible by analog means.

PROGRAMMABLE FREQUENCY SYNTHESIZER

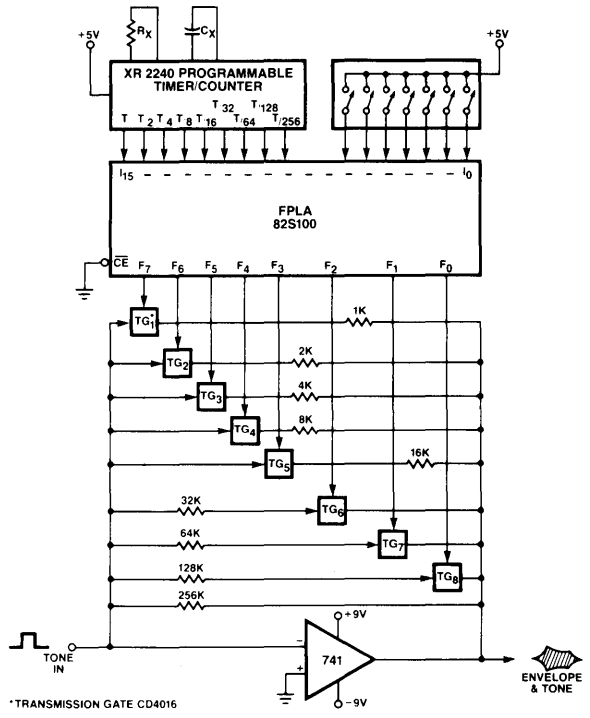


Figure 1.

AN IEEE-488 TO SPEECH SYNTHESIZER INTERFACE 23

The circuit shown in Figure 1 uses an FPLA for interfacing a Telesensory Systems Speech Synthesizer to an IEEE-488 standard instrument bus. This interface will allow blind electronic technicians to "hear" the digital output from digital test equipment equipped with an IEEE-488 interface.

The S15001-A Speech Synthesizer, described in Telesensory Systems Inc., Application Note 003, requires a Data Strobe to clock data on the bus, and it provides a Busy signal to complete the handshake. The inputs to the FPLA are the bus data lines D101 through D107, ATN, DAV and IFC. A power-on Reset input is provided for circuit initialization. Other inputs are the address comparator output, the Busy line from the speech board, the Listen feedback output of the FPLA and the Listen Only switch. The outputs of the FPLA are the bus handshake lines RFD and DAC, Listen, and Data Strobe. Circuit operation is as follows. The power on circuitry initializes the FPLA to a known state, unlisten or listen, depending on the Listen Always switch. Based on the interface bus signals, if the proper listen address is given and ATN is true, the handshake is completed and the Listen line goes true. When ATN is false and Listen is true, the interface accepts each successive byte of data with a Data Strobe signal to the speech board and then waits for the Busy signal from the speech board to go false to complete the bus handshake. The Unlisten command is implemented in FPLA programming; if the character "?" is sent on the data bus with ATN true, the feedback signal Listen goes false.

IEEE-488 STD SPEECH SYNTHESIZER INTERFACE

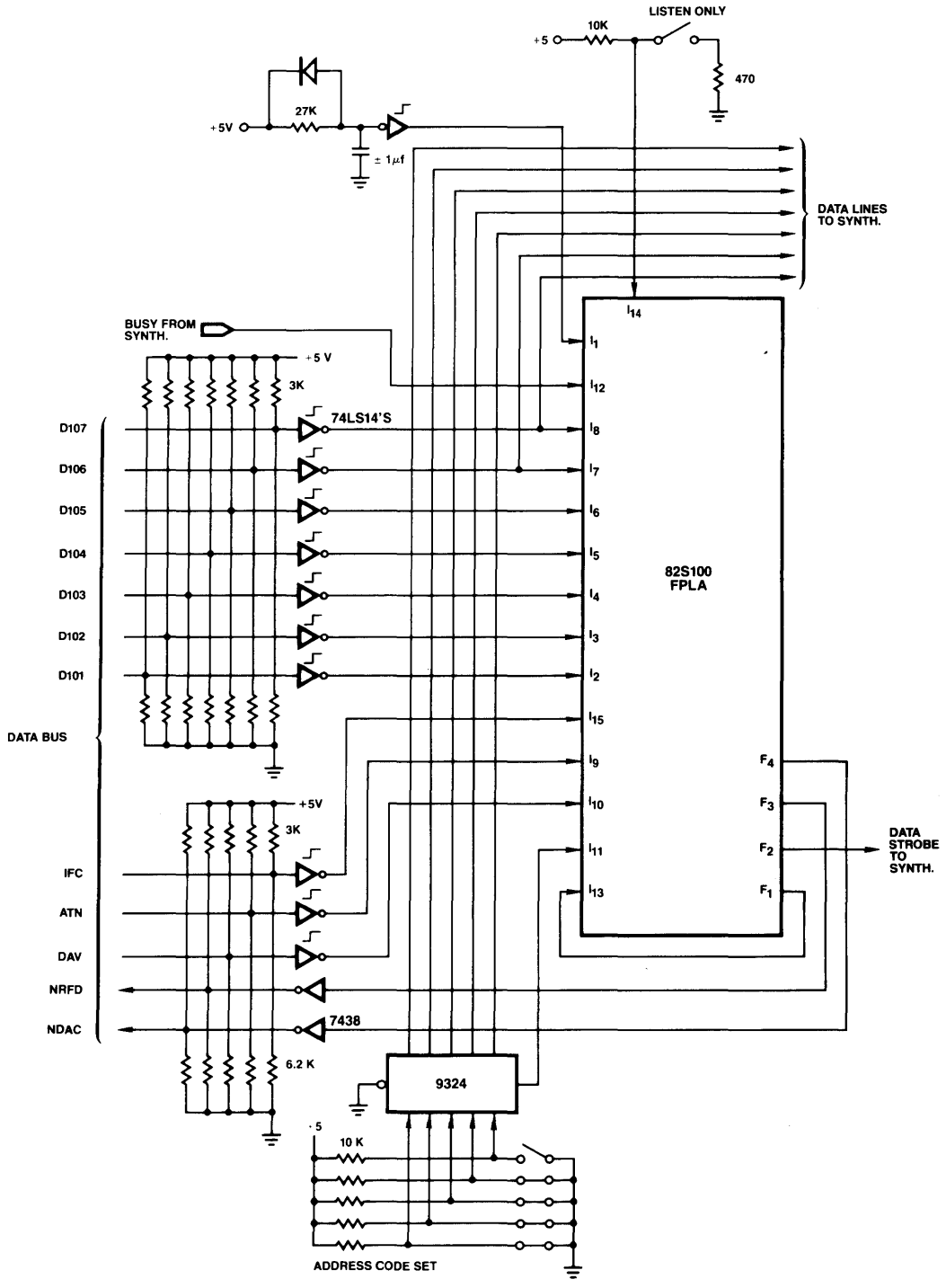


Figure 1.

The circuit shown in Figure 1 is a CAMAC Module for a dual channel, 12-bit A/D converter. CAMAC is an international standard for interfacing modular instrumentation to computers through a bused backplane called the Dataway. Within the context of this application, the Dataway contains the following signals:

SIGNAL	LINES	SIGNAL	LINES
Read Data	24	Station Number	24
Write Data	24	(control station address	
Function	5	to each station)	
Sub Address	4	Look-At-Me (LAM)	24
Status – Q, X	2	(control station interrupt)	
Control and Timing	3	to each station)	
(Z, S1, S2)		Power	±6V, ±24V

There are 32 possible functions and 15 sub addresses in each module. The CAMAC rules state that modules must completely decode all Function and Sub Address lines, and that the X status line must be asserted for every combination of Function and Sub Address implemented by the module. Q is a general purpose status line used primarily for testing conditions in modules. S1 and S2 are 200ns strobe signals which occur in sequence during a Dataway cycle. Z is a general reset function.

Decoding F and A, and generating Q, X, and LAM generally requires from 10 to 20 SSI and MSI packages. An FPLA can replace about 90% of this decoding logic, saving considerable design time and leaving more space for "functional" logic on the fixed size CAMAC printed circuit board.

The module in Figure 1 implements the functions tabulated in Figure 2.

Differing interpretations of the standards have resulted in some CAMAC users preferring A(15) for LAM functions while others prefer A(0). This example illustrates the power of the FPLA In that both options can be provided with no additional hardware.

CAMAC MODULE AD-212 FOR DUAL CHANNEL A/D CONVERTER

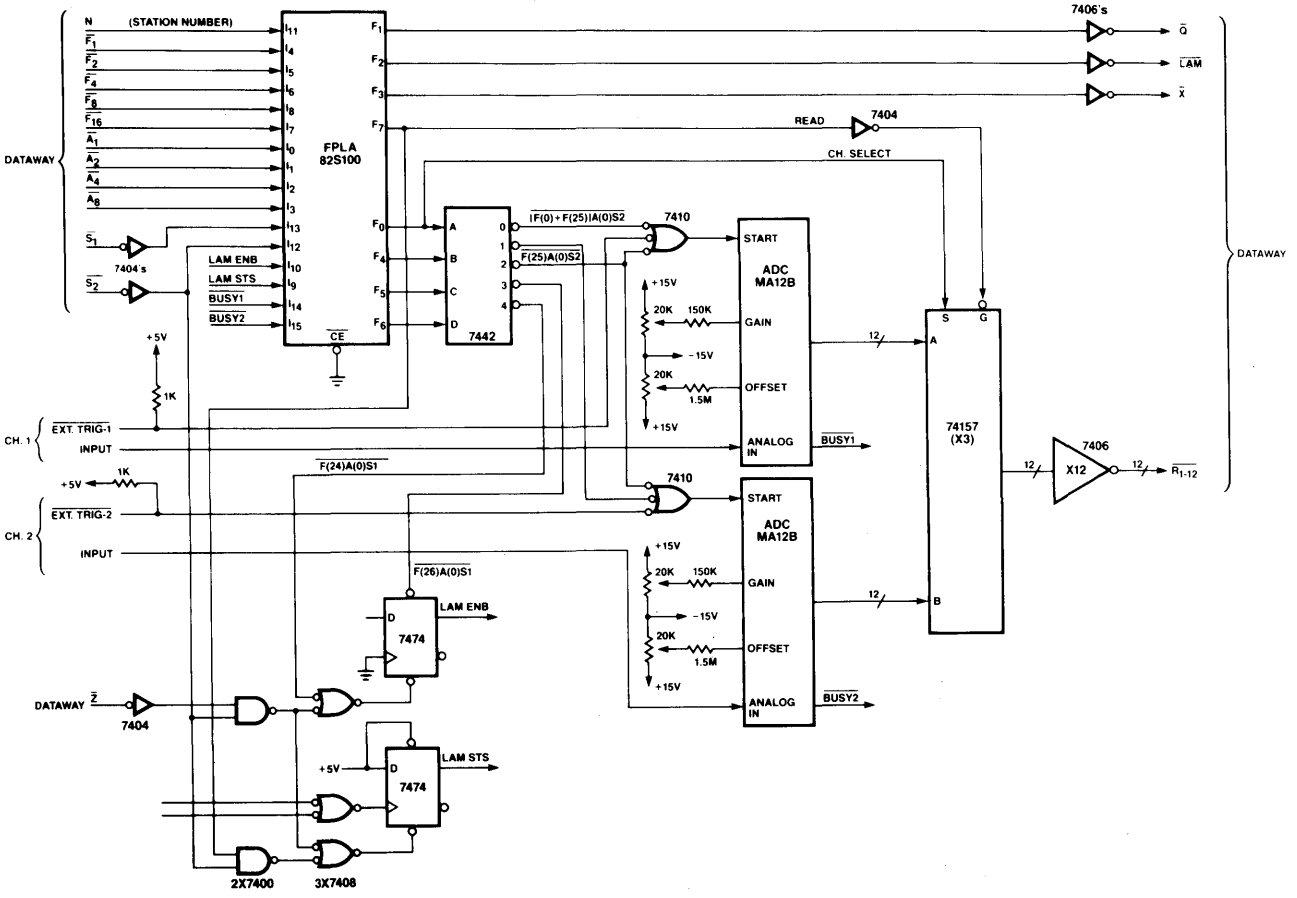


Figure 1.

FUNCTIONS EXECUTED BY CAMAC MODULE

CONDITION	FUNCTION	STATUS
F(0) A(0)	Read Channel 1	Q=$\overline{\text{BUSY 1}}$
F(0) A(1)	Read Channel 2	Q=$\overline{\text{BUSY 2}}$
*F(2) A(0)	Read Channel 1, start new conversion	Q=$\overline{\text{BUSY 1}}$
*F(2) A(1)	Read Channel 2, start new conversion	Q=$\overline{\text{BUSY 2}}$
F(8) A(0)	Test LAM	Q= LAM
F(8) A(15)	Test LAM	Q= LAM
F(24) A(0)	Rest LAM Enable	Q= 1
F(24) A(15)	Reset LAM Enable	Q= 1
*F(25) A(0)	Start Conversion in Channel 1	Q=$\overline{\text{BUSY 1}}$
*F(25) A(1)	Start Conversion in Channel 2	Q=$\overline{\text{BUSY 2}}$
*F(25) A(2)	Start Conversion in both channels	Q=$\overline{\text{BUSY 1} + \text{BUSY 2}}$
F(26) A(0)	Set LAM Enable	Q=1
F(26) A(15)	Set LAM Enable	Q=1
F(27) A(0)	Test Channel 1 BUSY	Q=BUSY 1
F(27) A(1)	Test Channel 2 BUSY	Q=BUSY 2

*For F(2) and F(25) functions, conversion is not started if Q=0.

Figure 2.

PROGRAMMABLE WAVEFORM GENERATOR 25

An FPLA can provide an easy way of generating complex waveforms, whose shape can be quickly tailored to fit each application. In the circuit of Figure 1, a 16-stage binary counter drives the FPLA inputs which are programmed to detect specific counts. At each programmed count the FPLA supplies a corresponding binary weighted output which is summed thru the ladder network to set the desired amplifier gain.

The FPLA program table is derived from a staircase approximation of the desired output waveform, and duty cycle. The frequency is set by the clock. Each cycle is repeated by programming F_7 to produce a "1" output at end of cycle count. At the next positive transition of the clock all counter stages are reset, and the cycle repeats until stopped by the start switch.

PROGRAMMABLE WAVEFORM GENERATOR

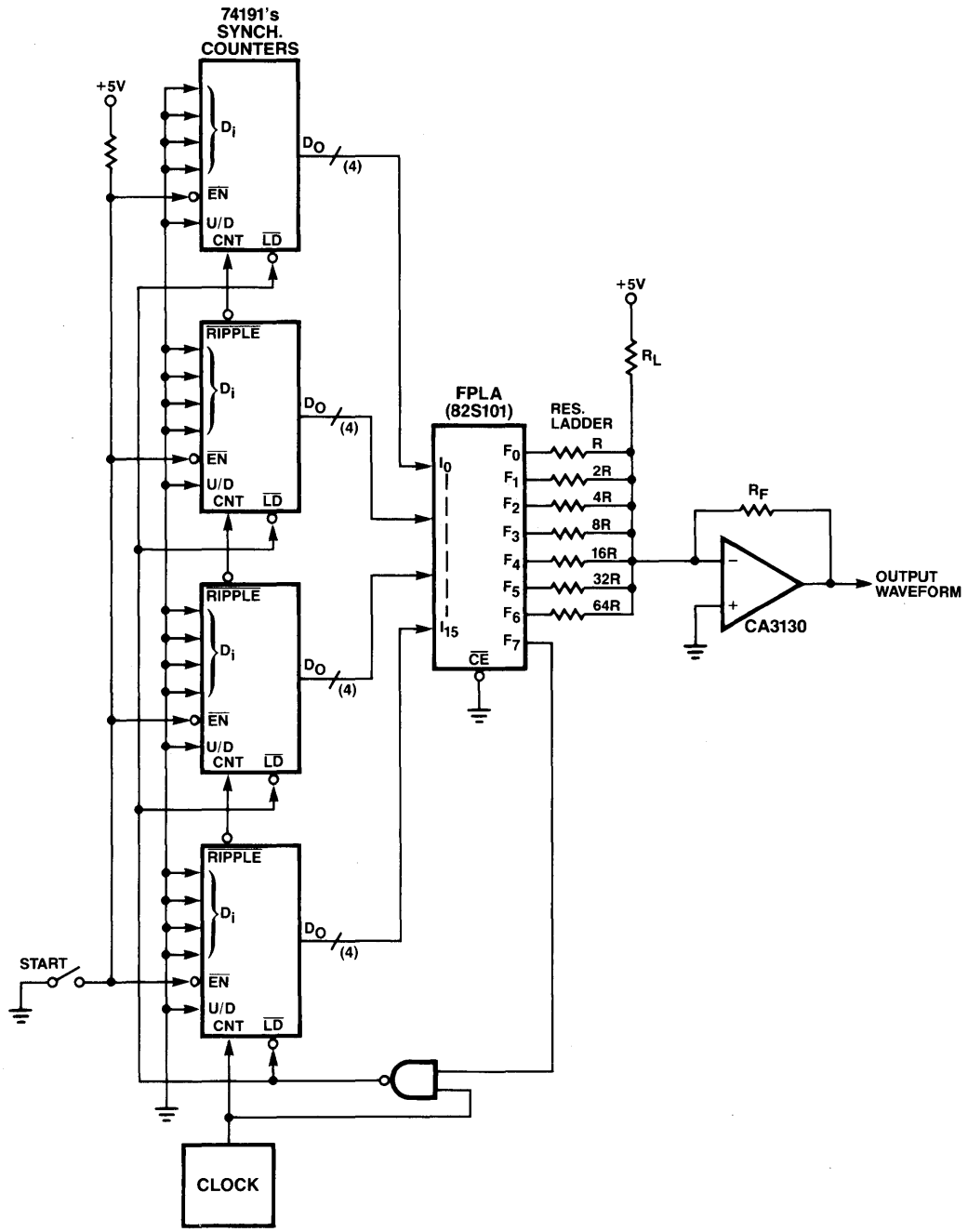


Figure 1.

The circuit shown in Figure 1 uses two FPLA's as the heart of a programmable function generator which can simulate any analog waveform by means of a piecewise linear approximation technique. A 555 timer serves as the clock generator for a 12-bit binary counter consisting of three 74LS163's, which establishes the time base of the generator. The FPLA's continually monitor the state of this counter to determine when the slope of the waveform should be changed and what its new value should be, as well as when to stop the clock, set the counter to zero, and discharge the output integrator. The value of the slope is retained in a pair of 74LS374 octal latches and fed into a 12-bit digital-to-analog converter (DA120IC), whose output is then integrated by an LF356 op-amp to yield the analog output waveform as a continuous series of linear slope segments. Other support circuitry includes a means to encode up to 4 patterns in the same pair of FPLA's, and a provision to generate either one output cycle and stop, or to repeat continuously. It is interesting to note that the same features which make the FPLA's ideally suited for use in generating highly irregular analog waveforms also make them useful for generating some highly-repetitive waveforms, such as high-precision sine waves. Therefore, this circuit can also be very useful in applications requiring either a test generator or a waveform synthesizer.

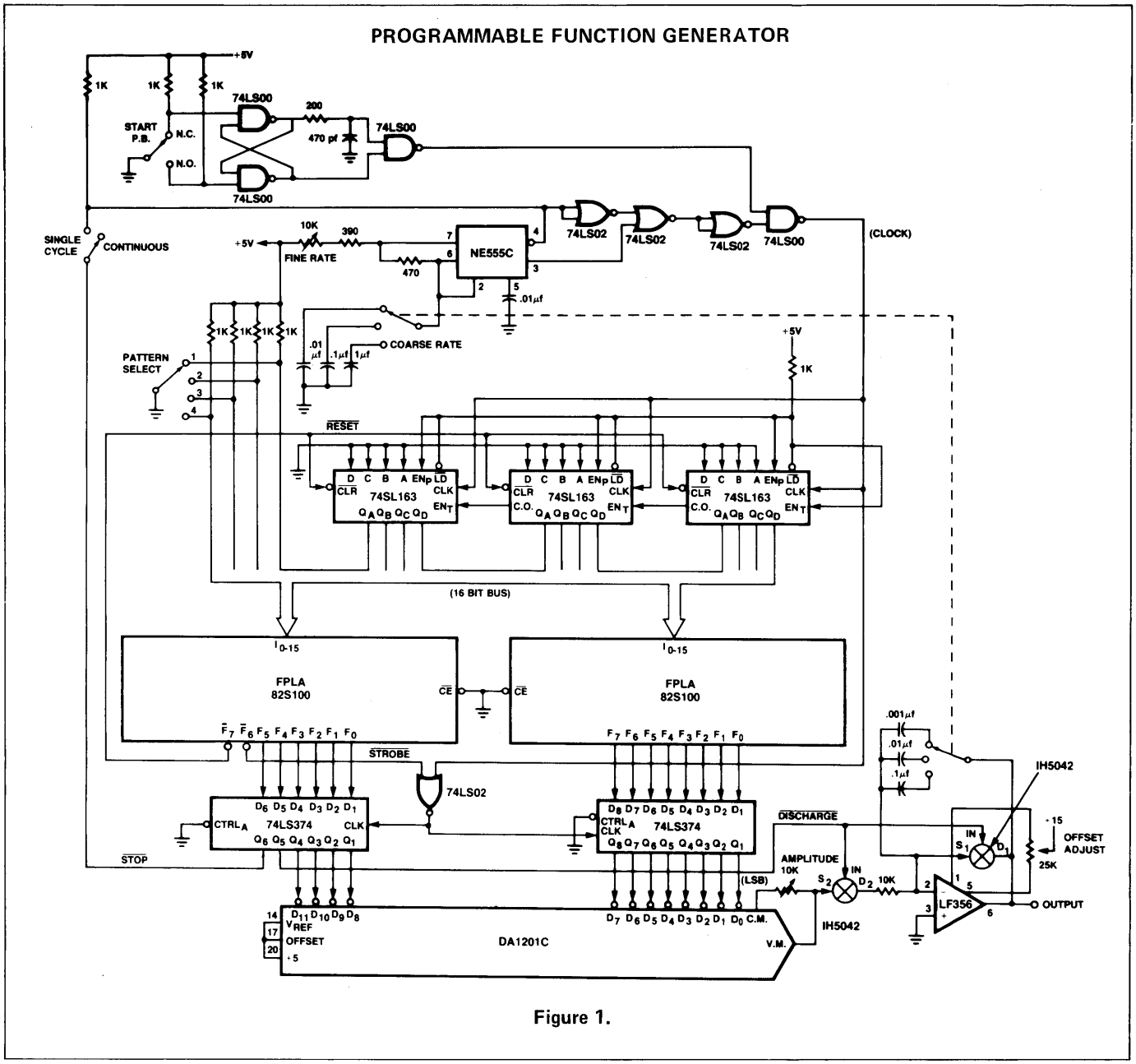


Figure 1.

The block diagram of Figure 1 shows an FPLA used to control the register select ports of 4-bit slice micrprocessors in a microprogrammed processor. Four bits for each port come from the microcode when a fifth bit is 0, or from various bits of the machine language instruction when the fifth bit is 1. Independent control of the A and B ports for 4 instruction formats is obtained with the following assignments.

FPLA INPUTS

$I_0 \rightarrow I_9$ = Pipeline Register (Control ROM) bits

Where:

$I_0 \rightarrow I_3$ = A Port select bits 0 thru 3
I_4 = A Port key bit
$I_5 \rightarrow I_8$ = B Port select bits 0 thru 3
I_9 = B Port Key bit

$I_{10} \rightarrow I_{15}$ = Instruction Register bits

Where:

I_{10} = IR Bit 3 I_{13} = IR Bit 6
I_{11} = IR Bit 4 I_{14} = IR Bit 13
I_{12} = IR Bit 5 I_{15} = IR Bit 14

FPLA OUTPUTS

$I_0 \rightarrow I_3$ = Drive A Port Bits 0 thru 3
$I_4 \rightarrow I_7$ = Drive B Port Bits 0 thru 3

Registers in the microprocessor slice register files are assigned to programmable functions in the machine language being implemented by microprogramming as follows:

MACHINE REGISTER	SLICE REGISTER
A Accumulator	0
B Accumulator	4
Index Reg. 1	1
Index Reg. 2	2
Index Reg. 3	3
Stack Pointer	5
Program Counter	6

Registers are selected by the FPLA when executing machine language instructions in the following formats:

1. **Single Word Memory Reference**
   Bits 13 and 14 call for indexed addresses as:
   00 = No Indexing 10 = Index Reg. 2
   01 = Index Reg. 1 11 = Index Reg. 3
2. **Double Word Memory Reference**
   Bits 4 and 5 call for indexed addressing, and are coded as above.
3. **General Register Operations**
   Bits 4, 5 and 6 indicate the register to be operated upon as:
   000 = Illegal 100 = A Accumulator
   001 = Index Reg. 1 101 = B Accumulator
   010 = Index Reg. 2 110 = Stack Pointer
   011 = Index Reg. 3 111 = Program Counter
4. **Double Register Operations**
   Bits 3 and 4 indicate one operand as:
   00 = A Accumulator 10 = Index Reg. 2
   10 = Index Reg. 1 11 = Index Reg. 3

   Bits 5 and 6 indicate the operand coded as above except:
   00 = B Accumulator.

The FPLA is used to decode and multiplex bits for each register port according to Table 1. This gives rise to the final Program Table for the FPLA shown in Figure 2.

BLOCK DIAGRAM OF REGISTER PORT SELECTOR

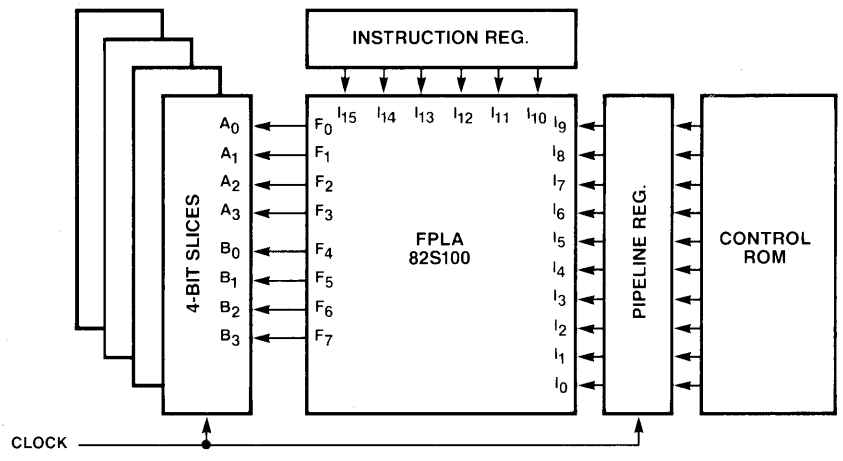


Figure 1.

CROM CODE SELECT ASSIGNMENT

Control ROM Bits 4	3	2	1	0	Register Port Bits 3	2	1	0	Function
0	x	x	x	x	CR3	CR2	CR1	CR0	Direct microprogram control
1	0	0	0	1	0	0	IR14	IR13	Single Word Memory Ref.
1	0	0	1	0	0	0	IR5	IR4	Double Word Memory Ref.
1	0	0	1	1	If IR6 = (0) 0	0	IR5	IR4	General Register Operations
					If code = (101) 0	1	0	0	
					If code = (110) 0	1	0	1	
					If code = (111) 0	1	1	0	
1	0	1	0	0	0	0	IR4	IR3	Two-reg. Operations First Operand
1	0	1	0	1	0	$\overline{(IR6 \cdot IR5)}$	IR6	IR5	Two-reg. Operations Second Operand

Table 1.

FPLA PROGRAM TABLE FOR REGISTER PORT SELECTOR

NO.	PRODUCT TERM INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL H	H	H	H	H	H	H	H OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	L	H	–	–	–								•	•	•	•	A	•	•	•
1	–	–	–	–	–	–	–	–	–	–	–	L	–	H	–	–								•	•	•	•	•	A	•	•
2	–	–	–	–	–	–	–	–	–	–	–	L	–	–	H	–								•	•	•	•	•	•	A	•
3	–	–	–	–	–	–	–	–	–	–	–	L	–	–	–	H								•	•	•	•	•	•	•	A
4	H	–	–	–	–	–	–	–	–	–	–	H	L	L	L	H								•	•	•	•	•	•	A	•
5	–	H	–	–	–	–	–	–	–	–	–	H	L	L	L	H								•	•	•	•	•	•	•	A
6	–	–	–	H	–	–	–	–	–	–	–	H	L	L	H	L								•	•	•	•	•	•	A	•
7	–	–	–	–	H	–	–	–	–	–	–	H	L	L	H	L								•	•	•	•	•	•	•	A
8	–	–	L	H	–	–	–	–	–	–	–	H	L	L	H	H								•	•	•	•	•	•	A	•
9	–	–	L	–	H	–	–	–	–	–	–	H	L	L	H	H								•	•	•	•	•	•	•	A
10	–	–	H	L	H	–	–	–	–	–	–	H	L	L	H	H								•	•	•	•	•	A	•	•
11	–	–	H	H	L	–	–	–	–	–	–	H	L	L	H	H								•	•	•	•	•	A	•	A
12	–	–	H	H	H	–	–	–	–	–	–	H	L	L	H	H								•	•	•	•	•	A	A	•
13	–	–	–	–	H	–	–	–	–	–	–	H	L	H	L	L								•	•	•	•	•	•	A	•
14	–	–	–	–	–	H	–	–	–	–	–	H	L	H	L	L								•	•	•	•	•	•	•	A
15	–	–	L	L	–	–	–	–	–	–	–	H	L	H	L	H								•	•	•	•	•	A	•	•
16	–	–	H	–	–	–	–	–	–	–	–	H	L	H	L	H								•	•	•	•	•	•	A	•
17	–	–	–	H	–	–	–	–	–	–	–	H	L	H	L	H								•	•	•	•	•	•	•	A
18	–	–	–	–	–	–	L	H	–	–	–	–	–	–	–	–								A	•	•	•	•	•	•	•
19	–	–	–	–	–	–	L	–	H	–	–	–	–	–	–	–								•	A	•	•	•	•	•	•
20	–	–	–	–	–	–	L	–	–	H	–	–	–	–	–	–								•	•	A	•	•	•	•	•
21	–	–	–	–	–	–	L	–	–	–	H	–	–	–	–	–								•	•	•	A	•	•	•	•
22	H	–	–	–	–	–	H	L	L	L	H	–	–	–	–	–								•	•	A	•	•	•	•	•
23	–	H	–	–	–	–	H	L	L	L	H	–	–	–	–	–								•	•	•	A	•	•	•	•
24	–	–	–	H	–	–	H	L	L	H	L	–	–	–	–	–								•	•	A	•	•	•	•	•
25	–	–	–	–	H	–	H	L	L	H	L	–	–	–	–	–								•	•	•	A	•	•	•	•
26	–	–	L	H	–	–	H	L	L	H	H	–	–	–	–	–								•	•	A	•	•	•	•	•
27	–	–	L	–	H	–	H	L	L	H	H	–	–	–	–	–								•	•	•	A	•	•	•	•
28	–	–	H	L	H	–	H	L	L	H	H	–	–	–	–	–								•	A	•	•	•	•	•	•
29	–	–	H	H	L	–	H	L	L	H	H	–	–	–	–	–								•	A	•	A	•	•	•	•
30	–	–	H	H	H	–	H	L	L	H	H	–	–	–	–	–								•	A	A	•	•	•	•	•
31	–	–	–	–	H	–	H	L	H	L	L	–	–	–	–	–								•	•	A	•	•	•	•	•
32	–	–	–	–	–	H	H	L	H	L	L	–	–	–	–	–								•	•	•	A	•	•	•	•
33	–	–	L	L	–	–	H	L	H	L	H	–	–	–	–	–								•	A	•	•	•	•	•	•
34	–	–	H	–	–	–	H	L	H	L	H	–	–	–	–	–								•	•	A	•	•	•	•	•
35	–	–	–	H	–	–	H	L	H	L	H	–	–	–	–	–								•	•	•	A	•	•	•	•

Figure 2.

FPLAs bring economy and flexibility to the design of the control unit in microprogrammed computers, by allowing complete freedom in allocating subroutines in microstore, and enhancing the use of variable formats in the op-code field from the Instruction Register.

The block diagram in Figure 1 shows a general MPCU organization. A key element in this structure is the op-code decoder for cracking each op-code from the IR into sub-routine-start addresses in control memory, and address mode definition.

To implement these functions for the MPCU in the PDP-11 minicomputer, three FPLAs are all that is required, as shown in the circuit in Figure 2. The first FPLA is used as Address Mode Interpreter, while FPLA-2 and FPLA-3 function as Instruction Interpreter. Translator function (op-code, source address mode, or destination address mode) is selected by control inputs A and B from memory, also defined in Figure

TYPICAL STRUCTURE OF MICROPROGRAMMED CONTROL UNIT

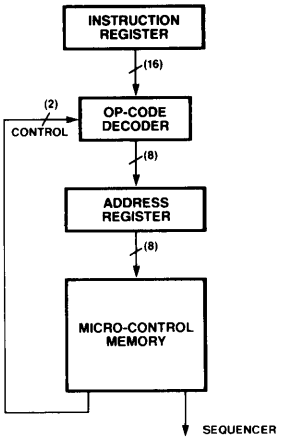
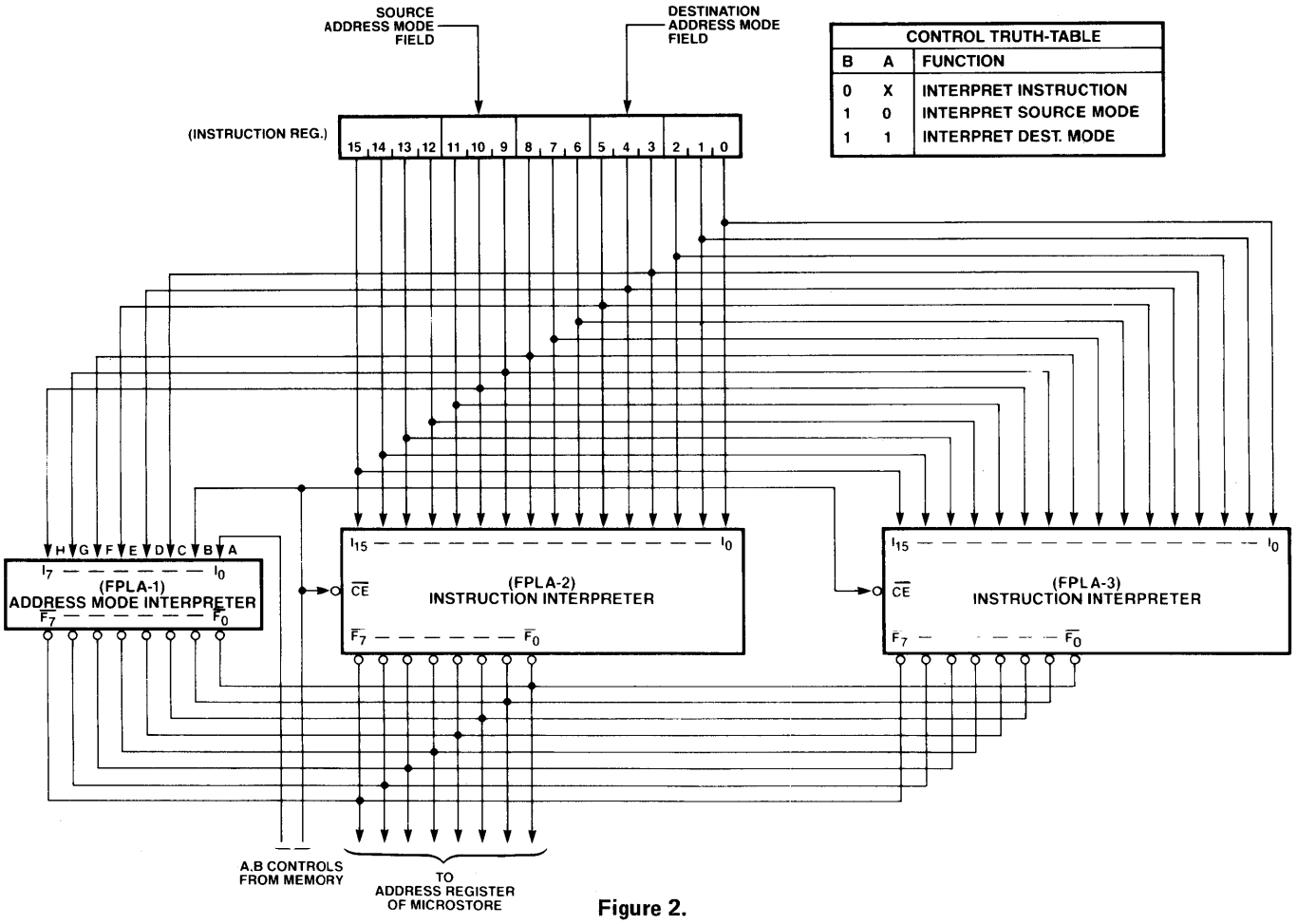


Figure 1.

OP-CODE DECODER NETWORK
ALL FPLAs ARE OPEN COLLECTOR (82S101). PULL-UP RESISTORS NOT SHOWN



CONTROL TRUTH-TABLE		
B	A	FUNCTION
0	X	INTERPRET INSTRUCTION
1	0	INTERPRET SOURCE MODE
1	1	INTERPRET DEST. MODE

Figure 2.

2. PDP-11 instructions have variable formats, from 4 to 16 bits. Their binary codes are tabulated in the input field of the Program Tables in Figures 3 and 4 for the Instruction Interpreter FPLAs. The output field of both FPLAs is programmed with a binary number corresponding to an arbitrary starting address in control memory. Bits 3 to 5 and 9 to 11 of the IR contain address mode information which is decoded by FPLA-1 as shown in the input field of the Program Table in Figure 5. The outputs of this FPLA are also assigned an arbitrary code pointing to a starting address in control memory. Note that all undefined instruction codes are not programmed, and will appear as an all 1's address from the interpreter. This leaves 13 spare product terms in the Instruction Interpreter FPLAs to be used for editing, or future expansion.

A ROM version of this circuit would require 65K x 8 for the Instruction Interpreter and 128 x 8 for the Addressing Mode Interpreter. With 4K x 8 ROMs, this would require a total of 17 packages.

PROGRAM TABLE FOR INSTRUCTION INTERPRETER FPLA-2

INSTRUCTION	PRODUCT TERM NO.	INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HALT	0	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
WAIT	1	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H
RETURN FROM INTERRUPT	2	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L
BREAKPOINT TRAP	3	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H
INPUT/OUTPUT TRAP	4	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	L
RESET	5	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	H
RETURN, INHIBIT TRACE TRAP	6	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H	L
JUMP	7	L	L	L	L	L	L	L	L	L	H	–	–	–	–	–	–
RETURN FROM SUBROUTINE	8	L	L	L	L	L	L	L	L	H	L	L	L	L	–	–	–
RESERVED INSTRUCTION (8 CODES)	9	L	L	L	L	L	L	L	L	H	L	L	L	H	–	–	–
RESERVED INSTRUCTION (8 CODES)	10	L	L	L	L	L	L	L	L	H	L	L	H	L	–	–	–
NO OPERATION	11	L	L	L	L	L	L	L	L	H	L	H	L	L	L	L	L
CHANGE CODITION CODE (16 CODES)	12	L	L	L	L	L	L	L	L	H	L	H	–	H	–	–	–
CHANGE CONDITION CODE (16 CODES)	13	L	L	L	L	L	L	L	L	H	L	H	H	–	–	–	–
SWAP BYTES	14	L	L	L	L	L	L	L	L	H	H	–	–	–	–	–	–
UNCONDITIONAL BRANCH	15	L	L	L	L	L	L	L	H	–	–	–	–	–	–	–	–
BRANCH NOT EQUAL	16	L	L	L	L	L	L	H	L	–	–	–	–	–	–	–	–
BRANCH EQUAL	17	L	L	L	L	L	L	H	H	–	–	–	–	–	–	–	–
BRANCH GREATER OR EQUAL	18	L	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–
BRANCH LESS THAN	19	L	L	L	L	L	H	L	H	–	–	–	–	–	–	–	–
BRANCH GREATER THAN	20	L	L	L	L	L	H	H	L	–	–	–	–	–	–	–	–
BRANCH LESS OR EQUAL	21	L	L	L	L	L	H	H	H	–	–	–	–	–	–	–	–
JUMP TO SUBROUTINE	22	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–	–
CLEAR	23	L	L	L	L	H	L	H	L	L	L	–	–	–	–	–	–
COMPLEMENT	24	L	L	L	L	H	L	H	L	L	H	–	–	–	–	–	–
INCREMENT	25	L	L	L	L	H	L	H	L	H	L	–	–	–	–	–	–
DECREMENT	26	L	L	L	L	H	L	H	L	H	H	–	–	–	–	–	–
NEGATE	27	L	L	L	L	H	L	H	H	L	L	–	–	–	–	–	–
ADD CARRY	28	L	L	L	L	H	L	H	H	L	H	–	–	–	–	–	–
SUBTRACT CARRY	29	L	L	L	L	H	L	H	H	H	L	–	–	–	–	–	–
TEST	30	L	L	L	L	H	L	H	H	H	H	–	–	–	–	–	–
ROTATE RIGHT	31	L	L	L	L	H	H	L	L	L	L	–	–	–	–	–	–
ROTATE LEFT	32	L	L	L	L	H	H	L	L	L	H	–	–	–	–	–	–
ARITHMETIC SHIFT RIGHT	33	L	L	L	L	H	H	L	L	H	L	–	–	–	–	–	–
ARITHMETIC SHIFT LEFT	34	L	L	L	L	H	H	L	L	H	H	–	–	–	–	–	–
MARK	35	L	L	L	L	H	H	L	H	L	L	–	–	–	–	–	–
SIGN EXTEND	36	L	L	L	L	H	H	L	H	H	H	–	–	–	–	–	–
MOVE	37	L	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–
COMPARE	38	L	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT TEST	39	L	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–
BIT CLEAR	40	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT SET	41	L	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–
ADD	42	L	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–
MULTIPLY	43	L	H	H	H	L	L	L	–	–	–	–	–	–	–	–	–
DIVIDE	44	L	H	H	H	L	L	H	–	–	–	–	–	–	–	–	–
SHIFT ARITHMETIC	45	L	H	H	H	L	H	L	–	–	–	–	–	–	–	–	–
SHIFT ARITHMETIC COMBINED	46	L	H	H	H	L	H	H	–	–	–	–	–	–	–	–	–
EXCLUSIVE OR	47	L	H	H	H	H	L	L	–	–	–	–	–	–	–	–	–

ACTIVE LEVEL L L L L L L L L

OUTPUT FUNCTION 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 3.

PROGRAM TABLE FOR INSTRUCTION INTERPRETER FPLA-3

INSTRUCTION	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLOATING ADD	0	L	H	H	H	H	L	H	L	L	L	L	L	L	–	–	–
FLOATING SUBTRACT	1	L	H	H	H	H	L	H	L	L	L	L	L	H	–	–	–
FLOATING MULTIPLY	2	L	H	H	H	H	L	H	L	L	L	L	H	L	–	–	–
FLOATING DIVIDE	3	L	H	H	H	H	L	H	L	L	L	L	H	H	–	–	–
SUBTRACT 1 & BRANCH	4	L	H	H	H	H	H	H	–	–	–	–	–	–	–	–	–
BRANCH PLUS	5	H	L	L	L	L	L	L	L	–	–	–	–	–	–	–	–
BRANCH MINUS	6	H	L	L	L	L	L	L	H	–	–	–	–	–	–	–	–
BRANCH HIGHER	7	H	L	L	L	L	L	H	L	–	–	–	–	–	–	–	–
BRANCH LOWER OR SAME	8	H	L	L	L	L	L	H	H	–	–	–	–	–	–	–	–
BRANCH OVERFLOW CLEAR	9	H	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–
BRANCH OVERFLOW SET	10	H	L	L	L	L	H	L	H	–	–	–	–	–	–	–	–
BRANCH CARRY CLEAR	11	H	L	L	L	L	H	H	L	–	–	–	–	–	–	–	–
BRANCH CARRY SET	12	H	L	L	L	L	H	H	H	–	–	–	–	–	–	–	–
EMULATOR TRAP	13	H	L	L	L	H	L	L	L	–	–	–	–	–	–	–	–
TRAP	14	H	L	L	L	H	L	L	H	–	–	–	–	–	–	–	–
CLEAR BYTE	15	H	L	L	L	H	L	H	L	L	L	–	–	–	–	–	–
COMPLEMENT BYTE	16	H	L	L	L	H	L	H	L	L	H	–	–	–	–	–	–
INCREMENT BYTE	17	H	L	L	L	H	L	H	L	H	L	–	–	–	–	–	–
DECREMENT BYTE	18	H	L	L	L	H	L	H	L	H	H	–	–	–	–	–	–
NEGATE BYTE	19	H	L	L	L	H	L	H	H	L	L	–	–	–	–	–	–
ADD CARRY TO BYTE	20	H	L	L	L	H	L	H	H	L	H	–	–	–	–	–	–
SUBTRACT CARRY FROM BYTE	21	H	L	L	L	H	L	H	H	H	L	–	–	–	–	–	–
TEST BYTE	22	H	L	L	L	H	L	H	H	H	H	–	–	–	–	–	–
ROTATE RIGHT BYTE	23	H	L	L	L	H	H	L	L	L	L	–	–	–	–	–	–
ROTATE LEFT BYTE	24	H	L	L	L	H	H	L	L	L	H	–	–	–	–	–	–
ARITH SHIFT RIGHT BYTE	25	H	L	L	L	H	H	L	L	H	L	–	–	–	–	–	–
ARITH SHIFT LEFT BYTE	26	H	L	L	L	H	H	L	L	H	H	–	–	–	–	–	–
MOVE TO PGM STATUS	27	H	L	L	L	H	H	L	H	L	L	–	–	–	–	–	–
MOVE FROM PGM STATUS	28	H	L	L	L	H	H	L	H	H	H	–	–	–	–	–	–
MOVE BYTE	29	H	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–
COMPARE BYTE	30	H	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT TEST BYTE	31	H	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–
BIT CLEAR BYTE	32	H	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT SET BYTE	33	H	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–
SUBTRACT	34	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–

ACTIVE LEVEL: L L L L L L L L

OUTPUT FUNCTION: 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 4.

PROGRAM TABLE FOR ADDRESS MODE INTERPRETER FPLA-1 WHEN B = "0" ALL DEVICE OUTPUTS ARE FORCED HIGH, DELEGATING BUS CONTROL TO THE INSTRUCTION INTERPRETER FPLAs

INSTRUCTION	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	H 7	G 6	F 5	E 4	D 3	C 2	B 1	A 0
DESELECT ADDR. MODE INTERP.	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–
SOURCE, REGISTER MODE	1	–	–	–	–	–	–	–	–	L	L	L	–	–	–	H	–
SOURCE, REGISTER DEFERRED	2	–	–	–	–	–	–	–	–	L	L	H	–	–	–	H	L
SOURCE, AUTO INCREMENT	3	–	–	–	–	–	–	–	–	L	H	L	–	–	–	H	L
SOURCE, AUTO INCR DEFERRED	4	–	–	–	–	–	–	–	–	L	H	H	–	–	–	H	L
SOURCE, AUTO DECREMENT	5	–	–	–	–	–	–	–	–	H	L	L	–	–	–	H	L
SOURCE, AUTO DECR DEFERRED	6	–	–	–	–	–	–	–	–	H	L	H	–	–	–	H	L
SOURCE, INDEX	7	–	–	–	–	–	–	–	–	H	H	L	–	–	–	H	L
SOURCE, INDEX DEFERRED	8	–	–	–	–	–	–	–	–	H	H	H	–	–	–	H	L
DESTINATION, REGISTER MODE	9	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	H
DESTINATION, REGISTER DEFERRED	10	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	H
DESTINATION, AUTO INCREMENT	11	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	H
DESTINATION, AUTO INCR DEFERRED	12	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H
DESTINATION, AUTO DECREMENT	13	–	–	–	–	–	–	–	–	–	–	–	H	L	L	H	H
DESTINATION, AUTO DECR DEFERRED	14	–	–	–	–	–	–	–	–	–	–	–	H	L	H	H	H
DESTINATION, INDEX	15	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H	H
DESTINATION, INDEX DEFERRED	16	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	H

UNUSED

ACTIVE LEVEL: L L L L L L L L

OUTPUT FUNCTION: 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 5.

One of the most difficult tasks of a general purpose microprogrammed emulator shown in Figure 1 is the decoding of the target instruction. The circuit shown in Figure 2 provides a fairly general instruction decoder whose decoding can be changed to decode a variety of instruction sets by programming an FPLA. The equivalent circuit using

BLOCK DIAGRAM OF MICROPROGRAMMED EMULATOR

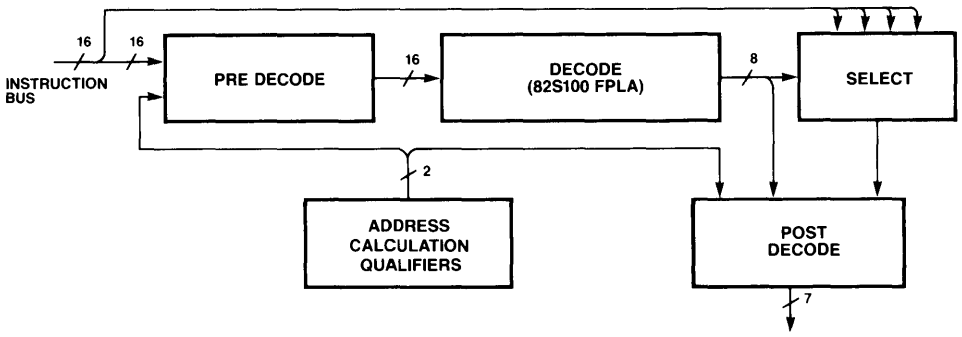


Figure 1.

PROGRAMMABLE INSTRUCTION DECODER NETWORK

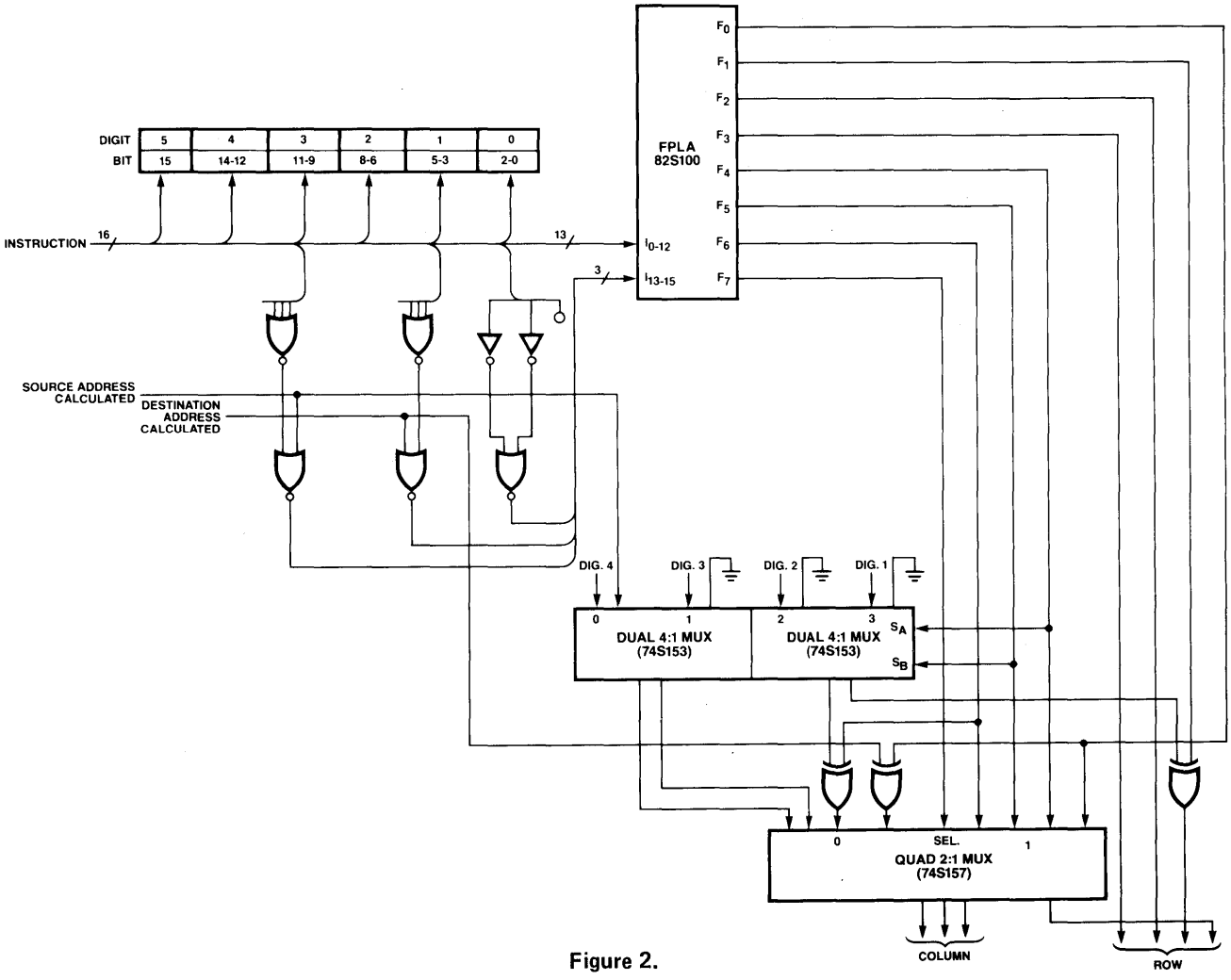


Figure 2.

PROMs would take over one hundred IC's.

The circuit has as input a sixteen bit instruction and two bits that tell whether address calculation has taken place. This is so that when the instruction is decoded and no memory reference is required, the decoder will generate the microinstruction address of the appropriate macroinstruction routine. If the instruction requires a memory reference, the address of the address calculation routine is generated, in which the address calculation bit is set. The microinstruction causes the decoder to be used again, but the second time through, the presence of the address calculation bit causes the decoder to generate the address of the appropriate microroutine, instead of the address calculation routine.

The FPLA in the circuit has been programmed to decode the DEC PDP-11 series of instructions. These are tabulated in the FPLA Program Table of Figure 3. Given a 16 bit opcode from a PDP-11, the circuit will return a microcode starting address. In addition, if a memory reference is required, the circuit will generate the address of the memory reference routine. The microcode can then set a status bit, so that during the second pass thru the network the memory reference routine is bypassed.

FPLA PROGRAM TABLE FOR PDP-11 DECODE

	PRODUCT TERM																ACTIVE LEVEL								
	INPUT VARIABLE																H	H	H	H	H	H	H	H	
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENTS
0	–	L	H	–	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	CALCULATE SOURCE ADDRESS, WORD
1	–	–	L	H	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	
2	–	H	–	L	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	
3	H	L	H	–	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	CALCULATE SOURCE ADDRESS, BYTE
4	H	L	H	–	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
5	H	H	L	–	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
6	H	H	L	–	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
7	H	–	L	H	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
8	H	–	L	H	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
9	–	L	H	–	–	–	–	–	–	–	–	–	–	L	H	–	•	•	A	•	•	A	•	A	CALCULATE DESTINATION ADDRESS, WORD
10	–	–	L	H	–	–	–	–	–	–	–	–	–	L	H	–	•	•	A	•	•	A	•	•	
11	–	H	–	L	–	–	–	–	–	–	–	–	–	L	H	–	•	•	A	•	•	A	•	•	
12	L	L	L	L	L	L	L	L	–	H	–	–	–	–	H	–	•	•	A	•	•	A	•	•	
13	–	L	L	L	H	–	–	–	–	–	–	–	–	–	H	–	•	•	A	•	•	A	•	•	
14	L	H	H	H	L	–	–	–	–	–	–	–	–	–	H	–	•	•	A	•	•	A	•	•	
15	L	H	H	H	H	L	L	–	–	–	–	–	–	–	H	–	•	•	A	•	•	A	•	•	
16	H	L	H	–	–	–	–	–	–	–	–	–	–	L	H	H	•	•	A	•	•	A	•	•	CALCULATE DESTINATION ADDRESS, BYTE
17	H	H	L	–	–	–	–	–	–	–	–	–	–	L	H	H	•	•	A	•	•	A	•	A	
18	H	–	L	H	–	–	–	–	–	–	–	–	–	L	H	H	•	•	A	•	•	A	•	A	
19	H	L	L	L	H	–	–	–	–	–	–	–	–	–	H	H	•	•	A	•	•	A	•	A	
20	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	–	A	•	•	•	A	A	A	A	INTERRUPT INSTR.
21	L	L	L	L	L	L	L	L	–	–	–	–	–	–	L	–	•	A	A	A	A	A	A	•	SPEC/JMP/RTS/SWAP
22	L	L	L	L	L	L	L	L	H	L	L	L	H	–	–	–	A	A	A	A	A	A	A	A	RESERVED
23	L	L	L	L	L	L	L	L	H	L	L	H	L	–	–	–	A	A	A	A	A	A	A	A	
24	L	L	L	L	L	L	L	L	H	L	L	H	H	–	–	–	A	A	•	•	A	A	A	A	SPL
25	L	L	L	L	L	L	L	L	H	L	H	L	–	–	–	–	A	•	•	•	•	A	•	•	COND. CODE CLR
26	L	L	L	L	L	L	L	L	–	L	H	H	–	–	–	–	A	•	•	•	•	A	•	A	COND. CODE SET
27	L	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•	BRANCH, TRUE COND.
28	L	L	L	L	H	L	L	–	–	–	–	–	–	–	L	–	•	•	•	•	•	•	A	•	JSR
29	–	L	L	L	H	L	H	–	–	–	–	–	–	L	L	–	•	•	A	A	A	•	A	•	SINGLE OP., #1
30	–	L	L	L	H	H	L	–	–	–	–	–	–	L	L	–	•	•	A	A	A	•	•	•	SINGLE OP., #2
31	–	L	L	L	H	H	H	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	A	RESERVED
32	L	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	•	A	A	A	A	•	•	EIS
33	L	H	H	H	H	L	H	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•	RESERVED
34	L	H	H	H	H	L	H	L	L	L	L	–	–	–	–	–	A	A	•	•	A	A	A	A	FLOATING POINT
35	L	H	H	H	H	H	L	–	–	–	–	–	–	–	–	–	A	A	•	A	A	A	A	A	RESERVED
36	L	H	H	H	H	H	H	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	A	SOB
37	H	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	•	BRANCH, FALSE COND.
38	H	L	L	L	H	L	L	L	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	•	EMT
39	H	L	L	L	H	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	A	TRP
40	H	L	L	L	H	H	L	H	L	L	–	–	–	L	L	–	•	A	A	A	A	•	•	•	MTPS
41	–	L	L	L	H	H	L	H	L	H	–	–	–	L	L	–	A	A	•	•	A	•	•	•	MT/F • I/D
42	–	L	L	L	H	H	L	H	H	L	–	–	–	L	L	–	A	A	A	•	A	•	•	•	
43	H	L	L	L	H	H	L	H	H	H	–	–	–	L	L	–	•	A	A	A	A	•	•	•	MKRS
44	H	H	H	L	–	–	–	–	–	–	–	–	–	L	L	–	•	A	•	•	•	•	A	•	SUBTRACT
45	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	A	A	A	•	A	A	A	A	EXT. FLOAT. PNT
46	L	L	H	L	H	H	L	H	L	L	–	–	–	–	–	–	A	A	•	A	A	A	A	•	MARK

Figure 3.

SINGLE OR BLOCK ADDRESS DECODING FOR M6800 μP 30

The 82S100 is ideally suited for decoding a 16 bit Address Bus into as many as 48 locations or blocks of locations. As shown in Figure 1, only 4¼ packages are required to generate 48 active-LOW chip enable or device select outputs.

FPLA outputs F4, F5, and F6 are programmed active-LOW. When an assigned address is on the bus, these outputs select one of the three 74154 Decoders, while outputs F0 through F3 functions as a 0-to-15 binary counter to index the decoder according to the programmed address. assignments. The "don't care" input programming feature allows block sizes of integral powers of 2 (as in memory chips) to be as easily assigned as singly decoded addresses.

The proper memory clock timing for an M6800 system is provided to the 74154 decoders by the $\overline{\phi 2 \cdot VMA}$ output of the 7400 Nand Gate.

This circuit would be most useful in a small microprocessor system having a lot of miscellaneous devices to be addressed.

ADDRESS DECODING IN MC6800 μP BASED SYSTEM

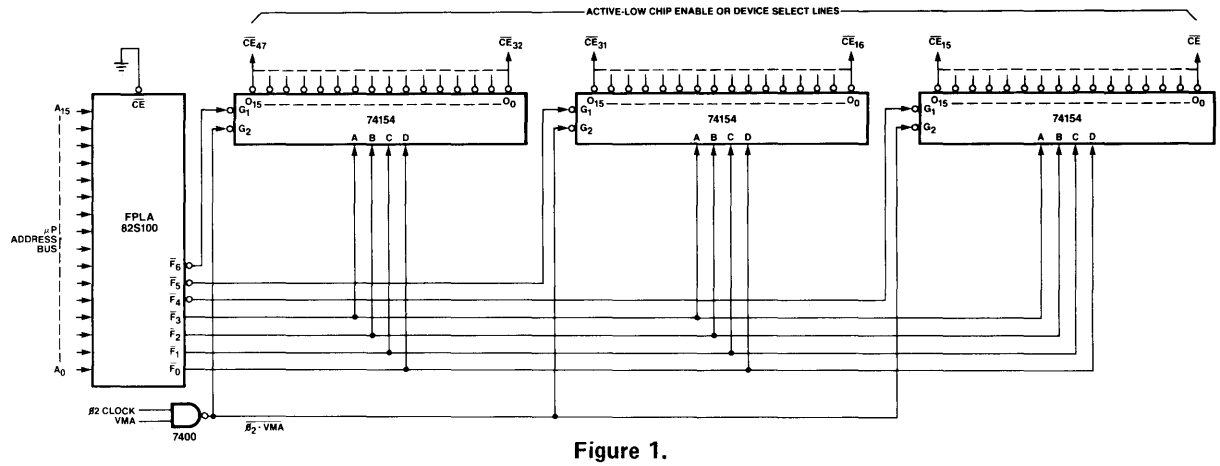


Figure 1.

INTERRUPT PRIORITY CONTROL FOR M6800 μP 31

The circuit in Figure 1 provides interrupt prioritization for a maximum of eleven devices, thereby eliminating the need for software polling techniques. The FPLA combines the functions of an eleven input "OR" gate, a quad Multiplexer, an eleven input Priority Encoder, and a 13x4 ROM. The circuitry adds a maximum of 125 ns to the settling time of the address bus.

Without prioritizing, all interrupt request inputs would be OR'ed together and applied to the $\overline{IRQ}$ input (pin 4) of the MPU. Upon receiving an interrupt request the microprocessor successively addresses FFF8 and FFF9. The 16 bits of data at these two addresses comprise the starting address for the interrupt routine. The MPU must then poll each device to determine who initiated the request and, in the case of multiple requests, must further determine which device will be serviced first. With the priority circuity shown, all of the inputs are brought to the FPLA. Here they are OR'ed together at F_4 to produce the $\overline{IRQ}$ signal and where they are available for controlling F_0 through F_3 during the interrupt sequence. The other inputs are A_1 through A_4 from the MPU, and the output of the interrupt vector decode logic, for recognizing when the microprocessor outputs addresses FFF8 and FFF9. The function of the circuit is summarized in the truth-table of Figure 2. When there is no interrupt request the address bus is the same as the address outputted by the MPU. When any of the inputs goes LOW, F_4 goes LOW and the FPLA is ready to modify bits A_1 thru A_4 of the address bus with a specific starting address. These have been assigned to the device which initiated the interrupt in accordance with the vector table of Figure 3. This substitution occurs when I_4 goes HIGH. With proper programming the FPLA will automatically indicate the correct starting address when multiple requests are present. For example, if I_5 is to have the highest priority, the product matrix of I_6 through I_{15} contain I_5 as inhibit function. If I_6 is the next highest priority, all the product matrixes of I_7 through I_{15} would also contain I_6 as an inhibit function. The sum matrix is programmed according to the starting address assigned to the interrupt to be serviced. For example, if the device connected to I_7 is to have a starting address at FFE4 and FFF5, the product term for I_7 would be connected in the sum matrix to produce a (0010) at F0, F1, F2, and F3 respectively. The logic equation set to be programmed in the FPLA is tabulated in

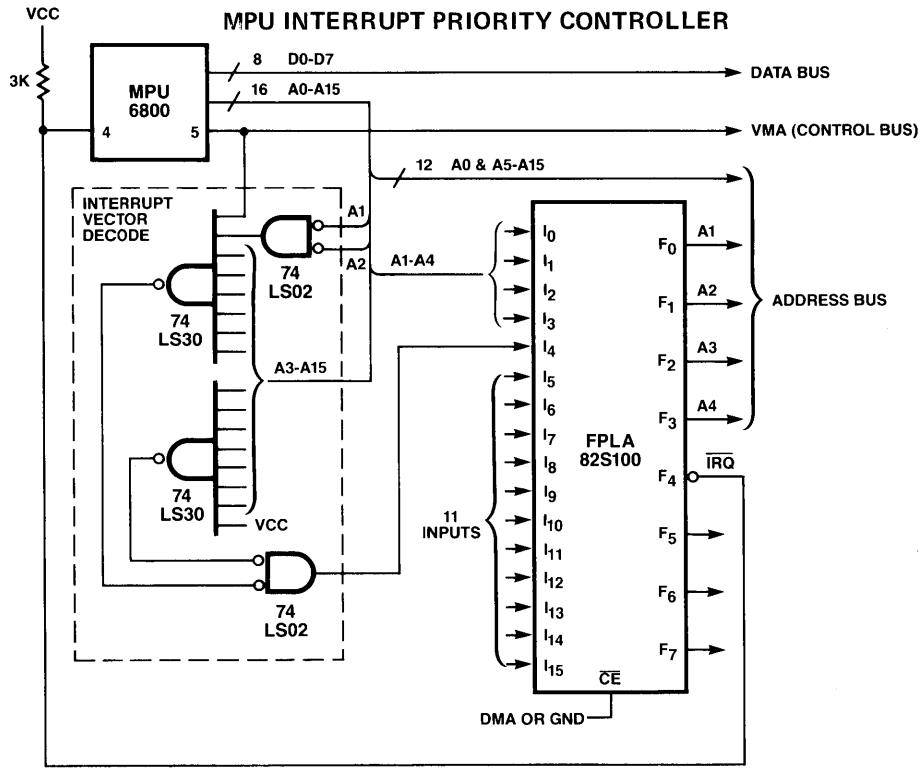


Figure 1.

Figure 4. The range of interrupt addresses is FFE0 through FFF9. The data at these locations can be either hardwired in, bootstrapped in during initialization, or firmware resident.

There are two extra address pairs and three spare outputs. These give the circuit a few additional capabilities not normally found with conventional priority circuits. The extra addresses can be used to designate special routines for certain combinations of multiple inputs. The extra outputs can be programmed to provide supplementary information. These can signal the operator, or MPU, of input combinations that are illegal or require immediate servicing. Outputs F5, F6, and F7 could be used to illuminate LEDs and/or be applied to microprocessor inputs such as $\overline{NMI}$, $\overline{HALT}$, or $\overline{RESET}$. If there is a need to place the address bus in the Hi-Z state, $\overline{CE}$ of the FPLA would be a function of, for example, a DMA controller.

The circuit is designed for use with the M6800 MPU family of components, but can be easily applied to other microprocessors.

CIRCUIT TRUTH-TABLE

I_5-I_{15}	I_4	$\overline{CE}$	$\overline{F4}$	ADD. BUS
X	X	H	HI-Z	HI-Z
H	L	L	H	A0 - A15
ANY INPUT(S) LOW	L	L	L	A0 - A15
	H	L	L	ADDRESS VECTOR FFE0 → FFF9

Figure 2.

VECTOR ASSIGNMENT

	ADDRESS BUS							
HEX	A15 – – – – – – –			A4	A3	A2	A1	A0
FFE0	1111	1111	111	0	0	0	0	0
FFE1	1111	1111	111	0	0	0	0	1
•	•	•	•			FPLA		•
•	•	•	•			OUTPUTS		•
•	•	•	•					•
FFF9	1111	1111	111	1	1	0	0	1

Figure 3.

LOGIC EQUATION SET OF FPLA

$F_0 = \overline{I_4} \cdot I_0 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_5)
$F_1 = \overline{I_4} \cdot I_1 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_2 = \overline{I_4} \cdot I_2 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_3 = \overline{I_4} \cdot I_3 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_4 = \overline{I_5} + \overline{I_6} + \ldots . + \overline{I_{15}}$
$\overline{CE}$ = INPUT FROM DMA CONTROLLER OR GROUND

Figure 4.

UNIBUS ADDRESS MONITOR 32

The circuit in Figure 1 monitors how often a specific address or a specific group of addresses is referenced on the PDP-11 unibus. Up to eight groups can be monitored at the same time. The FPLA outputs are programmed to recognize a specific input address or group of addresses. When a particular address is referenced, the corresponding output fires a pulse to the corresponding counter and increments it. A switch selectable LED display and a frequency counter measurement test point is also provided. The circuit is very useful in measuring the effective data rate of peripherals.

UNIBUS ADDRESS MONITOR

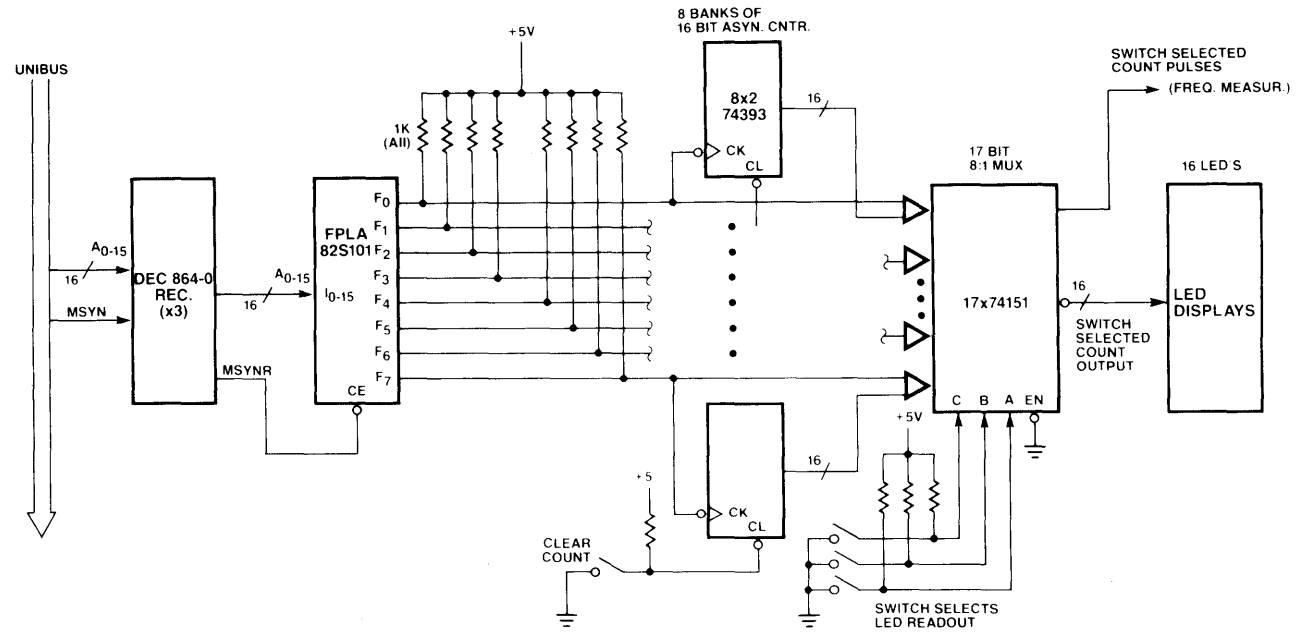


Figure 1.

HARDWARE-MACRO GENERATOR 33

The circuit shown in Figure 1 can be used for implementing hardware MACROS, or for calling conditional Diagnostics in a general processing system.

If $\overline{CE}$, EN1 and EN2 are set, and a 'keyword' instruction is fetched from memory, the FPLA is actuated to sequentially place new instructions in the input instruction stream. If possible the program counter is inhibited for subsequent instruction fetches, otherwise the last MACRO code resets the program counter. The next FPLA instruction is determined by decoding the last FPLA code (via 74174) until a 1's condition is set, which returns control to the memory bus. EN1 and EN2 allow conditional selection of different 'keyword' sets.

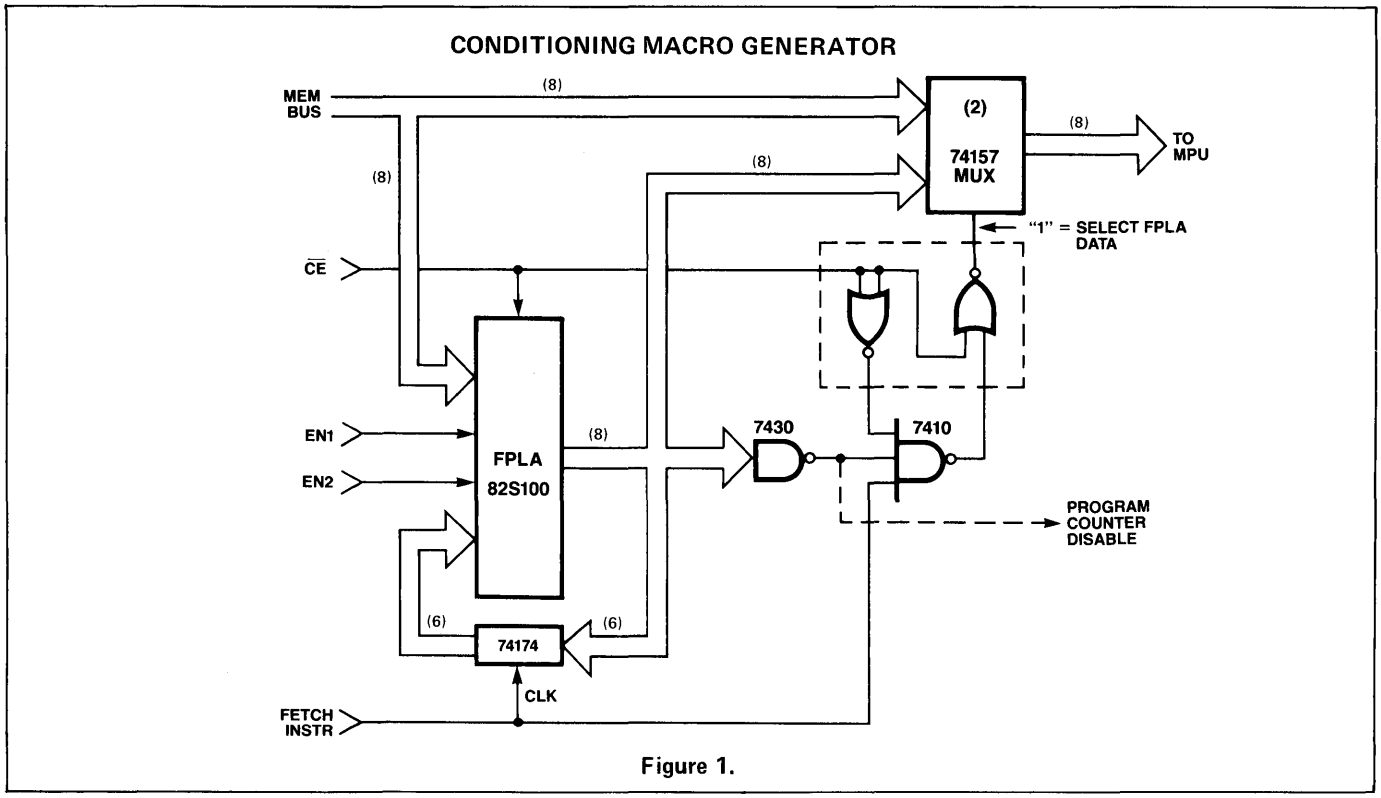


Figure 1.

FPLA ENHANCES PIPELINED PROCESS ARCHITECTURE 34

In a microcoded 16 bit processor the instruction fetch/ decode/execute cycles can be overlapped in a pipelined fashion to increase execution speed. As shown in the block diagram of Figure 1, while instruction A is executing in firmware, instruction B is decoded by the decode FPLA to generate the next starting firmware address, and instruction C is fetched from memory into the pipeline register.

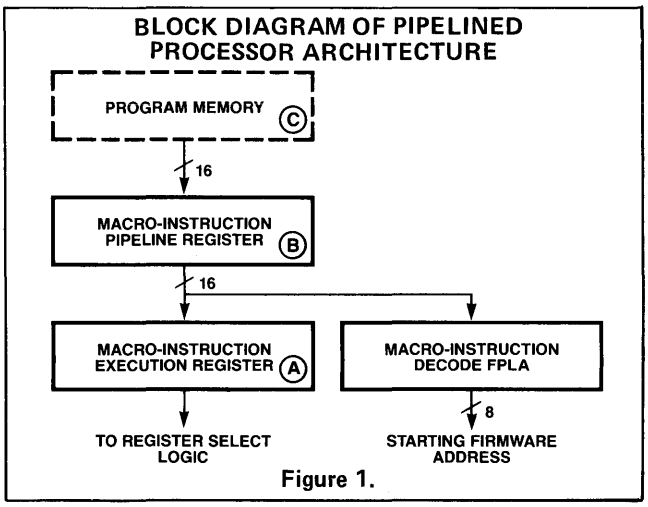


Figure 1.

The circuit in Figure 1 shows an FPLA as an 8-bit I/O port decoder for generating chip select signals to 32 PIA modules, type 6820, used in microprocessor designs. These could also be ACIA 6850, or other similar interface adapters.

Since the I/O address for 2 ports is fully decoded by one FPLA product term, the circuit can select 64 ports and a maximum of 96 when using all 48 product terms in one FPLA. For 16 or less ports, a single 1 of 16 decoder is sufficient.

Besides vastly reducing the number of decoding circuits, the FPLA avoids hardware limitations of software I/O addresses because it permits selection of more than one I/O port at a time, or one I/O port at more than one software address, or both. Also, since it allows to program some inputs as true "Don't Cares", a port can be assigned to be at an entire page or more of addresses.

Note that there remain two spare inputs to the FPLA. These can be used for hardware I/O control, such as automatic control for out of paper condition on printer.

I/O PORT DECODER

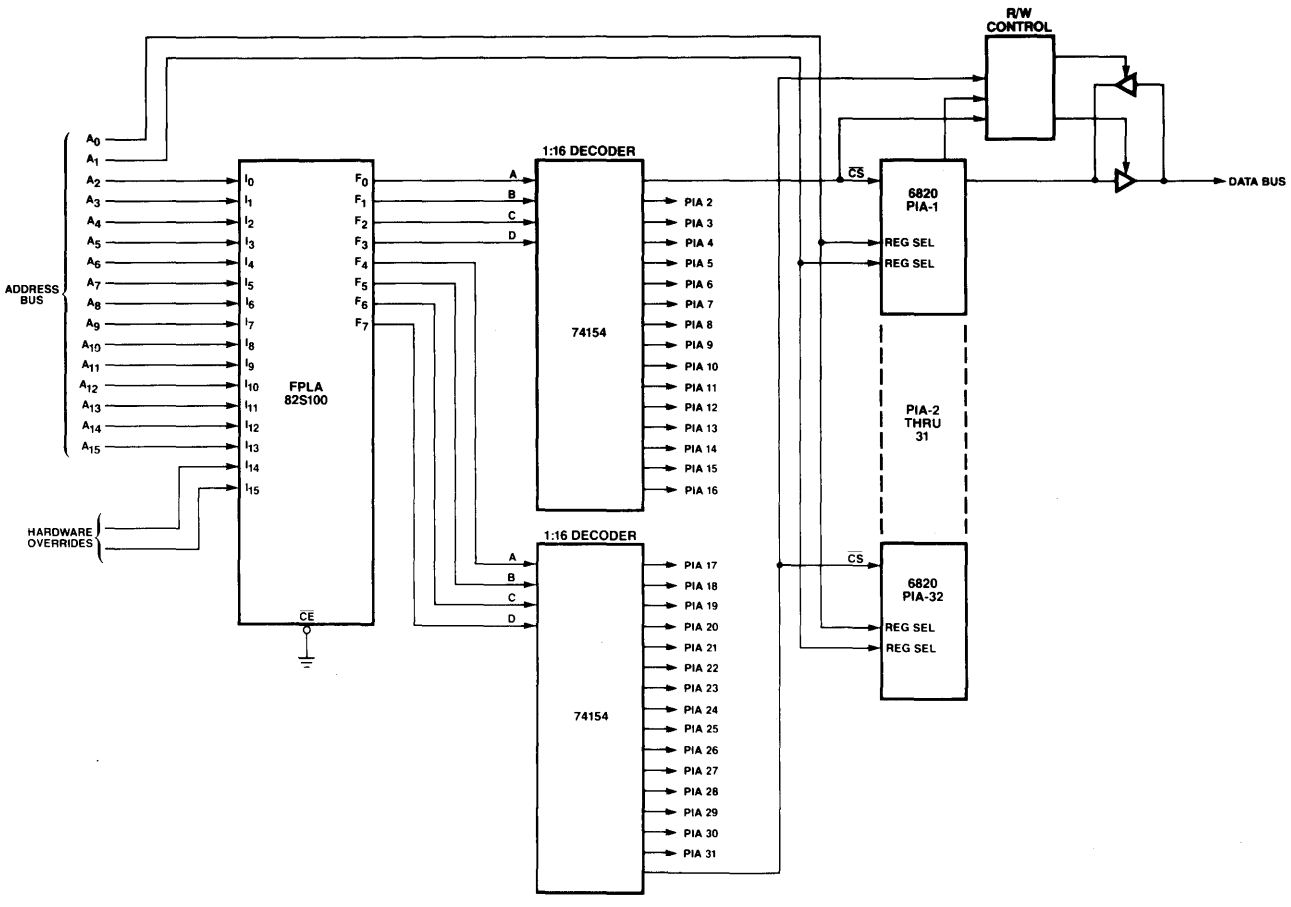


Figure 1.

PERIPHERAL ADDRESS DECODER 36

The circuit shown in Figure 1 is a simple but effective network in designs using microprocessors. The FPLA is used to fully decode the addresses of all peripheral circuits supporting the microprocessor system. If there are more than 8 peripherals, they can be easily accommodated by another FPLA. The outputs of the FPLA are connected directly to the chip select input of each device.

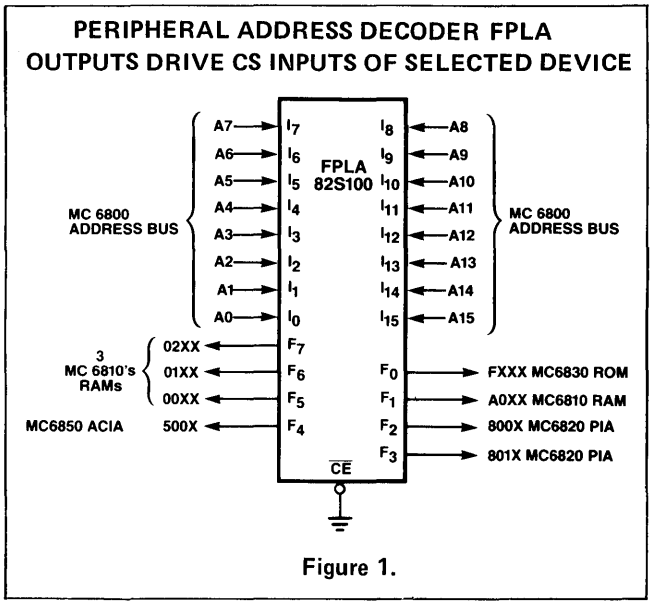


Figure 1.

MEMORY ADDRESS TRAP 37

The block diagram of Figure 1 shows an FPLA in conjunction with a small RAM for implementing a logically compact address trap, which can be readily modified. The FPLA will trap up to 48 locations in ROM and substitute a RAM location so that software subroutines may be easily called.

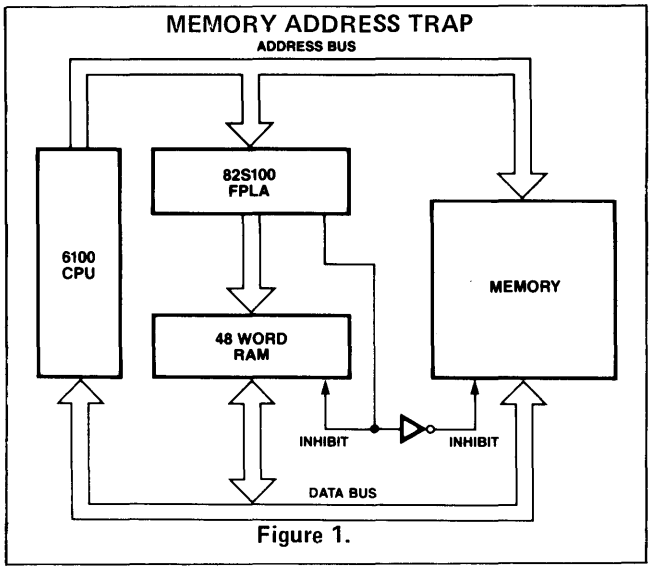


Figure 1.

MASKABLE INTERRUPT VECTOR GENERATOR 38

The circuit shown in Figure 1 generates maskable interrupt vectors for up to 8 interrupt sources. It also generates restart addresses and non-maskable interrupt addresses. Any of 8 possible interrupts sets the standard interrupt to the CPU by setting one of the interrupt flip-flops. When the CPU sends out the addresses which normally contains the vector address, the FPLA prioritizes the flip-flops so as to return the proper address. Each interrupt handler should clear its associated flip-flop via the shown output port. The device hold-off signal can be used to hold off the source of interrupts until each is serviced. The output addresses shown in the FPLA Program Table of Figure 2 represent a typical assignment.

INTERRUPT ADDRESS VECTOR GENERATOR FOR 6800 OR 650X MICROPROCESSORS

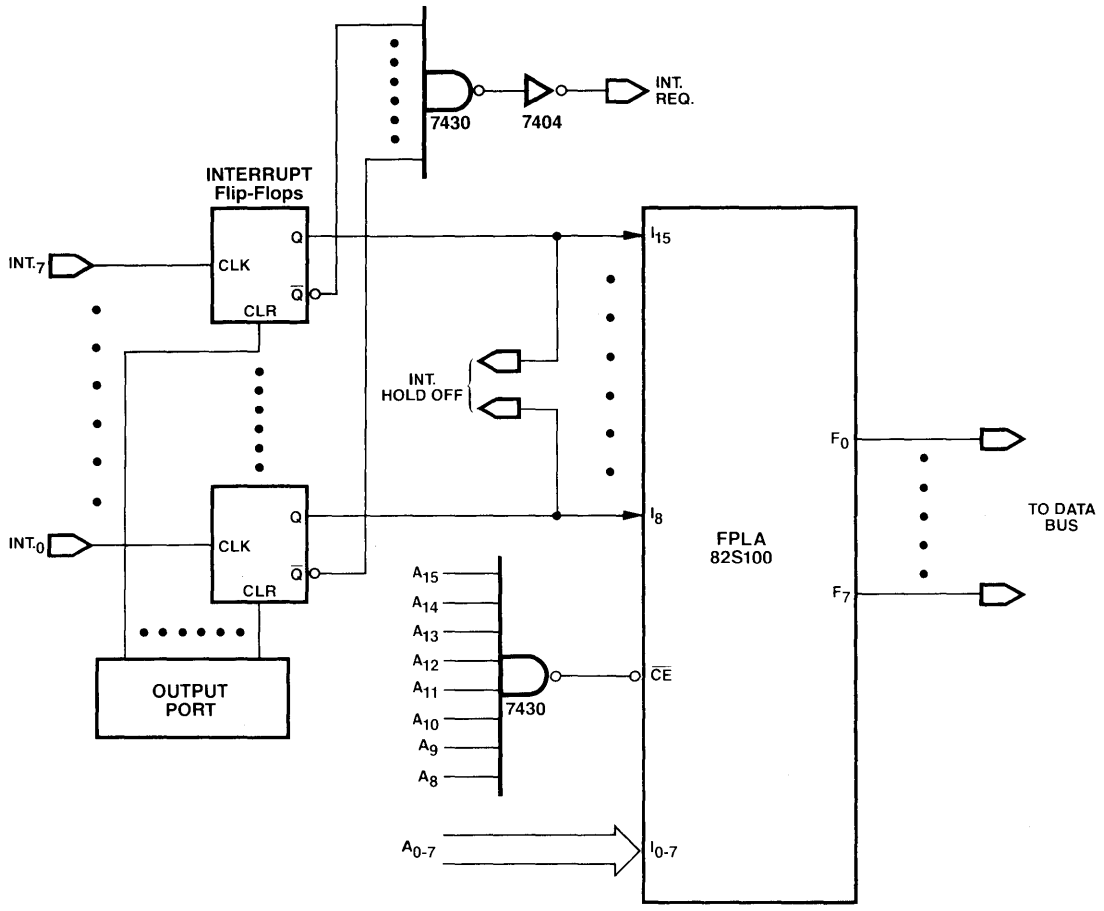


Figure 1.

TYPICAL FPLA PROGRAM TABLE

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H H H H H H H H — OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENTS
0	–	–	–	–	–	–	–	–	H	H	H	H	H	L	H	L	•	•	•	•	•	•	•	•	address of non-maskable interrupt routine
1	–	–	–	–	–	–	–	–	H	H	H	H	H	L	H	H	•	•	•	•	•	•	A	•	
2	–	–	–	–	–	–	–	–	H	H	H	H	H	H	L	L	•	•	•	•	•	•	•	•	address of restart routine
3	–	–	–	–	–	–	–	–	H	H	H	H	H	H	L	H	•	•	•	•	•	A	•	•	
4	–	–	–	–	–	–	–	H	H	H	H	H	H	H	H	L	•	•	•	•	•	•	•	•	address of highest priority (INT_0) interrupt handler
5	–	–	–	–	–	–	–	H	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	
6	–	–	–	–	–	–	H	L	H	H	H	H	H	H	H	L	•	•	A	•	•	•	•	•	address of second highest priority (INT_1) interrupt handler
7	–	–	–	–	–	–	H	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	
8	–	–	–	–	–	H	L	L	H	H	H	H	H	H	H	L	•	A	•	•	•	•	•	•	address of third highest priority (INT_2) interrupt handler
9	–	–	–	–	–	H	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	
16	–	H	L	L	L	L	L	L	H	H	H	H	H	H	H	L	A	A	•	•	•	•	•	•	address of second lowest priority (INT_6) interrupt handler
17	–	H	L	L	L	L	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	
18	H	L	L	L	L	L	L	L	H	H	H	H	H	H	H	L	A	A	A	•	•	•	•	•	address of lowest priority (INT_7) interrupt handler
19	H	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	

Figure 2.

15-INPUT PRIORITY ENCODER AND I.D. GENERATOR 39

In interfacing multiple devices to a mini or micro-computer bus, some sort of interrupt priority structure is generally needed. A conventional scheme uses a daisy chain circuit which allows a priority signal to pass if the device is not requesting an interrupt, and blocks it otherwise. In most computers the interrupting device is allowed to place its identifying code on the bus. If several devices share one P.C. board or chassis, one F.P.L.A. can be used to adjudicate priorities between the devices and generate the identifying codes, and thus save a large array of gates for each device serviced. The I/O assignment for the FPLA is shown in Figure 1. The FPLA program table is easily derived from the typical logic path in Figure 2, in which any arbitrary priority and device identification codes can be assigned.

FPLA IMPLEMENTS Nth ELEMENT OF DAISY-CHAIN PRIORITY INTERRUPT ENCODER

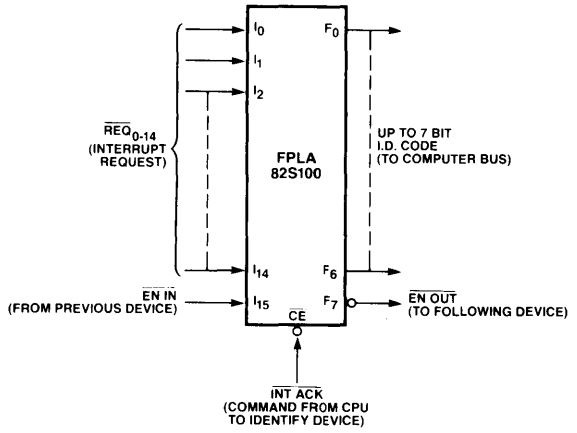


Figure 1.

TYPICAL LOGIC PATH WITHIN THE FPLA

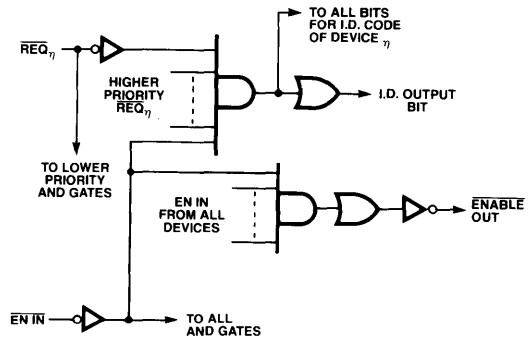


Figure 2.

FPLA SIMPLIFIES ALPHA-NUMERIC DISPLAY OPERATION 40

Controlling a cluster of 5 x 7 dot-matrix alpha-numeric displays is much more involved than numeric only, and generally requires a lot of circuitry which can be greatly reduced with FPLAs. The difficulty is that instead of displaying a whole character at once, as with 7-segment displays, multiplexing must occur on a column by column basis. Therefore the display must be fed information for part of each character, then illuminated, then the next part of each character and illuminated, etc.

The circuit shown in Figure 1 uses an FPLA to implement the necessary display control for any desired character-cluster lengths, obtained by simply reprogramming the FPLA. The circuit uses an H-P 5082-7150 display which requires all bits for column 1 of all characters to be serially clocked in. The display has a holding register, in this case 140 bits long (7 bits/column x 20 characters). When all 140 bits are clocked in, the serial clocking must stop and the column driver is turned on for the duration desired (1.5 ms). Then the column driver is turned off and the clocking begins for the next 140 bits (all of the column 2 bits) and column 2 is turned on.

Obviously all of the data for each column for all 20 characters must be readily available to this circuit, so a RAM was selected allowing external circuitry to load column data into the RAM. The circuit then takes this data and forms the 20 characters from it. In this case the RAM is loaded off of a transmission-line bus driven by a microprocessor. Placement of the data into the RAM is shown in Figure 2, and the microprocessor sends dot data to the RAM, not an ASCII character.

Close inspection of Figure 1 shows pin numbers and output bits mixed, but only to simplify PC layout. For instance, the RAM outputs feeding the 93L12 are mixed but not an error, so long as the RAM is being loaded with data in such a way as to form the character desired.

Since central control is afforded by the FPLA, the programming of the FPLA must be understood in order to know how the circuit works. With reference to Figure 3, note that all addresses are in octal. It takes 100 decimal memory locations to hold all of the date (5 cols/char x 20 chars). Assuming the RAM is loaded with the data in Figure 2, let's begin at address 0 set by the 93L66 counter, whose outputs also drive inputs to the 82S100 FPLA.

When the address is 0, column 5 is turned on by output F_7, but nothing will be displayed yet. After the 96L02 expires,

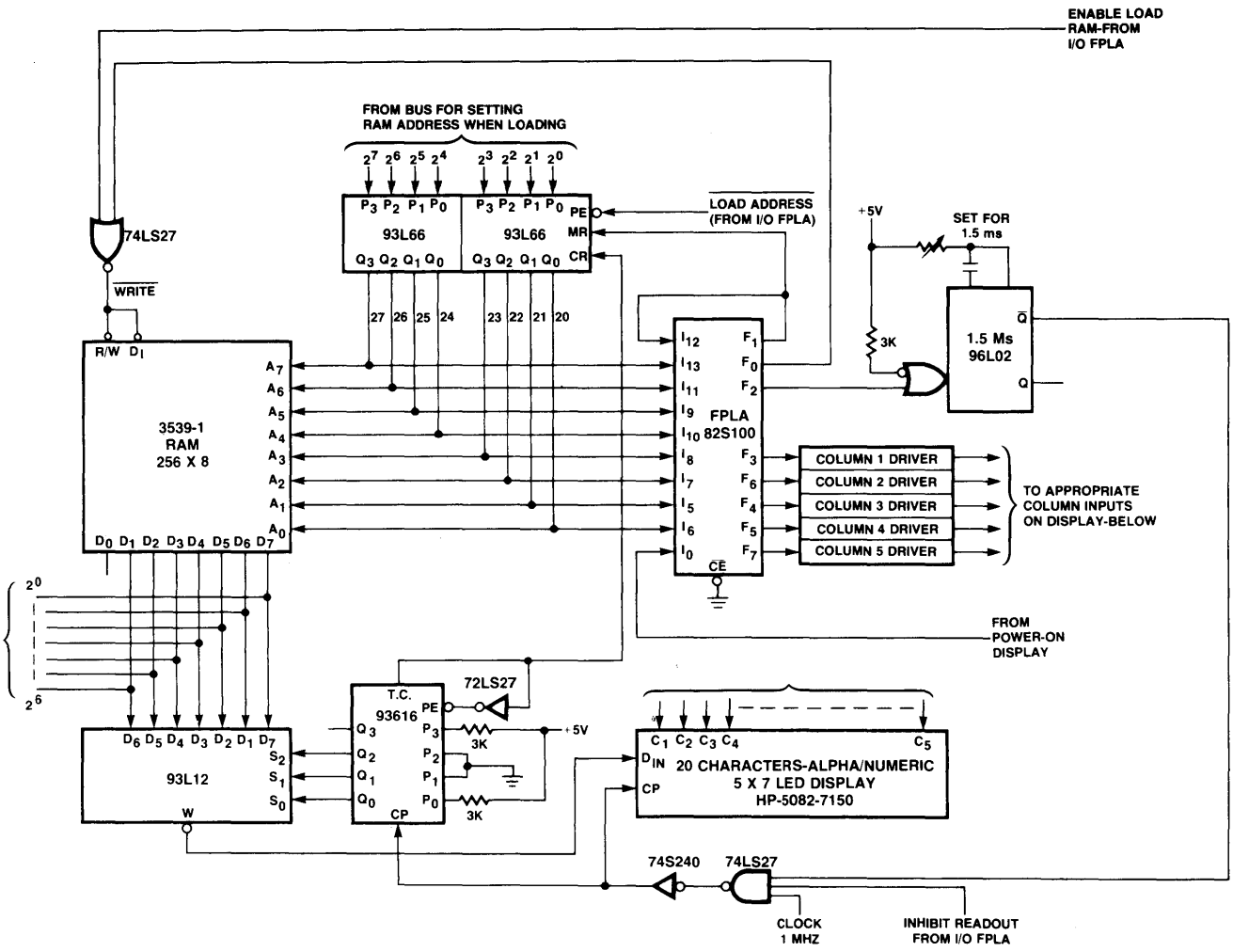


Figure 1.

data will begin to load into the display. Here's how it happens.

Since the address is 0, the RAM places the 7 column-1 bits for the character in address 0 on its output. The 93L16 is sitting at 001_2, thus the 93L12 is selecting a bit which appears at its W output, and is clocked into the display. At the next clock pulse, the 93L16 increments and the 93L12 selects the 2nd bit, which is subsequently clocked into the display. After 7 cycles the 93L16 hits top count, and the 93L66's are incremented causing address 1 to be referenced, which makes new data available to the 93L12. Top count also resets the 93L16 to octal 11, which is effectively 001_2. Note that the display and 93L16 use opposite trigger edges, so no race results. As the counter is incrementing nothing is happening at the 82S100, but as soon as address 24 is referenced the FPLA is activated. F_2 is enabled, which blocks the clock from further shifting-counting-selecting for the duration of 96L02 delay.

Simultaneously, F_3 enables the column-1 driver which enables all column-1 bits of each character. (Remember that the display has a holding register).

When the delay expires the clock is allowed to continue, data is shifted-counted-selected as before until the 93L66 reaches address 50, at which point the clock is again stopped and column 2 is enabled. This cycle is repeated until finally address 144 is reached, and the FPLA is again activated. F_1 is now enabled, resetting the address counter to zero so that the circuit recycles back to do all columns over again.

To avoid meaningless data from being displayed upon power-up, the FPLA can save additional circuitry by

supplying a CLR RAM signal. When power-on occurs, a timer enables FPLA input I_0 which is programmed to unconditionally reset the 93L66 counters to address zero, and enable the RAM to write. Now the clock is not inhibited so the counter begins counting (after the power-on I_0 goes away). But since F_1 is tied to I_{12} the counter addresses do not enable column drivers. Also, while counting, the RAM is writing zeros into itself, thus the RAM is being cleared at every address. Nothing else happens until the counter reaches 200, where the 82S100 is programmed to drop the F_0 output, and activate F_1. These will respectively disable RAM write and reset the 93L66 counter. This time at address 0, normal operation resumes because I_0 and I_{12} are not present.

The FPLA can also supply the additional function of clearing the RAM intentionally of its data. All that is required is to cause the 93L66 to be loaded with address 177, which the FPLA uses to force the circuit into the same clear-memory loop as the power-on did by enabling F_0 and F_1. This saves the trouble of loading each RAM address individually with zeros just to clear the display.

RAM MEMORY MAP

RAM ADR	Data Used For
00	Column 1 of character 1
↓	↓ ↓
23	Column 1 of character 20
24	Column 2 of character 1
↓	↓ ↓
47	Column 2 of character 20
50	Column 3 of character 1
↓	↓ ↓
73	Column 3 of character 20
74	Column 4 of character 1
↓	↓ ↓
117	Column 4 of character 20
120	Column 5 of character 1
↓	↓ ↓
143	Column 5 of character 20

CHARACTER NUMBERING OF ALPHA-NUMERIC DISPLAY

CHAR 20		CHAR 2	CHAR 1

Figure 2.

PROGRAM TABLE OF DISPLAY CONTROL FPLA

RAM ADDRESS ASSIGNMENT IS IN OCTAL, WHERE:
177 = MASTER CLEAR ADDRESS
XXX = POWER-ON OR CLEAR-MEMORY LOOP
200 = TERMINATE CLEAR-MEMORY LOOP

RAM ADDR$_8$	NO.	PRODUCT TERM INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL L	L	L	L	L	H	H	H
																		OUTPUT FUNCTION 7	6	5	4	3	2	1	0
000	0	–	–	L	L	L	L	L	L	L	L	L	–	–	–	–	L	A	•	•	•	•	A	•	•
024	1	–	–	L	L	L	H	L	L	H	L	L	–	–	–	–	L	•	•	•	•	A	A	•	•
050	2	–	–	L	L	L	L	H	H	L	L	L	–	–	–	–	L	•	A	•	•	•	A	•	•
074	3	–	–	L	L	L	H	H	H	H	L	L	–	–	–	–	L	•	•	•	A	•	A	•	•
120	4	–	–	L	L	H	H	L	L	L	L	L	–	–	–	–	L	•	•	A	•	•	A	•	•
144	5	–	–	L	L	H	L	H	L	H	L	L	–	–	–	–	L	•	•	•	•	•	•	A	•
177	6	–	–	L	L	H	H	H	H	H	H	H	–	–	–	–	L	•	•	•	•	•	•	A	A
XXX	7	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	•	•	•	•	•	•	A	A
XXX	8	–	–	L	H	–	–	–	–	–	–	–	–	–	–	–	L	•	•	•	•	•	•	•	A
200	9	–	–	H	H	L	L	L	L	L	L	L	–	–	–	–	L	•	•	•	•	•	•	A	•
I/O ASSIGNMENT		UNUSED		2^7	F_1	2^6	2^4	2^5	2^3	2^2	2^0	2^1	UNUSED				Power-on	COL. 5	COL. 2	COL. 4	COL. 3	COL. 1	REFRESH	RESET CNTR	ENABLE RAM WE

Figure 3.

Any digital instrument can be difficult to read when its measurements are displayed rapidly to high arithmetic precision, and are moderately noisy to begin with. The programmable discrete filter provides an inexpensive solution to this problem by performing a numerical smoothing operation on the data, as it arrives from the measurement device in BCD form. It is especially well suited in design situations where the raw signal being measured is inherently digital, and cannot be conditioned using analog techniques, such as a digital phase meter for navigational applications.

The design uses a single FPLA as a bus controller, data decoder, and fixed program sequencer to control an inexpensive 4-function calculator with display. This approach was selected primarily for its simplicity and low cost. But the "post-production" editing capability of the FPLA provides another important advantage: three program constants K, K-1, and B defined below are to be determined and updated periodically during instrument calibration in the field. Since efficient microcoding of the fixed program has left over fifty percent of the FPLA memory available for these variable parameters, each unit can be recalibrated several times before replacing the FPLA. In addition, storing the instrument adjustments digitally within the FPLA should tend to discourage unauthorized modification.

The circuit mechanizes the following filter algorithm:

$$F_n = \frac{(K-1)F_{n-1} + (X_n - B)}{K}$$

where, in the present application,

F_n = the next filtered value to be displayed after processing the current measurement.

F_{n-1} = the last filtered value computed and displayed.

X_n = a five-digit BCD measurement.

B = a constant offset correction determined during instrument calibration (up to five digits in length).

K = a single-digit filter weight, which can be custom programmed to match the filter to its operation environment.

To avoid unnecessary transients at startup, the first filtered value (F_1) is initialized to the first available corrected data value (X_1–B) before beginning the smoothing process.

The following table demonstrates the dramatic improvement in display readability that is possible with the discrete filter operating on a noisy data stream in real time:

Sample Number (n)	Raw Data (X_n)	Unfiltered Data (X_n–B)	Filtered Data (F_n)
1	124.87	123.45	123.45
2	126.67	125.25	123.65
3	126.87	125.45	123.85
4	120.77	119.35	123.35
5	126.57	125.15	123.55
6	126.77	125.35	123.75
7	121.57	120.15	123.35
8	127.47	126.05	123.65
9	122.37	120.95	123.35
10	125.67	124.25	123.45

The filter weight K and offset constant B used in this example are 9 and 1.42, respectively. Larger values of K produce heavier filtering (and greater noise rejection), but as K increases, the filter responds less rapidly to legitimate changes in the signal being measured. In fact, the discrete filter behavior is analogous to that of an RC low-pass filter whose time constant is

$$T = \frac{\Delta t}{\ln\left[\frac{K}{K-1}\right]}$$

where Δt is the time between measurement samples.

A flowchart of the 16-step algorithm, block diagram of the filter/display, FPLA program table, and detailed circuit diagram appear in Figures 1 thru 5.

In the block diagram of Figure 2, when the measurement cycle is complete and a valid five-digit BCD word is present on DIGIT 1 — DIGIT 5, the measurement device (not shown) pulls $\overline{\text{START}}$ momentarily LOW. This begins the computation cycle by clearing the Program Counter.

$\overline{\text{DONE}}$ is a handshake return from the filter to the measurement device which remains HIGH until the computation is complete. DIGIT 1 — DIGIT 5 should not be permitted to change until $\overline{\text{DONE}}$ goes LOW.

As shown in Figure 4 the first cycle in each program step is subdivided into 2 half-cycles: a 'Fixed Program' half-cycle and a 'Decode' half-cycle. During the first half-cycle, a digit address is always set up and latched for the Digit Selector Network. If the current program step calls for a fixed operation, key closures are also simulated continuously during this half-cycle. (The on-chip key debounce feature of the calculator dictates that each key closure be simulated without interruption for at least 11.4 msec, followed by a pause of the same duration).

During the second half-cycle, the selected data is decoded, and key closures are simulated for the appropriate digit key.

If the current program step does not call for a BCD digit to be decoded, the FPLA addresses a unique sixth digit during the 'Fixed Program' half-cycle. This sixth digit is hardwired to a binary 15, which is ignored during the 'Decode' half-cycle since it is not a valid BCD code.

FILTER IN/OUT must be LOW for at least one computation cycle after the unit is turned on, in order to 'unlock' the calculator chip and initialize the filter. A panel switch can be used to control this signal; this would permit the operator to select either filtered or unfiltered data for display. (Note that the bias offset correction is applied in either mode).

An external system clock controls the program sequence. CLOCK should be approximately 40 Hz, 50% duty cycle, and can be free-running.

The FPLA steps the calculator through the program, by simulating multiplexed key closures in the sequence dictated by the Program Counter.

When the last operation is completed, the FPLA is disabled, $\overline{\text{DONE}}$ goes LOW, and the Program Counter remains locked in its final state. The calculator displays the results of the computation until the next $\overline{\text{START}}$ pulse arrives from the external measurement device. FILTER IN/OUT may be permitted to change state at any time during this pause for display.

Each program step, corresponding to a single state of the Program Counter, consists of two separate cycles of equal time duration. The FPLA is always enabled for the first cycle and disabled for the second.

16-STEP ALGORITHM ADAPTED TO POLISH FORM

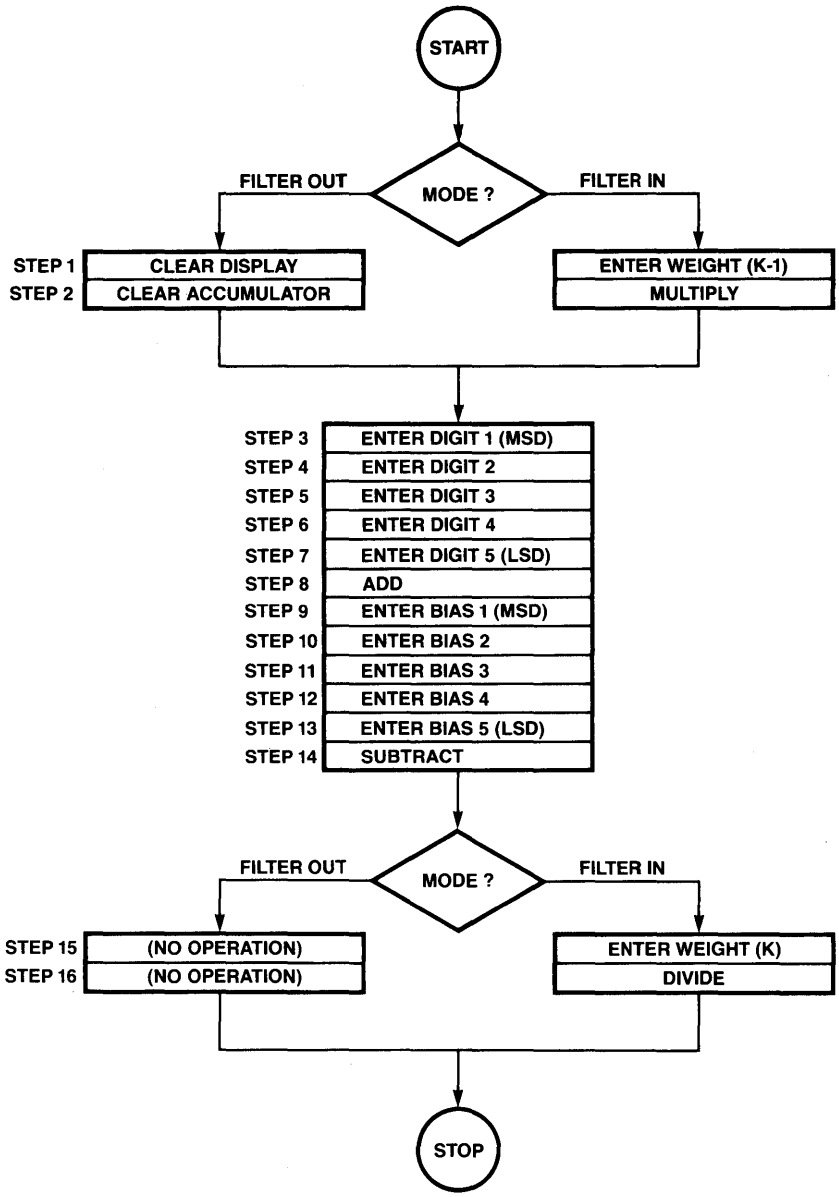


Figure 1.

DISCRETE FILTER/DISPLAY BLOCK DIAGRAM

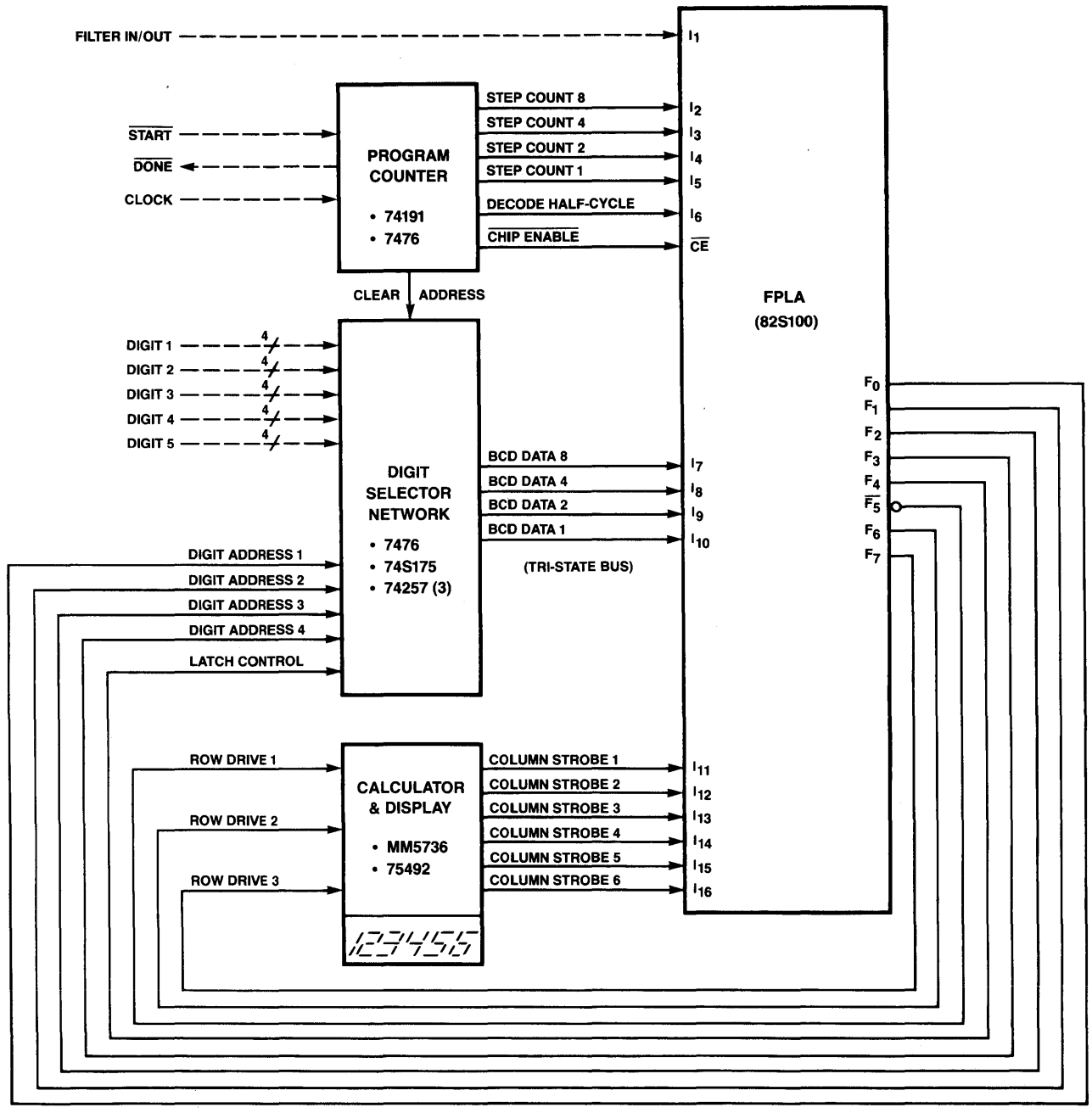


Figure 2.

FPLA PROGRAM TABLE
X VALUES IN THE TABLE ARE DETERMINED AT CALIBRATION.
P-TERMS 30 THRU 48 CAN BE USED FOR SUBSEQUENT FIELD CALIBRATION, IF NECESSARY, TO REPLACE P_3 AND P_{11} THRU P_{17}.

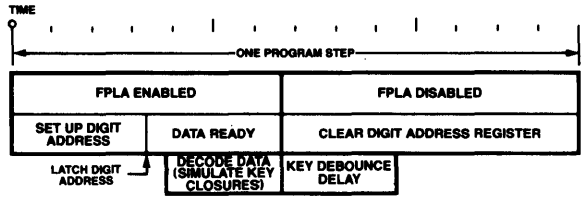
FUNCTION	NO.	PRODUCT TERM / INPUT VARIABLE																
		I 1	I 2	I 3	I 4	I 5	I 6	I 7	I 8	I 9	I 10	I 11	I 12	I 13	I 14	I 15	I 16	
Clear Display	1	L	L	L	L	L	L	–	–	–	–	L	H	H	H	H	H	
Clear Accumulator	2	L	L	L	L	H	L	–	–	–	–	L	H	H	H	H	H	
Enter Weight (K-1)	3	H	L	L	L	L	L	–	–	–	–	X	X	X	X	X	X	
Multiply	4	H	L	L	L	H	L	–	–	–	–	H	H	H	H	L	H	
Address Digit 1 (MSD)	5	–	L	L	H	L	L	–	–	–	–	–	–	–	–	–	–	
Address Digit 2	6	–	L	L	H	H	L	–	–	–	–	–	–	–	–	–	–	
Address Digit 3	7	–	L	H	L	L	L	–	–	–	–	–	–	–	–	–	–	
Address Digit 4	8	–	L	H	L	H	L	–	–	–	–	–	–	–	–	–	–	
Address Digit 5 (LSD)	9	–	L	H	H	L	L	–	–	–	–	–	–	–	–	–	–	
Add	10	–	L	H	H	H	L	–	–	–	–	H	H	H	L	H	H	
Enter Bias Digit 1 (MSD)	11	–	H	L	L	L	L	–	–	–	–	X	X	X	X	X	X	
Enter Bias Digit 2	12	–	H	L	L	H	L	–	–	–	–	X	X	X	X	X	X	
Enter Bias Digit 3	13	–	H	L	H	L	L	–	–	–	–	X	X	X	X	X	X	
Enter Bias Digit 4	14	–	H	L	H	H	L	–	–	–	–	X	X	X	X	X	X	
Enter Bias Digit 5 (LSD)	15	–	H	H	L	L	L	–	–	–	–	X	X	X	X	X	X	
Subtract (or add)	16	–	H	H	L	H	L	–	–	–	–	X	X	X	X	X	X	
Enter Weight (K)	17	H	H	H	H	L	L	–	–	–	–	X	X	X	X	X	X	
Divide	18	H	H	H	H	H	L	–	–	–	–	H	H	H	H	H	L	
No Operation	18	L	H	H	H	H	L	–	–	–	–	–	–	–	–	–	–	
Decode BCD '0'	20	–	–	–	–	–	H	L	L	L	L	L	H	H	H	H	H	
Decode BCD '1'	21	–	–	–	–	–	H	L	L	L	H	H	L	H	H	H	H	
Decode BCD '2'	22	–	–	–	–	–	H	L	L	H	L	H	H	L	H	H	H	
Decode BCD '3'	23	–	–	–	–	–	H	L	L	H	H	H	H	H	L	H	H	
Decode BCD '4'	24	–	–	–	–	–	H	L	H	L	L	H	H	H	H	L	H	
Decode BCD '5'	25	–	–	–	–	–	H	L	H	L	H	H	H	H	H	H	L	
Decode BCD '6'	26	–	–	–	–	–	H	L	H	H	L	H	L	H	H	H	H	
Decode BCD '7'	27	–	–	–	–	–	H	L	H	H	H	H	H	L	H	H	H	
Decode BCD '8'	28	–	–	–	–	–	H	H	L	L	L	H	H	H	L	H	H	
Decode BCD '9'	29	–	–	–	–	–	H	H	L	L	H	H	H	H	H	L	H	

	ACTIVE LEVEL							
	H	H	H	H	L	H	H	H
	OUTPUT FUNCTION							
	1	2	3	4	5	6	7	8
	•	•	A	•	A	•	•	A
	•	•	A	•	A	•	•	A
	•	•	A	•	A	X	X	•
	•	•	A	•	A	•	•	A
	A	•	•	A	A	•	•	•
	A	•	•	•	A	•	•	•
	•	A	•	A	A	•	•	•
	•	A	•	•	A	•	•	•
	•	•	A	A	A	•	•	•
	•	•	A	•	A	•	•	A
	•	•	A	•	A	X	X	•
	•	•	A	•	A	X	X	•
	•	•	A	•	A	X	X	•
	•	•	A	•	A	X	X	•
	•	•	A	•	A	X	X	•
	•	•	A	•	A	•	•	A
	•	•	A	•	A	X	X	•
	•	•	A	•	A	•	•	A
	•	•	A	•	A	•	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	•	A	•
	•	•	•	•	•	•	A	•
	•	•	•	•	•	•	A	•
	•	•	•	•	•	•	A	•

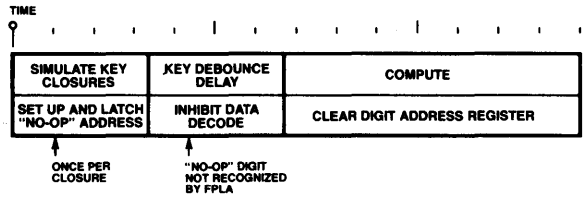
I/O ASSIGNMENT	FILTER IN/OUT (1=IN)	STEP COUNT 8	STEP COUNT 4	STEP COUNT 2	STEP COUNT 1	DECODE HALF-CYCLE	BCD8	BCD4	BCD2	BCD 1	COLUMN STROBE 1	COLUMN STROBE 2	COLUMN STROBE 3	COLUMN STROBE 4	COLUMN STROBE 5	COLUMN STROBE 6	DIGIT ADDRESS 1	DIGIT ADDRESS 2	DIGIT ADDRESS 3	DIGIT ADDRESS 4	LATCH CONTROL	ROW DRIVE 1	ROW DRIVE 2	ROW DRIVE 3

Figure 3.

SYSTEM TIMING DIAGRAM. CLOCK IS 40 HZ, 50% DUTY CYCLE



a) DATA DECODE. STEP COUNT = 0010→0110. ADDRESS SET UP NOT TO SCALE.



b) FIXED PROGRAM STEP. STEP COUNT = 0000, 0001, 0111→1111

Figure 4.

CIRCUIT DIAGRAM OF DISPLAY CONDITIONER

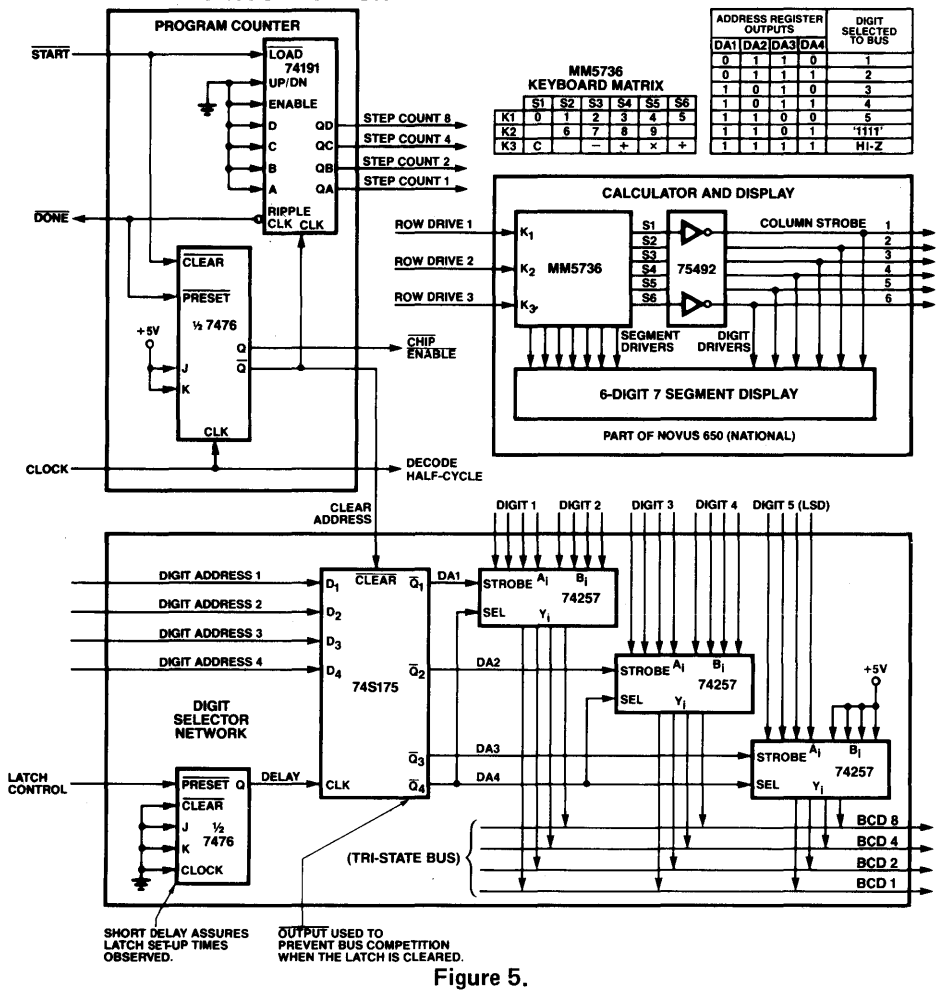


Figure 5.

There are numerous applications where a quick and accurate measurement of rate is desired. But in many of these cases, the events do not occur very often, and so they require a long time before a reading is obtained. Usually an analog low-pass filter is used to filter and derive the DC component of a monostable, triggered by the event. The voltage level is then directly proportional to the rate. However, in order to minimize errors due to the charging and discharging of the filter, long time constants are used, which adversely affect settling times. As illustrates in Figure 1, an accurate reading can only be made if the filter output is sampled when V(avg) equals V(charge) or V(discharge), as opposed to sampling at constant time intervals. Furthermore, if the proper sampling time as a function of input rate were known, one could compensate for filters with a faster rise-time, and thus obtain readings much sooner. This technique can be incorporated in a rate monitor system as shown in the circuit of Figure 2. The FPLA is used as a look-up table addressed by the counters after the monostable (MONO) goes LOW. The output counters are loaded every time a product term is activated, causing output F_7 to go HIGH. When an event occurs, the output counter has stored in it the proper sampling time (Ts), and will count down to zero during the monostable time. When it reaches zero, the A/D will be triggered.

SAMPLING FOR MAXIMUM RATE MEASUREMENT ACCURACY.
Tm, $\overline{Tm}$ ARE RESPECTIVELY THE ON AND OFF TIMES OF THE MONOSTABLE

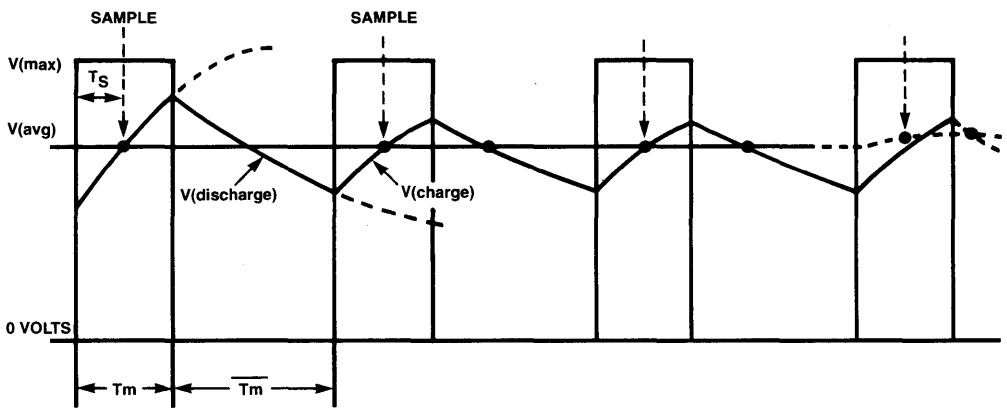


Figure 1.

The value of Ts varies with V(max), Tm and the RC time constant of the averager. It can be computed for various values of $\overline{Tm}$ as follows:

$$V_1 = V_2 \exp(-\overline{Tm}/RC) \qquad (1)$$

$$V_2 = Vmax + (V_1 - Vmax) \exp(-Tm/RC) \qquad (2)$$

V_2 can be found by substituting (2) into (1). Each corresponding value of Ts to be digitized in the FPLA is then obtained by solving:

$$Vavg = Vmax \frac{Tm}{Tm + \overline{Tm}} = V_2 \exp(-Ts/RC)$$

This circuit can be used in many industrial, medical and control applications. Using Heart Rate as a example, Tm = 150 msec, implying a maximum rate of 400BPM. Minimum heart rate is a function of the number of inputs to the FPLA, and the counter clock rate. Using the Signetics 82S100, and 1 kHz, the minimum H.R. = 1 BPM. Since one FPLA output must be used as clock enable only 7 outputs remain. These should be sufficient, for in this case Ts (max) = Tm/2 = 75. As a result the 48 product terms have to be used sparingly, but in most cases this provides more than adequate precision. We can see how the FPLA lends itself to this kind of application for, if we were to use a PROM, it would require a 16K x 8 configuration.

LOW FREQUENCY EVENT MONITOR

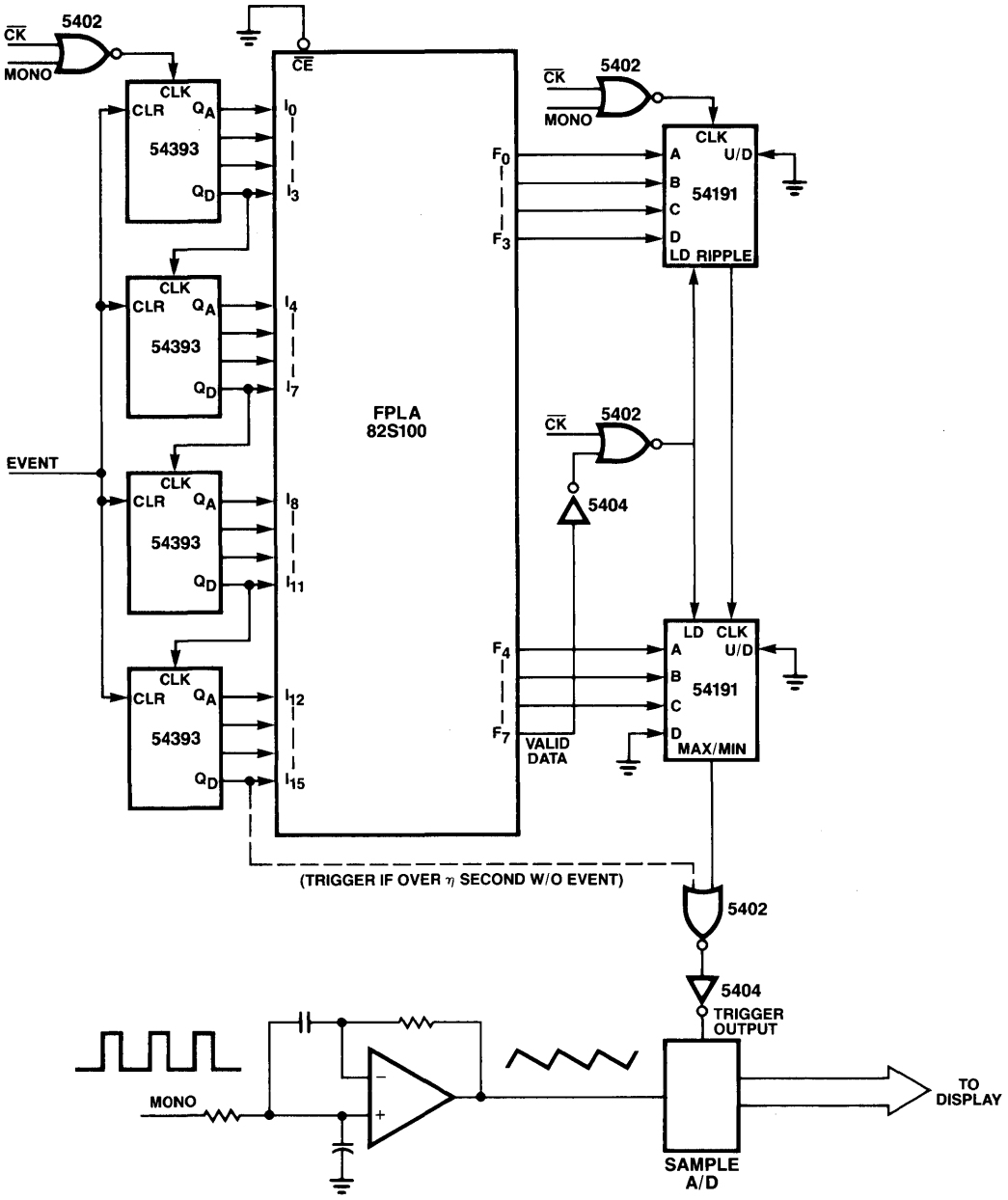


Figure 2.

The circuit shown in Figure 1 is a logic analyzer which can store 8 channels of data at rates up to 4 MHz and allow viewing of the data on any oscilloscope. A flip-flop type input circuit enables spikes and glitches far shorter than the clock to be detected. A trigger word detector allows the recording sequence to begin on any combination of input data.

The FPLA functions as the control element for the device. Switch closures, three clock phases (2 for recording and 1 fixed for playback), counter stages, and latches are fed into the FPLA which, in turn, generates Write enable for the memory, S-R information for the latches, and increment pulses for the counters.

There are two modes of operation: record and real time. In record, pressing the record switch arms the devices, so it can record when the trigger word appears. Recording stops depending on the position of the "time frame select" switch and a counter indicating the number of cycles since the start of recording. This way either 1/8, 1/2, or 7/8 of the total period may be displayed before the trigger word. After recording, the device goes into playback mode, except that the device is automatically re-armed after each playback cycle.

The input data first goes to a comparator which serves as an impedance buffer and a variable threshold detector for different logic families. The two latches gather data during alternate write cycles. With the D flip-flop and the EX-OR gates, they function as data change detectors, so only truly useful data is written in memory.

The playback cycle is carried out at 200 kHz; fast enough to avoid flicker, but slow enough for most scopes. The horizontal position is taken from the memory address, while the vertical position is generated from the data and the output multiplex address. The use of an FPLA allows easy modification of operating modes and timing. Since the FPLA is a Schottky device, the maximum clock frequency can be increased simply by using faster memory.

LOGIC ANALYZER

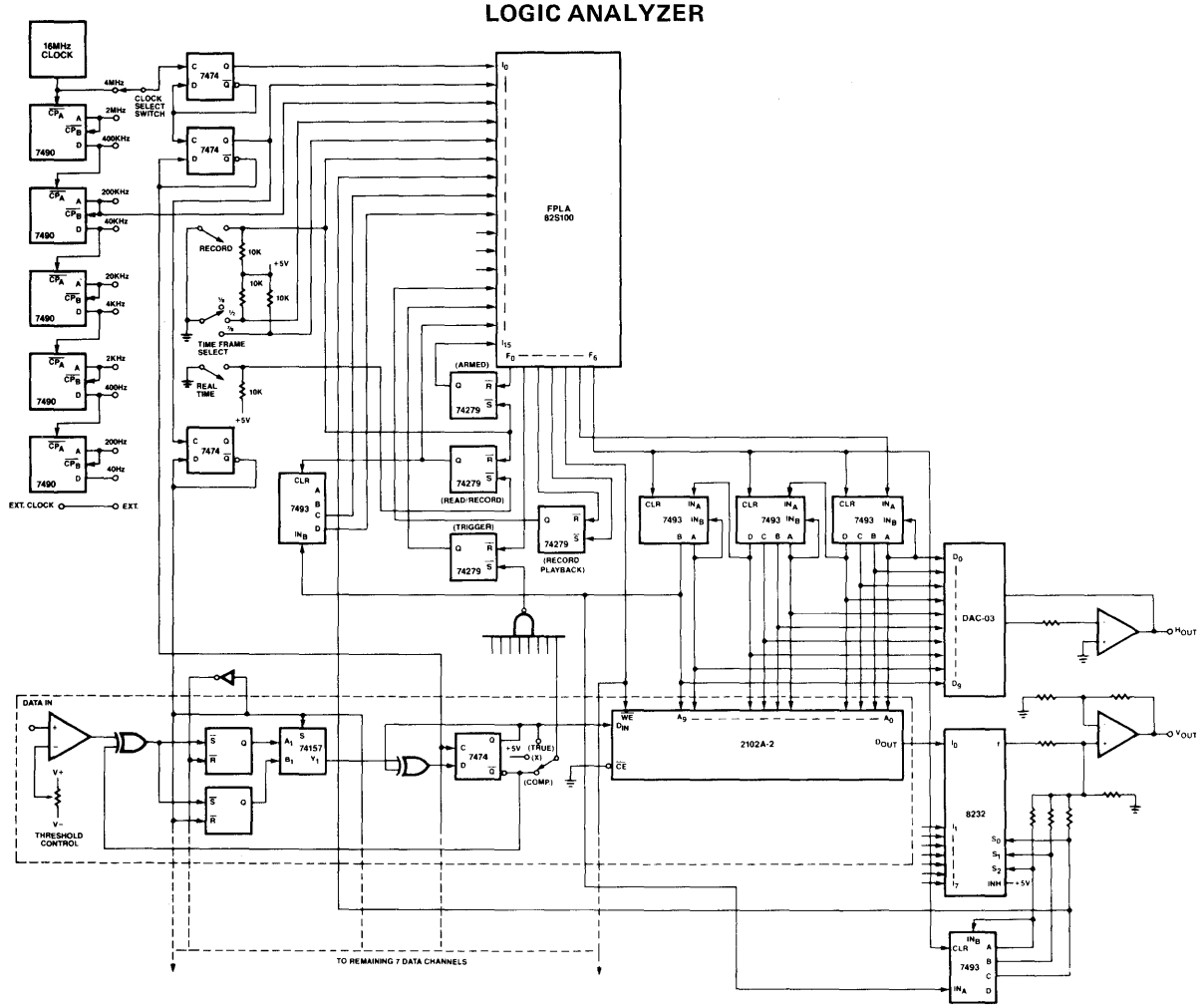


Figure 1.

In the design of any TV display it is necessary to develop the standard SYNC and BLANKING signals to drive the monitor. Although there are ICs which develop these signals, their beam position counter signals are not brought out to external pins. Therefore, to develop video one must duplicate the counter, and nothing will be saved. The circuit shown in Figure 1 uses an FPLA to develop SYNC and BLANKING, plus any video that remains fixed on the TV. Since the outputs of the vertical and horizontal beam position counter are available at the input to the FPLA, any combination of these signals can be used to develop fixed video on the monitor. This fixed video can be anything, such as ruling lines for a TV typewriter, or the background for a video game. For example, in designing a Pong-like video game the FPLA could be used to make the border, the net across the center of the screen, a foul line around the periphery of the court, plus all the necessary SYNC and BLANKING. The remaining circuit would generate the two movable paddles and the ball.

The circuit in Figure 1 generates all necessary signals to operate a standard T.V. monitor in non-interlaced mode with 262 lines per frame, and 60 frames per second. It follows that 262 lines/frame x 60 frames/sec = 15,720 lines/sec → 63.6 μsec/line.

SYNC VIDEO GENERATOR CIRCUIT

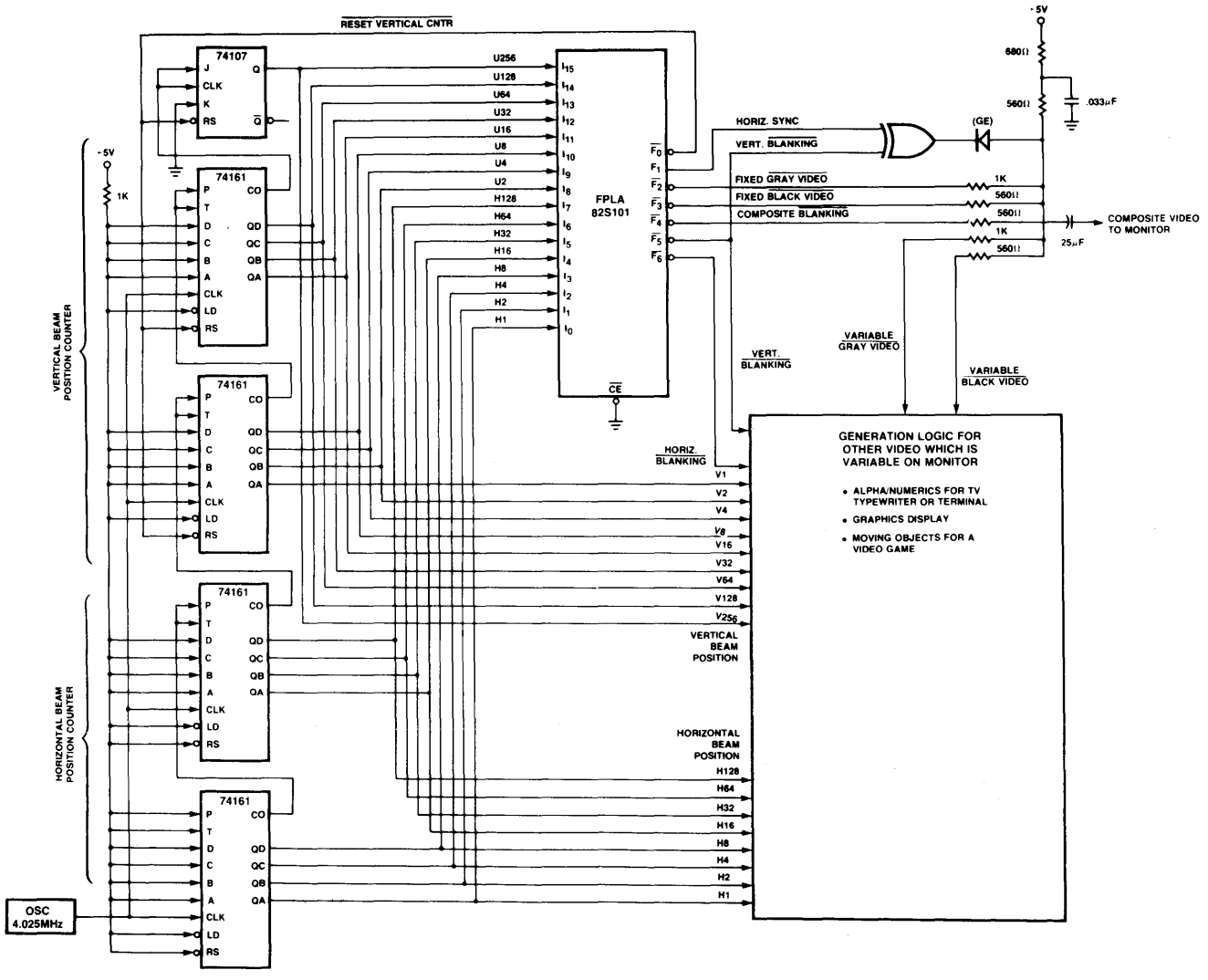
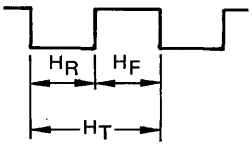


Figure 1.

Assuming each horizontal time of 63.6 μsec to be divided into 256 time units called picture elements (pels), we obtain 63.6 μsec/256 = 248.44 ns/(time unit) → 4.025 MHz. The TV monitor visible area is partitioned timewise as shown in Figure 2. The total horizontal time (H_T) is composed of two parts:

1) The forward visible time (H_F)
2) The retrace time (H_R)



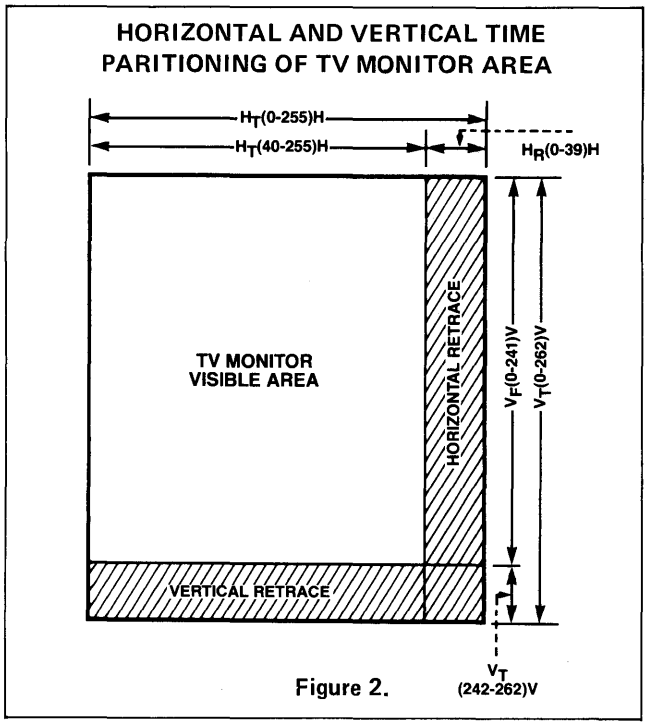
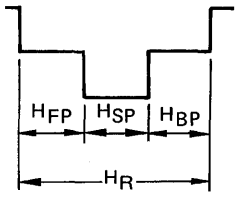


Figure 2.

Where $H_T = H_F + H_R = 63.6\mu sec$

The horizontal retrace time (H_R) is composed of three parts:

1) Front Porch (H_{FP})
2) Horizontal Sync Pulse (H_{SP})
3) Back Porch (H_{BP})



From the Radio Engineers Handbook we get the following information:

$H_R = .155\ H_T = 9.9\ \mu sec = 40$ pels
$H_{FP} = .02H_T = 1.3\ \mu sec = 5$ pels
$H_{SP} = .08H_T = 5.1\ \mu sec = 19$ pels
$H_{BP} = .06H_T = 3.816\mu sec = 16$ pels
$V_R = 0.75\ V_T = 20$ lines

The horizontal sync signals can be developed by using an eight bit counter, operating modulo 256 at the picture element rate of 4.025 MHz, with its outputs connected to an FPLA. There the appropriate counts will be decoded to generate the sync signals as follows:

Count	Signal	Count	Signal
0 → 39	H_R	25 → 39	H_{BP}
0 → 4	H_{FP}	40 → 255	H_F
5 → 23	H_{SP}		

Every time the horizontal counter wraps back to all zeros, the vertical counter will be incremented by one. There will be 20 lines of vertical retrace (V_R), and 242 lines of vertical visible time (V_F) for a total vertical time (V_T) of 262 lines. Starting at the top of the screen the 9 bit vertical counter is all zeros, and is incremented by one at the end of each horizontal forward time (right hand edge of screen). When the counter reaches 241, the vertical blanking signal will be enabled and the polarity of horizontal sync reversed. The counting of horizontal lines continues until count 262 is reached, at which time the vertical counter is asynchronously reset to zero and the vertical blanking signal turned off, giving:

Count	Signal
0-241	V_F
242-263	V_R (reverse polarity of horizontal sync)

The logic equation set for decoding the appropriate counts with the FPLA is tabulated in Figure 3, while the corresponding FPLA Program Table is shown in Figure 4.

FPLA LOGIC EQUATION SET IN TERMS OF WEIGHTED BINARY INPUTS FROM THE COUNTER

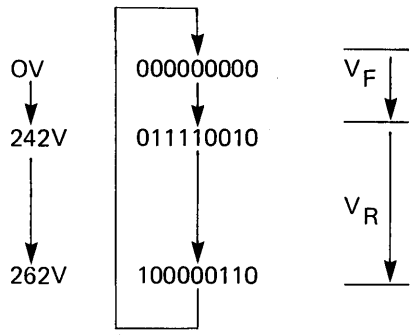
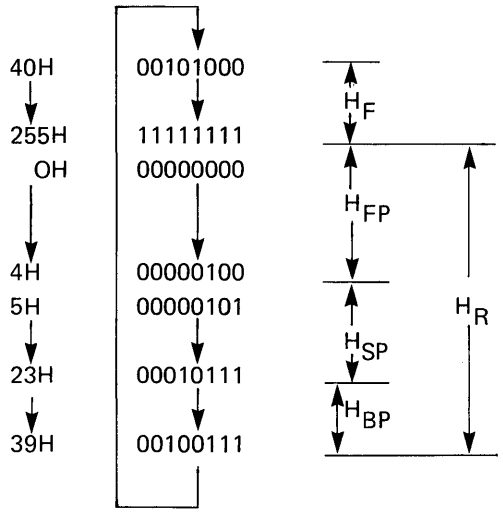
LOGIC EQUATION	ADDRESS SEQUENCE
Reset Vertical CNTR = 256V • 4V • 2V (262V) Vertical Blanking = (242V - 262V) $(\overline{256V}\bullet 128V\bullet 64V\bullet 32V\bullet 16V\bullet \overline{8V}\bullet \overline{4V}\bullet 2V)+$ $+ (\overline{256V}\bullet 128V\bullet 64V\bullet 32V\bullet 16V\bullet \overline{8V}\bullet 4V)+$ $+ (\overline{256V}\bullet 128V\bullet 64V\bullet 32V\bullet 16V\bullet 8V\bullet)+$ $+ (256V)$	
Horizontal Blanking = (OH-39H) $(\overline{128H}\bullet\overline{64H}\bullet\overline{32H})+$ $+ (\overline{128H}\bullet\overline{64H}\bullet 32H\bullet\overline{16H}\bullet\overline{8H})$ Horizontal SYNC = (5H-23H) $(\overline{128H}\bullet\overline{64H}\bullet\overline{32H}\bullet\overline{16H}\bullet\overline{8H}\bullet 4H\bullet\overline{2H}\bullet 1H)+$ $+ (\overline{128H}\bullet\overline{64H}\bullet\overline{32H}\bullet\overline{16H}\bullet\overline{8H}\bullet 4H\bullet 2H)+$ $+ (\overline{128H}\bullet\overline{64H}\bullet\overline{32H}\bullet\overline{16H}\bullet 8H)+$ $+ (\overline{128H}\bullet\overline{64H}\bullet\overline{32H}\bullet 16H\bullet\overline{8H})$	

Figure 3.

FPLA PROGRAM TABLE

COMMENTS	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H	L	L	L	L	L	H	L
																		OUTPUT FUNCTION: 7	6	5	4	3	2	1	0
RESET	0	H	–	–	–	–	–	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
HORIZONTAL BLANKING	1	L	H	H	H	H	L	L	H	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	2	L	H	H	H	H	L	H	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	3	L	H	H	H	H	H	–	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	4	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
VERTICAL BLANKING	5	–	–	–	–	–	–	–	–	L	L	L	–	–	–	–	–	A	A	•	A	•	•	•	•
	6	–	–	–	–	–	–	–	–	L	L	H	L	–	–	–	–	A	A	•	A	•	•	•	•
HORIZONTAL SYNC	7	–	–	–	–	–	–	–	–	L	L	L	L	L	H	L	H	A	•	•	•	•	•	A	•
	8	–	–	–	–	–	–	–	–	L	L	L	L	L	H	H	–	A	•	•	•	•	•	A	•
	9	–	–	–	–	–	–	–	–	L	L	L	L	H	–	–	–	A	•	•	•	•	•	A	•
	10	–	–	–	–	–	–	–	–	L	L	L	H	L	–	–	–	A	•	•	•	•	•	A	•
	11																								
SPARE TERMS AVAILABLE FOR MAKING FIXED BLACK OR GRAY VIDEO	12																								
	13																								
	14																								
	15																								
	16																								
	17																								
	18																								
	19																								
	20																								
	21																								
	22																								
	23																								
	24																								
	25																								
	26																								
	27																								
	28																								
	29																								
	30																								
	31																								
	32																								
	33																								
	34																								
	35																								
	36																								
	37																								
	38																								
	39																								
	40																								
	41																								
	42																								
	43																								
	44																								
	45																								
	46																								
	47																								
I/O ASSIGNMENT		256V	128V	64V	32V	16V	8V	4V	2V	128H	64H	32H	16H	8H	4H	2H	1H	SPARE	H_R	V_R	BLANKING	BLACK VIDEO	GRAY VIDEO	H_{SP}	RESET

Figure 4.

The circuit shown in Figure 1 provides tennis court boundaries and scorekeeping capabilities for electronic games, such as the one introduced in EDN August 5, 1976. The TV screen is divided into a matrix of 256 by 256 squares. The FPLA serves to connect the squares so that they take on the shape of a tennis court. Sync signals are generated along with other signals that are necessary for determining which boundary the ball has hit. The score is displayed 0 to 15 using an 8 segment format. The score is disabled while the ball is rallied. The FPLA is particularly suited to this application since the scoreboard requires 16-input AND gates.

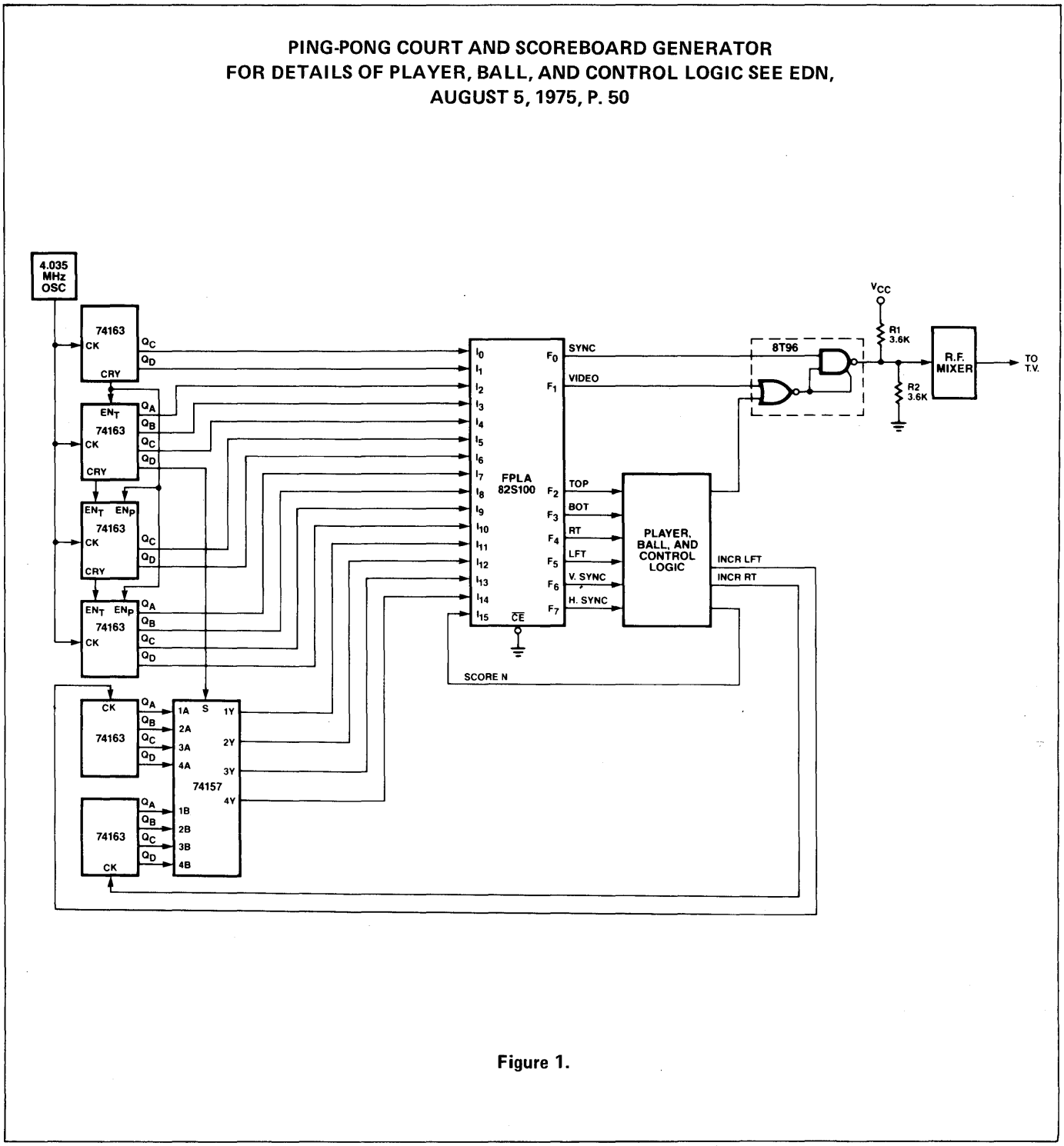


Figure 1.

A TV video display can be controlled via incoming data by using an FPLA to detect machine commands as data from a tape reader. Although a ROM could supply the necessary functions, it cannot use its extra inputs and outputs to form flip-flops that can be set internally by selected character or groups of characters. This technique is used in the proposed design to detect the order of characters passing through the tape reader for creating a "tape backward" alarm, used when reading tape by hand. The block diagram of Figure 1 shows an FPLA used to control the interface between a TV and a tape reader.

The system most critical control function is that of detecting invalid characters, including parity errors, which can be avoided by strobing the outputs, giving them time to settle. Carriage return (CR) is used to reset the character counter in the TV display, while line feed (LF) is used to advance the line counter by one. The "A" character designates a "time" message and is used to select one of two message lengths to detect dropouts or run-ons.

These functions are implemented in the FPLA connected as shown in Figure 2, and suitably programmed. Nothing that CR is always followed by LF, if the first F/F is set by CR

BLOCK DIAGRAM OF T.V. TAPE READER

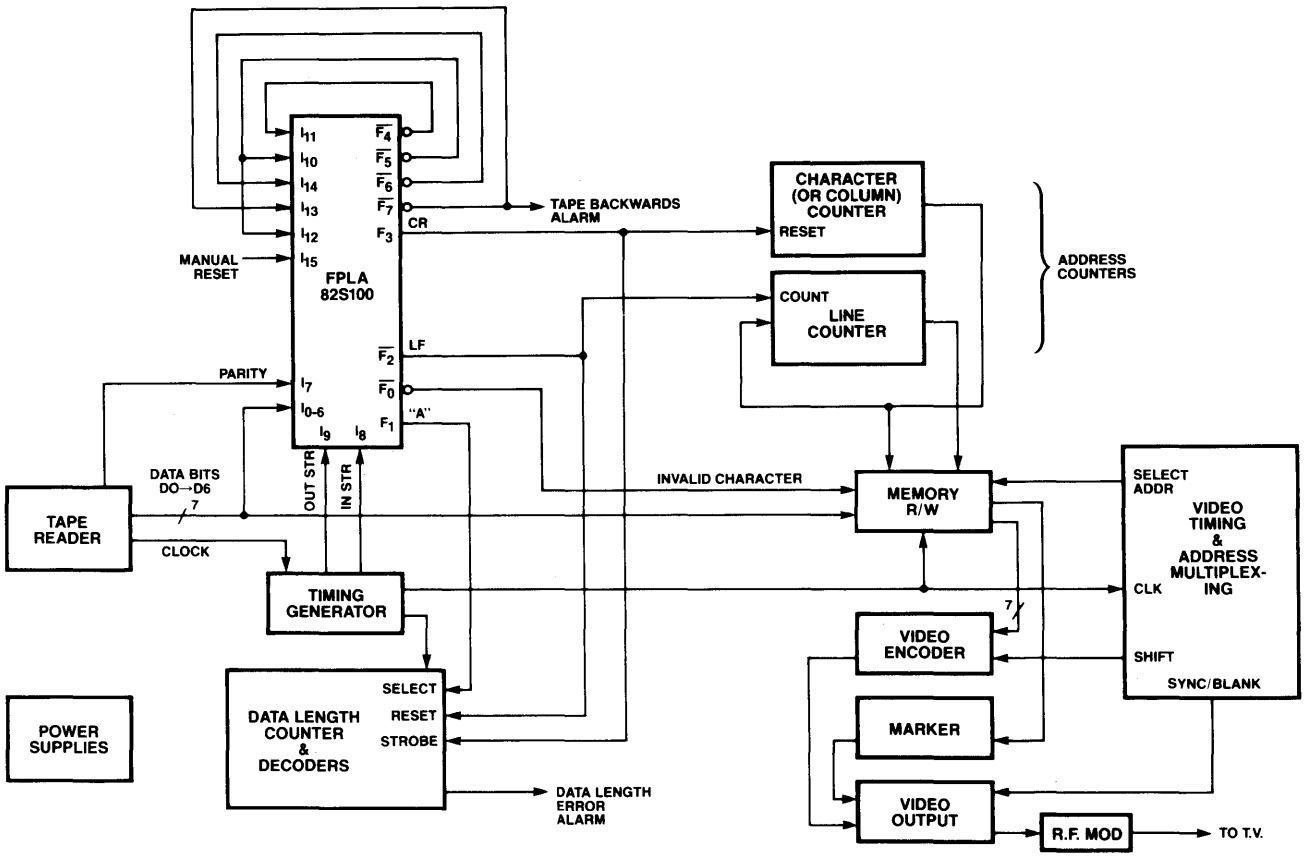


Figure 1.

and reset by LF, its output state is determined by the last of the two characters read. The output that is HIGH after CR is received is "AND" gated with a character that is always present in a message, and then used to set a second F/F. If the tape is being read in the right direction, the 1st F/F is reset before its output can be used to set the 2nd F/F (which goes to the alarm). This approach permits any series of data to be checked for a certain order of characters to flag special conditions, where data can be a set of sensors outputs, as well as digital words.

The proposed TV tape reader processes incoming data, checking for invalid characters (including parity errors), length of messages (number of characters), which length out of two instructions for positioning characters on the TV screen, and which direction the data is moving through the reader. The same circuit using TTL gates would require over 25 chips. An FPLA instead allows up to 42 characters to be detected for validation and controls, with only six product terms dedicated to implement the fold-back flip-flops.

FPLA I/O ASSIGNMENT SHOWING FOLD-BACK CONNECTIONS FOR REALIZING F/F_1 AND F/F_2 CONTROL FLIP-FLOPS

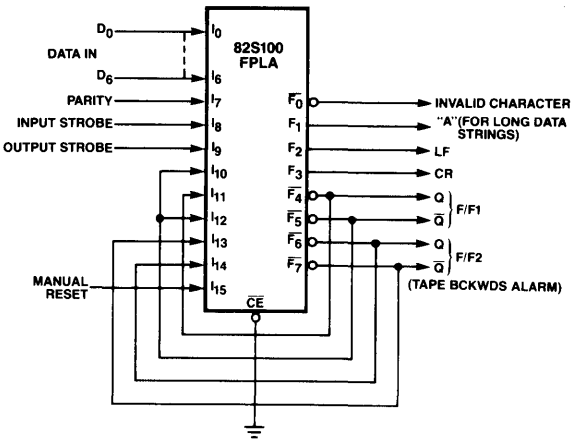


Figure 2.

The input selection property of FPLAs can be combined with simple Set/Reset flip-flops to implement basic logic modules which can be cascaded as shown in Figure 1 to detect group of words in a string of data, such as used by data terminals and other computer controlled devices.

BLOCK DIAGRAM OF MULTISTAGE DIGITAL MESSAGE DECODER

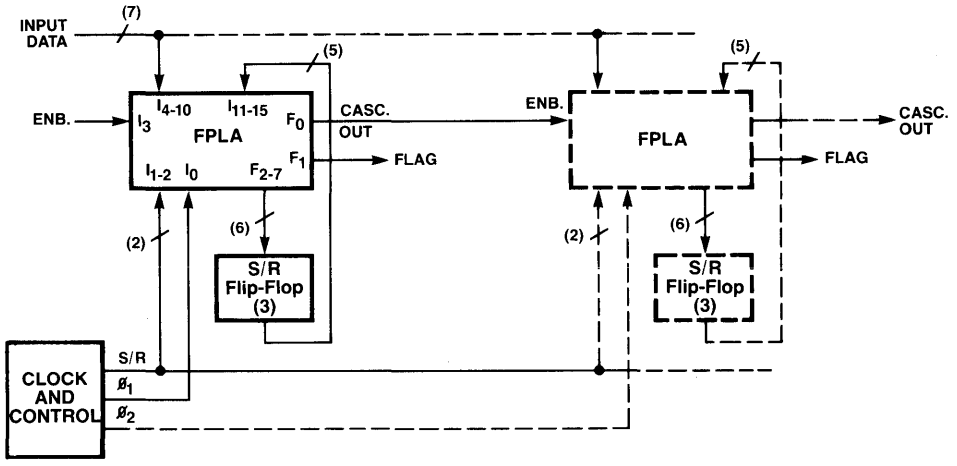


Figure 1.

A typical module, shown in Figure 2, contains 3 stages comprised of an FPLA controlling the setting and resetting of inputs of 3 flip-flops, after proper decoding of the incoming data. Additional circuitry common to all stages/modules generates all necessary clocking signals.

The FPLA is programmed such that product term P_0 contains inputs Ø1, CLK SET, ENABLE (or CASC OUT) from the previous stage, and the True or Complement inputs of the data bus corresponding to the word being decoded. When all these conditions are met, Q1 is set. But if the received word is incorrect, P_1 instead receives as inputs P_0, CLK RESET, and Ø1. This will cause Q1 to be reset anytime an incorrect word is clocked into the module.

To keep the data True signal from the previous stage from being lost while reading the current word, two clock phases are used. While the current state is in use, the previous one is disabled. And since clock Ø2 is a simple inversion of Ø1, only one FPLA input is needed for both.

TYPICAL MODULE OF DIGITAL MESSAGE DECODER
DASH CONNECTIONS RUN *OUTSIDE* FPLA

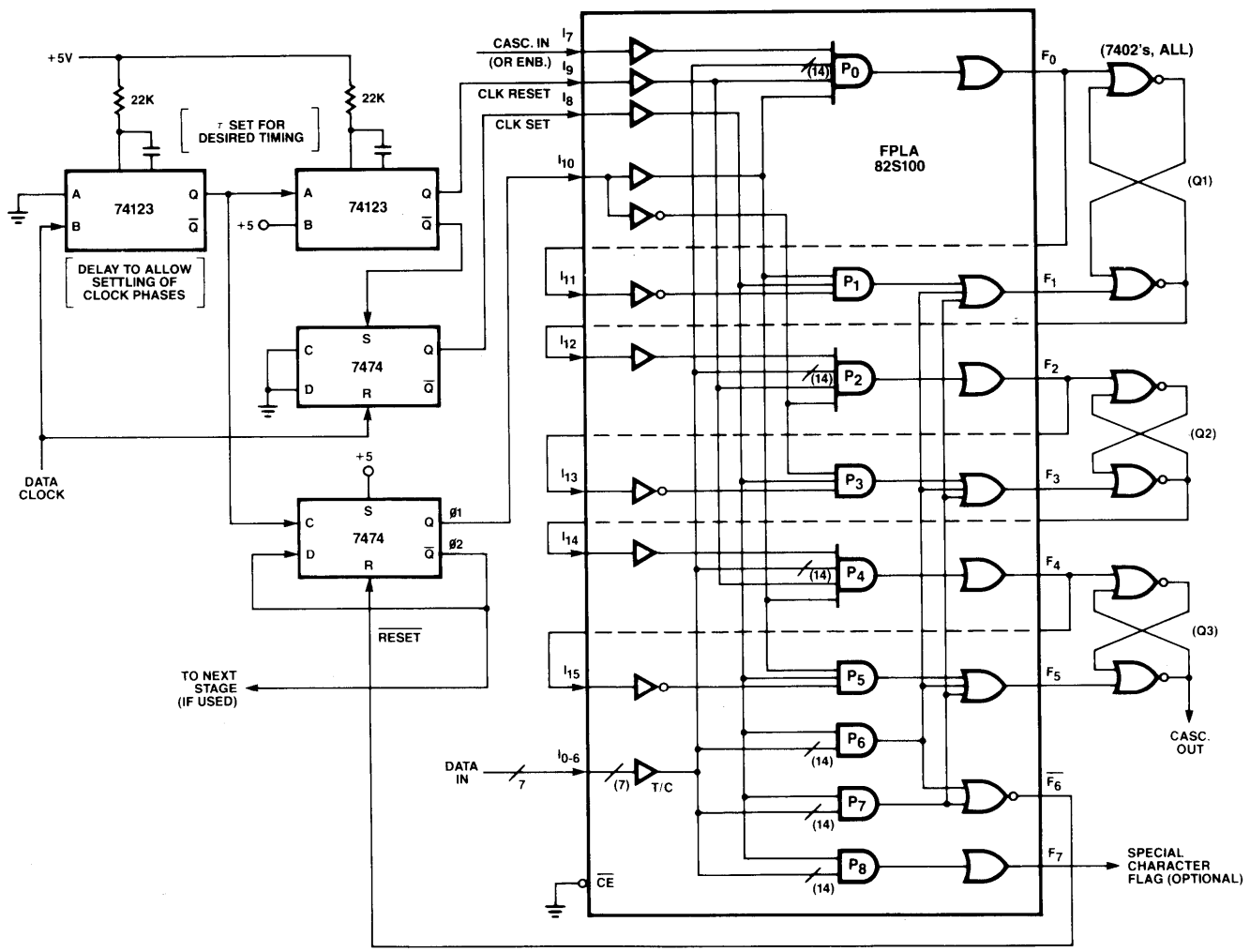


Figure 2.

Note that both phases are generated because if an additional module is used, its first clock phase will be number 2. This way if the circuit is used as an electronic combination lock, it can be changed partly by interchanging FPLAs. That is, each FPLA contains a three word code that can be moved to a different location. As additional security precaution, one person could hold the key, while only another is entrusted with the code. There could even by two keys, both simultaneously needed.

To appreciate the difficulty of breaking such a code to gain access of a system so equipped, note that with two PLA chips it would take over a century to check all the combinations, a rate of one thousand times a second. With four, the time is about 5×10^{11} years.

TIMING DIAGRAM OF MESSAGE DECODER CIRCUIT

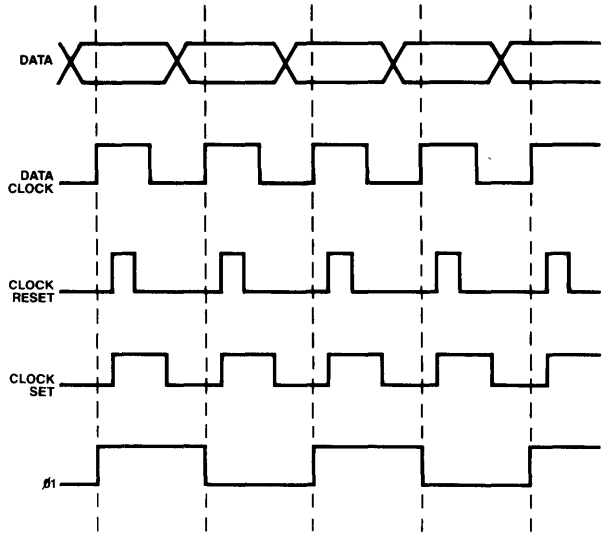


Figure 3.

An FPLA can provide an efficient means to interface a calculator chip to other circuts. The calculator can then be used as a terminal, a part of an instrument, or to drive a printer or CRT. The circuit also allows the calculator to be remotely operated. The interface is complete as shown in Figure 1. Timing is derived from the BCD Digit Output. The control and flow of data in and out is described below and by the calculator specification (National MM3738). The actual implementations of the equipment connected to the interface will depend on the desired application. The timing is derived from the Digit select outputs. The first 10 terms in the FPLA may be used alone as a 7 segment to BCD decoder.

The interface contains signals allocated to the FPLA inputs and outputs as follows:

A) Control – $I_{15\text{-}16}$, where:

I_{15}	I_{16}	Function
0	0	Read data from the calculator
0	1	Input numbers 0 - 9, ., %
1	0	Input functions +, –, etc.
1	1	Request D.P. location

B) Data out – $F_{0\text{-}3}$, F_7, where:

1. With interface control at "00", read multiplexed data in BCD form F_0 (A) to F_3 (D). Digit location corresponds to the BCD digit output. Also, F_7 is a "0" for output 1-9, and becomes a "1" when output 0 is displayed. Two other codes exist when F_7 is a "1":

DCB A	Code
111 0	Negative Sign
111 1	Error Condition

2. With interface control at "11", and the D.P. line gated to I_8 (Data input D), the DCP location will be output in BCD corresponding to the digit location of the decimal point. Two word cycles of the calculator output are required to get both data and decimal point location into a buffer or memory.

SYSTEM BLOCK DIAGRAM OF CALCULATOR TO INSTRUMENT INTERFACE

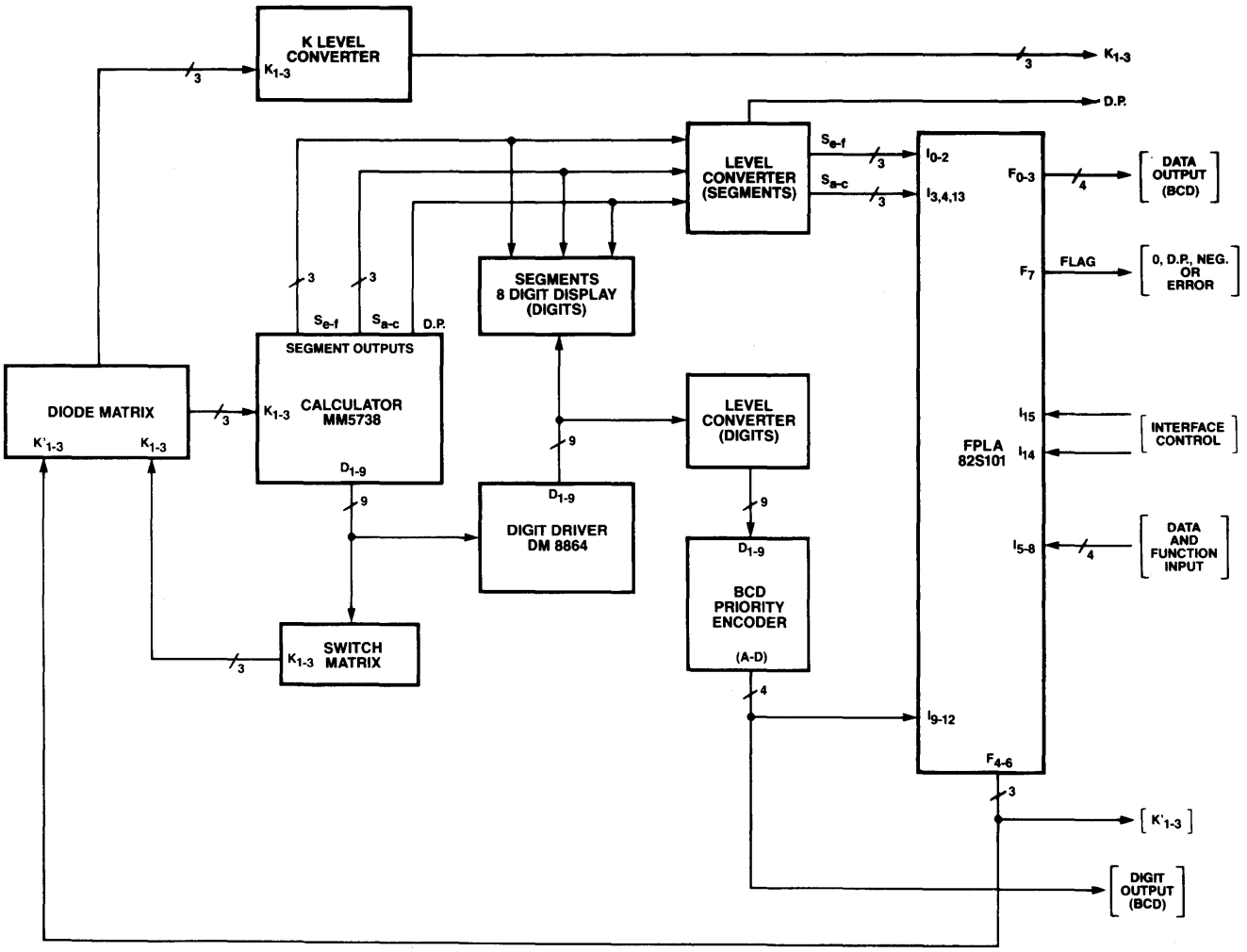


Figure 1.

C) Data Input – With the interface control at "01", data is clocked in at I_5, I_6, I_7, and I_8 as follows:

I_8	I_7	I_6	I_5	Data
0	0	0	0	"0"
0	0	0	1	"1"
0	0	1	0	"2"
0	0	1	1	"3"
0	1	0	0	"4"
0	1	0	1	"5"
0	1	1	0	"6"
0	1	1	1	"7"
1	0	0	0	"8"
1	0	0	1	"9"
1	0	1	0	"."
1	0	1	1	% function

D) Function Input – With the interface control at "10", functions are input as follows:

I_8	I_7	I_6	I_5	Function
0	0	0	1	C
0	0	1	0	K
0	0	1	1	–
0	1	0	0	+
0	1	0	1	MR
0	1	1	0	MS
0	1	1	1	X
1	0	0	0	÷
1	0	0	1	=

FPLA PROGRAM TABLE
Di (I_8) IS ALSO USED FOR D.P. LOCATION

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H H H H H H H H — OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENT	
0	L	L	L	–	–	–	–	–	–	–	–	H	L	L	L	L	A	•	•	•	•	•	•	•	0	OUTPUT DATA
1	L	L	L	–	–	–	–	–	–	–	–	H	H	H	L	H	•	•	•	•	•	•	•	A	1	
2	L	L	H	–	–	–	–	–	–	–	–	L	H	L	L	L	•	•	•	•	•	•	A	•	2	
3	L	L	L	–	–	–	–	–	–	–	–	L	H	H	L	L	•	•	•	•	•	•	A	A	3	
4	L	L	L	–	–	–	–	–	–	–	–	L	L	H	L	H	•	•	•	•	•	A	•	•	4	
5	L	L	L	–	–	–	–	–	–	–	–	L	L	H	H	L	•	•	•	•	•	A	•	A	5	
6	L	L	L	–	–	–	–	–	–	–	–	L	L	L	H	–	•	•	•	•	•	A	A	•	6	
7	L	L	L	–	–	–	–	–	–	–	–	H	–	H	L	L	•	•	•	•	•	A	A	A	7	
8	L	L	L	–	–	–	–	–	–	–	–	L	L	L	L	L	•	•	•	•	A	•	•	•	8	
9	L	L	L	–	–	–	–	–	–	–	–	L	L	H	L	L	•	•	•	•	A	•	•	A	9	
10	L	H	–	L	L	L	H	L	L	L	L	–	–	–	–	–	•	•	•	A	•	•	•	•	0	INPUT DATA
11	L	H	–	L	L	L	H	L	L	L	H	–	–	–	–	–	•	•	•	A	•	•	•	A	1	
12	L	H	–	L	L	H	H	L	L	H	L	–	–	–	–	–	•	•	•	A	•	•	A	•	2	
13	L	H	–	L	H	L	L	L	L	H	H	–	–	–	–	–	•	•	•	A	•	•	A	A	3	
14	L	H	–	L	H	L	H	L	H	L	L	–	–	–	–	–	•	•	•	A	•	A	•	•	4	
15	L	H	–	L	H	H	L	L	H	L	H	–	–	–	–	–	•	•	•	A	•	A	•	A	5	
16	L	H	–	L	H	H	H	L	H	H	L	–	–	–	–	–	•	•	•	A	•	A	A	•	6	
17	L	H	–	H	L	L	L	L	H	H	H	–	–	–	–	–	•	•	•	A	•	A	A	A	7	
18	L	H	–	H	L	L	H	H	L	L	L	–	–	–	–	–	•	•	•	A	A	•	•	•	8	
19	L	H	–	L	L	H	L	H	L	L	H	–	–	–	–	–	•	•	A	•	A	•	•	A	9	
20	L	H	–	L	L	H	H	H	L	H	L	–	–	–	–	–	•	•	A	•	A	•	A	•	•	
21	L	H	–	L	H	L	L	H	L	H	H	–	–	–	–	–	•	•	A	•	A	•	A	A	%	
22	H	L	–	L	L	L	H	L	L	L	H	–	–	–	–	–	•	A	•	•	•	•	•	A	C	
23	H	L	–	L	L	H	L	L	L	H	L	–	–	–	–	–	•	A	•	•	•	•	A	•	K	
24	H	L	–	L	L	H	H	L	L	H	H	–	–	–	–	–	•	A	•	•	•	•	A	A	–	
25	H	L	–	L	H	L	L	L	H	L	L	–	–	–	–	–	•	A	•	•	•	A	•	•	+	
26	H	L	–	L	H	L	H	L	H	L	H	–	–	–	–	–	•	A	•	•	•	A	•	A	MR	
27	H	L	–	L	H	H	L	L	H	H	L	–	–	–	–	–	•	A	•	•	•	A	A	•	MS	
28	H	L	–	L	H	H	H	L	H	H	H	–	–	–	–	–	•	A	•	•	•	A	A	A	X	
29	H	L	–	H	L	L	L	H	L	L	L	–	–	–	–	–	•	A	•	•	A	•	•	•	÷	
30	H	L	–	H	L	L	H	H	L	L	H	–	–	–	–	–	•	A	•	•	A	•	•	A	=	
31	H	H	–	L	L	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A	1	D.P. LOCATION
32	H	H	–	L	L	H	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•	2	
33	H	H	–	L	L	H	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	A	3	
34	H	H	–	L	H	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•	4	
35	H	H	–	L	H	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	A	5	
36	H	H	–	L	H	H	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	•	6	
37	H	H	–	L	H	H	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	A	7	
38	H	H	–	H	L	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•	8	
39	H	H	–	H	L	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	A	9	
40	L	L	H	–	–	–	–	–	–	–	–	L	H	H	H	H	A	•	•	•	A	A	A	•	OUTPUT ERROR	
41	L	L	H	–	–	–	–	–	–	–	–	L	L	L	H	L	A	•	•	•	A	A	A	A		
I/O ASSIGNMENT	CONTROL	SC	D	C	B	A	Di	Ci	Bi	Ai	Sg	Sf	Se	Sb	Sa	FLAG	K_3	K_2	K_1	D	C	B	A		

Figure 2.

Note that when the Functions or Data are input to the calculator, they are read out on the data lines for comparison (if desired). This feature is different from the Data Output by the presence of a "1" in either K_1, K_2, or K_3.

E) Digit Outputs (not an output of the FPLA) – These lines are the digit select outputs of the calculator encoded in BCD. They are used primarily when storing Output Data and Decimal Point location in memory.

F) K_1, K_2, K_3 – These outputs at F_4, F_5 and F_6 are used to input data to the calculator. The calculator recognizes data on the basis of a K input and the Digit Select cycle. The FPLA will activate the proper K line in synchronism with the digit select lines. The K_1, K_2, and K_3 lines as shown on the diagram will indicate switch (keyboard) operation as well as remote data input. The K lines may be used to discriminate between data input and data output functions at the interface.

G) Digit Select inputs to FPLA, I_9, I_{10}, I_{11}, and I_{12} (A' to D' respectively) – These lines are used to select the proper K output when inputting data or functions.

H) Segment inputs to FPLA (I_0 to I_4) These are S_a, S_b, S_e, S_f, S_g respectively, and uniquely identify the BCD outputs of 0 to 9 and the (–) sign. The error condition requires segment S_c input to I_{14} for a unique identification.

The FPLA Program Table containing all codes necessary to service the above interface lines is shown in Figure 2.

MICROPROCESSOR TO CALCULATOR-CHIP INTERFACE 49

The circuit shown in Figure 1 will interface directly between a microprocessor and a calculator chip in systems where low cost, high computational power, but no need for a high speed computation ability (i.e., FFTs and R.T. systems) exist. It could be used in low cost, low speed navigational systems and business applications. FPLA U1 acts as a keyboard simulator receiving output port information from the microprocessor via the data bus, and converting a binary code into the correct keyboard entry to the calculator. The MP will know when to output by interrogating Input Port 1 through FPLA U2 to ascertain if the matrix it wishes to talk to is available at that time. Since the calculator will run in the kHz range and the MP in the MHz range, no problem should be encountered in interfacing.

The U1, U2 combination provide the calculator with the serial inputs normally provided by push buttons. U3 and U4 provide the microprocessor with the answer when the computation is completed.

FPLA-U3 would decode from 8 strobe lines and 7 segment signals an appropriate binary code on to the data bus, thereby providing the microprocessor with answers and evidence of completion of a specific task. It also could provide a method of acknowledgement of receipt of numbers or tasks. FPLA-U4 decodes the upper eight bits of numbers. Since FPLAs are field programmable the system can be adapted to different calculator configurations and efficient software configurations for the microprocessor chosen.

CALCULATOR CHIP TO μP INTERFACE

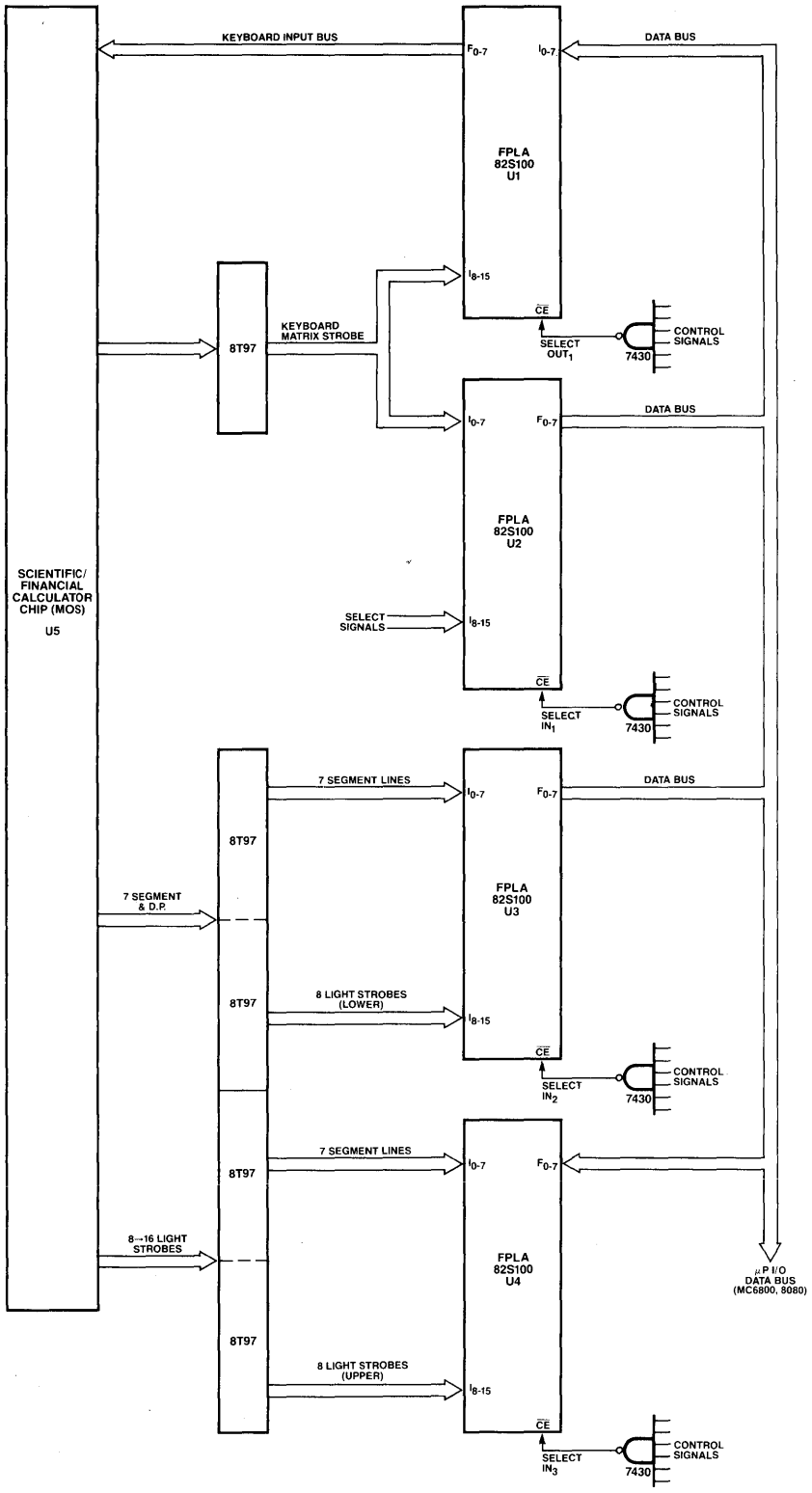


Figure 1.

The circuit shown in Figure 1 uses two Signetics' 82S100 FPLAs, and a few discrete components, to implement a full 64 character ASCII keyboard compact enough to be hand-held or built into the front panel of a computer. Its compactness is due to the use of only 23 keys: 16 character keys, 4 mode keys, and 3 special purpose keys. The keyboard switches need only be SPST-NO; the transistor is not critical, and the LEDs should be efficient enough to be visible with 6 mA current.

The circuit output is a positive logic 7-bit ASCII code, with data valid on the rising edge of the positive-going strobe pulse. Approximately 2 ms of strobe delay is provided for key contact debouncing.

Each of the 16 character keys generates 1 of 4 different characters, depending upon which of four modes the keyboard is in, as selected by the mode keys. When a mode is selected, the keyboard stays in that mode for as many characters as needed until another mode is selected. Included is protection against false character codes due to two keys being depressed at once, key contact debouncing, and 2-key rollover to speed data entry. Also provided are three single purpose keys for carriage return, line feed, and rubout. All characters and functions are supplied by the two FPLAs, programmed as shown in the program tables in Figures 2 and 3, with just 17 Product Terms used in each FPLA. Partially used or partially functional FPLAs may also be used, as long as there are 17 useable Product Terms, and all the 8 outputs operate. The decoding scheme is straightforward. The 16 character keys are divided up into two 8-member groups, and within each group the individual keys are encoded down to the least significant 3 bits of the output. Bit 3 (the fourth bit) is "0" for the group (1) keys and "1" for the group (2) keys. Bits 4 and 5 are set by which of the four modes the keyboard is in. Bit 6 is simply the inverse of bit 5 except when one of the special keys (CR, LF, or RO) is depressed. The remaining of the 17 Product Terms are used to generate the strobe pulse and implement the 2-key rollover feature with the false code protection.

ASCII KEYBOARD ENCODE LOGIC

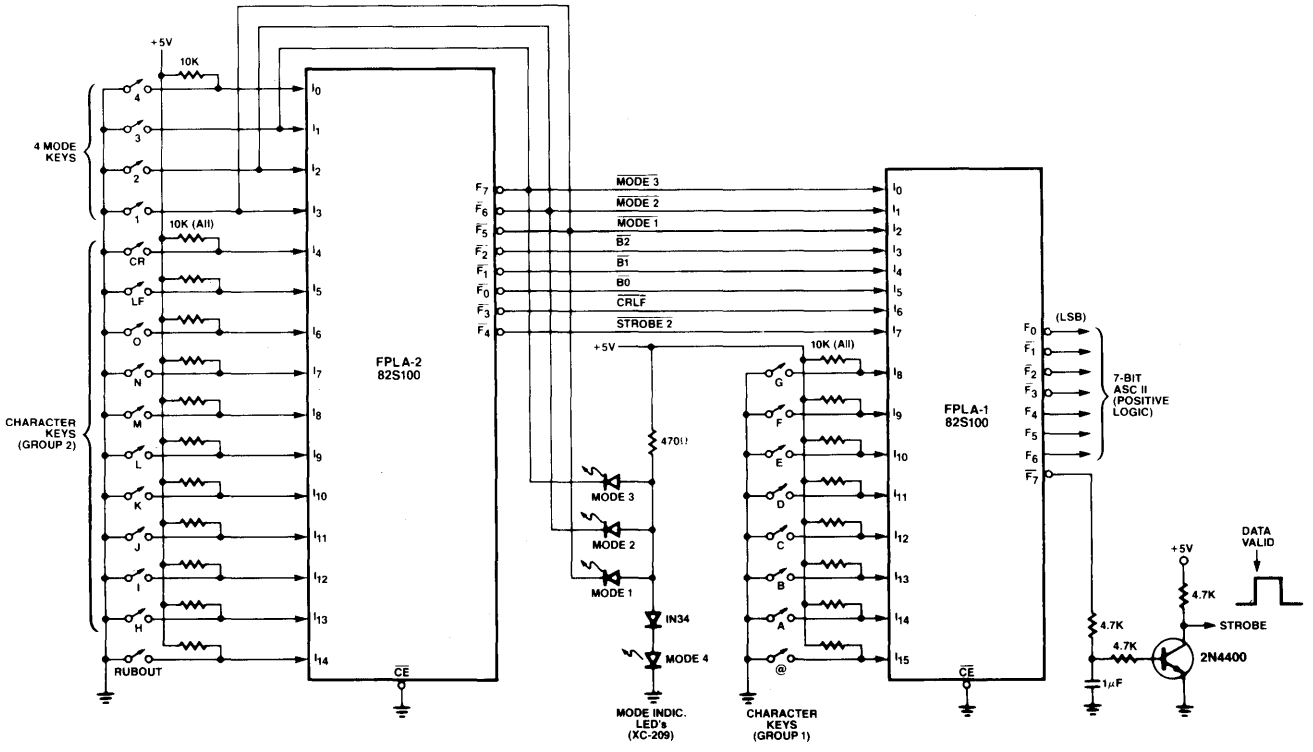


Figure 1.

PROGRAM TABLE FOR FPLA-1

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																L	H	H	H	L	L	L	L
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	H	H	H	H	–	–	–	–	H	–	–	–	•	•	•	•	•	A	•	•
1	–	–	H	H	–	–	H	H	–	–	–	H	–	–	–	–	•	•	•	•	•	•	A	•
2	–	H	–	H	–	H	–	H	–	–	H	–	–	–	–	–	•	•	•	•	•	•	•	A
3	–	–	–	–	–	–	–	–	–	H	–	–	–	–	L	–	•	A	•	•	•	•	•	•
4	–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	•	•	•	•	A	•	•	•
5	–	–	–	–	–	–	–	–	–	H	–	–	–	H	H	–	•	•	A	•	•	•	•	•
6	–	–	–	–	–	–	–	–	–	H	–	–	–	H	–	H	•	•	•	A	•	•	•	•
7	–	–	–	–	–	–	–	–	–	H	–	–	–	L	–	–	•	A	•	•	•	•	•	•
8	L	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
9	H	L	H	H	H	H	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
10	H	H	L	H	H	H	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
11	H	H	L	H	H	H	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
12	H	H	H	H	L	H	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
13	H	H	H	H	H	L	H	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
14	H	H	H	H	H	H	L	H	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
15	H	H	H	H	H	H	H	L	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
16	H	H	H	H	H	H	H	H	L	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•

Figure 2.

PROGRAM TABLE FOR FPLA-2

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																L	L	L	L	L	L	L	L
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	H	–	–	–	–	H	H	H	H	–	H	–	–	–	–	•	•	•	•	•	A	•	•
1	–	H	–	–	H	H	–	–	H	H	H	–	–	–	–	–	•	•	•	•	•	•	A	•
2	–	H	–	H	–	H	–	H	–	H	–	H	–	–	–	–	•	•	•	•	•	•	•	A
3	–	L	H	H	H	H	H	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
4	–	H	L	H	H	H	H	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
5	–	H	H	L	H	H	H	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
6	–	H	H	H	L	H	H	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
7	–	H	H	H	H	L	H	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
8	–	H	H	H	H	H	L	H	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
9	–	H	H	H	H	H	H	L	H	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
10	–	H	H	H	H	H	H	H	L	H	H	H	–	–	–	–	•	•	•	A	•	•	•	•
11	–	H	H	H	H	H	H	H	H	L	H	H	–	–	–	–	•	•	•	A	•	•	•	•
12	–	H	H	H	H	H	H	H	H	H	L	H	–	–	–	–	•	•	•	A	A	•	•	•
13	–	H	H	H	H	H	H	H	H	H	H	L	–	–	–	–	•	•	•	A	A	•	•	•
14	–	H	–	–	–	–	–	–	–	–	–	–	L	H	H	H	•	•	A	•	•	•	•	•
15	–	H	–	–	–	–	–	–	–	–	–	–	H	L	H	H	•	A	•	•	•	•	•	•
16	–	H	–	–	–	–	–	–	–	–	–	–	H	H	L	H	A	•	•	•	•	•	•	•

Figure 3.

In many applications it is desirable to expand the capabilities of inexpensive calculators so that they may be easily interfaced to printers, BCD displays, or computer data input. The output of most calculator chips is automatically multiplexed and presented in standard 7-segment form. For each digit there is a digit select output signal which appears sequentially while segment outputs provide the decoded figure.

CONVERSION TRUTH-TABLE TO BE PROGRAMMED IN THE FPLA

NUMBER	INPUTS: DIGIT SELECT LINE 10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0	INPUTS: SEGMENT LINE a	b	c	d	e	f	g	OUTPUTS: BCD 2^3	2^2	2^1	2^0	DIGIT 2^2	2^1	2^0	STROBE
0	DON'T CARE (X)								1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1									0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
2									1	1	0	1	1	0	1	0	0	1	0	0	0	0	0
3									1	1	1	1	0	0	1	0	0	1	1	0	0	0	0
4									0	1	1	0	0	1	1	0	1	0	0	0	0	0	0
5									1	0	1	1	0	1	1	0	1	0	1	0	0	0	0
6									1	0	1	1	1	1	1	0	1	1	0	0	0	0	0
7									1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
8									1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
9									1	1	1	1	0	1	1	1	0	0	1	0	0	0	0
BLANK									0	0	0	0	0	0	0	(1	1	1	1)*	0	0	0	0
	1	0	0	0	0	0	0	0	DON'T CARE (X)							0	0	0	0	0	0	0	1
	0	1	0	0	0	0	0	0								0	0	0	0	0	0	1	1
	0	0	1	0	0	0	0	0								0	0	0	0	0	1	0	1
	0	0	0	1	0	0	0	0								0	0	0	0	0	1	1	1
	0	0	0	0	1	0	0	0								0	0	0	0	1	0	0	1
	0	0	0	0	0	1	0	0								0	0	0	0	1	0	1	1
	0	0	0	0	0	0	1	0								0	0	0	0	1	1	0	1
	0	0	0	0	0	0	0	1								0	0	0	0	1	1	1	1

* or (0000)

Figure 1.

The circuit in Figure 1 shows how an FPLA can provide a simple means for converting the 7-segment calculator output to Binary Coded Decimal. Further decoding of up to eight calculator digit select lines provides a binary number indicating which digit is being displayed, along with a strobe pulse to indicate when a valid digit occurs. The full conversion truth-table to be programmed in the FPLA is tabulated in Figure 2.

CIRCUIT DIAGRAM

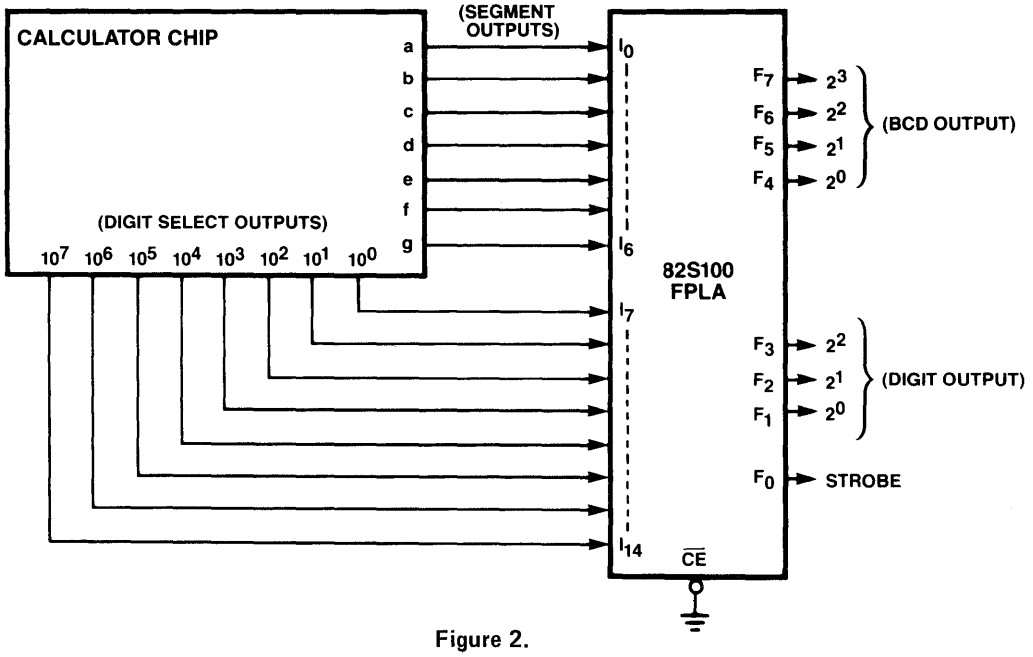


Figure 2.

In wide range and multifunction instruments, the FPLA can be effectively used to provide proper decimal point display and to illuminate function legends. The circuit in Figure 1 shows the application of an FPLA in an autoranging digital multimeter. The FPLA receives inputs from a shift register in the autoranging section. It also accepts inputs from the function select switch. The open collector outputs of the FPLA could directly drive the decimal point within its limit. Other outputs are used to turn on appropriate LEDs displaying legends such as V, MV, R, KΩ, A, MA etc. Buffering may be necessary. The FPLA can accommodate large flexible ranges as well as many functions, thus reducing design time and IC gate packages in a family of instruments.

D.P. AND LEGEND DISPLAY DRIVER

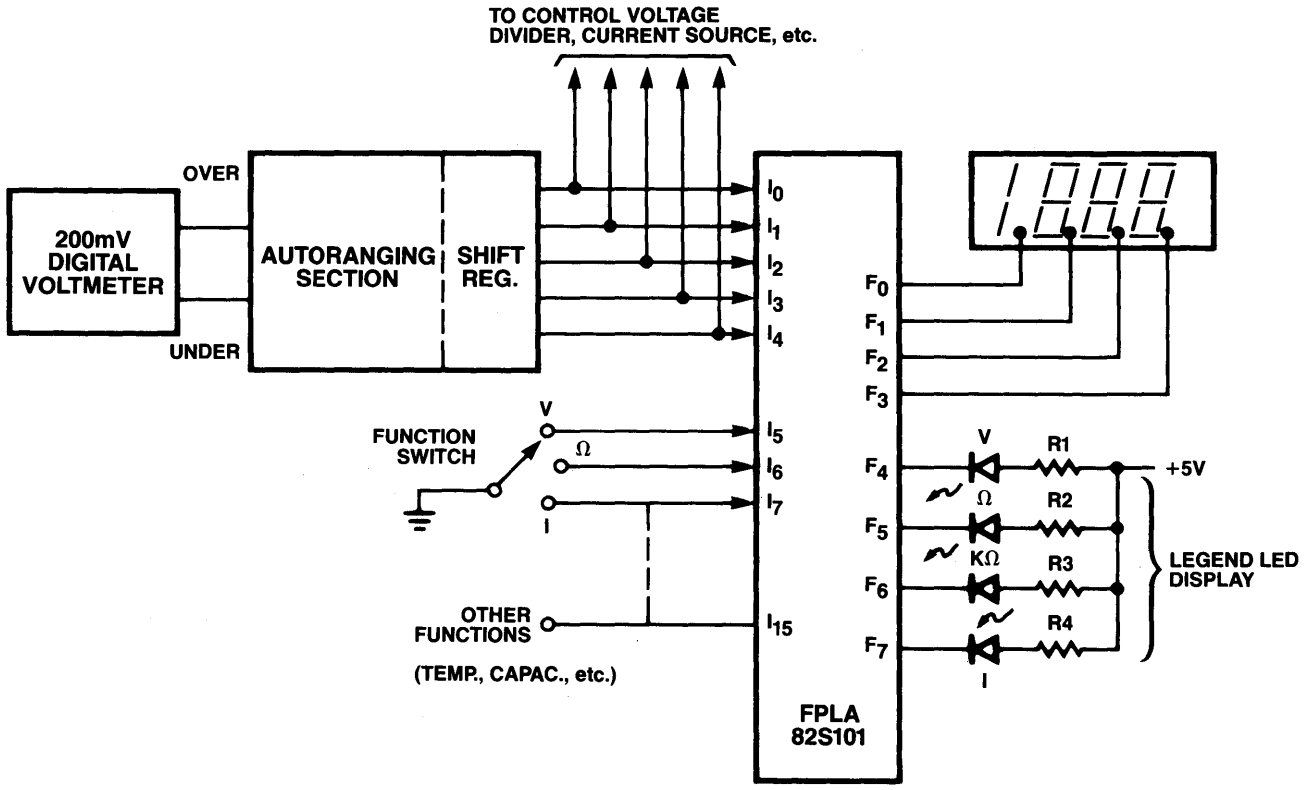


Figure 1.

The circuit diagrammed in Figure 1 is designed to accept a string of hexadecimal characters from a calculator type keyboard and provide a 10 second signal to a user defined actuator such as a solenoid driven door latch, when the character string matches the programmed combination.

Four 16-character combinations may be programmed in the FPLA. The combination to be used is specified by the position of the two combination select switches S_1 and S_2. Combinations of shorter length may be programmed to allow easier entry to the device being locked, but maximum security will be realized using a 16-character combination. Before using, the lock must first be reset in a known state. Depressing the RESET button on the keyboard forces the reset line LOW. The RESET signal is buffered by the FPLA and appears as an active-LOW signal called BRESET, which causes the TRUE flip-flop to be forced true, the OPEN timer to be forced off, and the character counter to be loaded with zero. The trial combination is then entered by depressing the numerical keys in sequence. Upon depressing the last key in a correct sequence a 10 second actuation will occur.

Depressing a numerical key causes a hexadecimal data pattern to appear on the four data lines D_0-D_3. An active-LOW strobe from the keyboard informs the FPLA that a valid data pattern is being presented. The STROBE signal is buffered by the FPLA to become the active-LOW signal BSTROBE. While STROBE is LOW, the data field to the FPLA is compared to the programmed data character asso-

1 TO 16 DIGIT COMBINATION LOCK

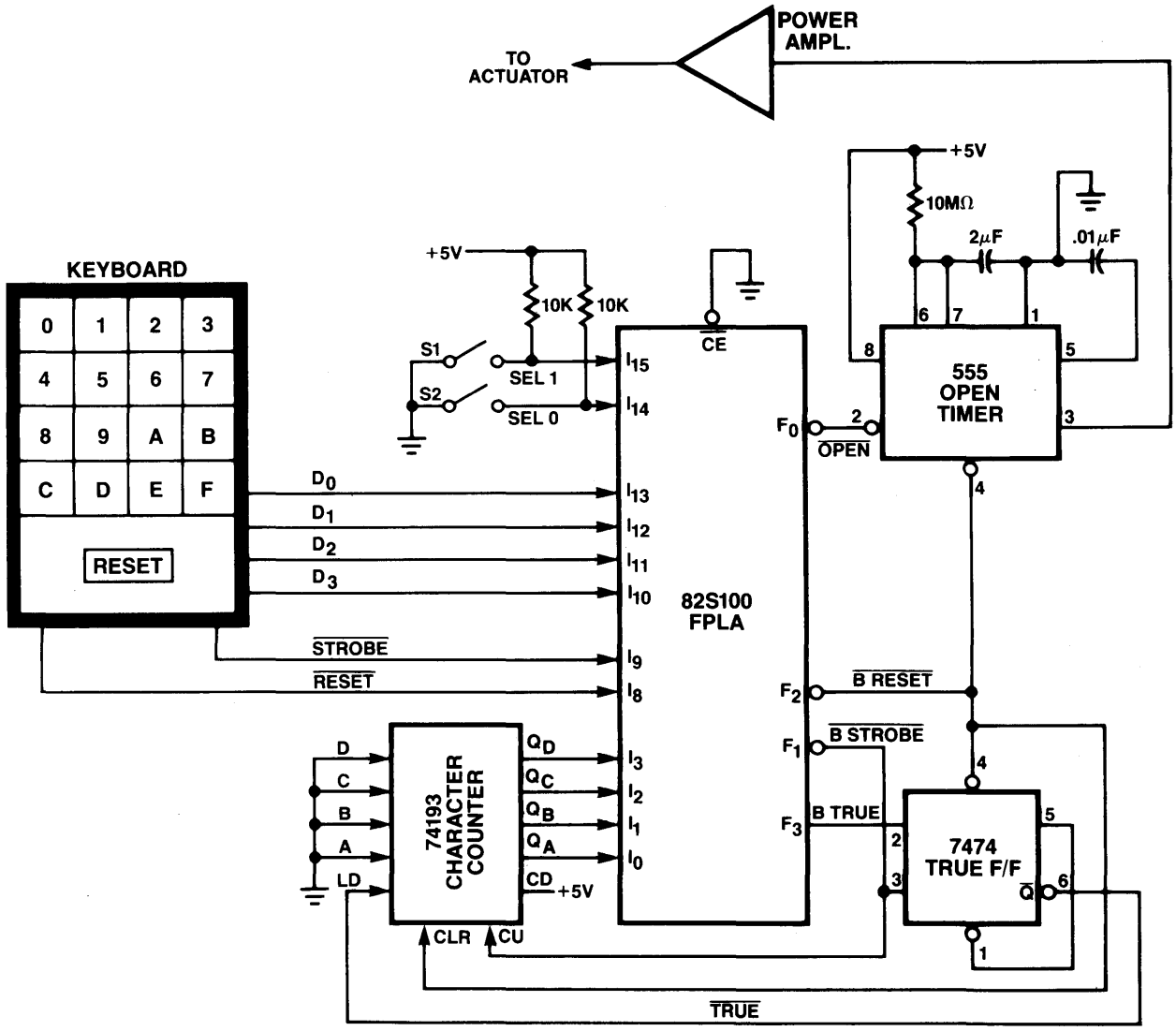


Figure 1.

ciated with the current character count (C_0-C_3), and combination select (SEL_0-SEL_1). If the data matches the programmed character, signal BTRUE will be forced HIGH; if not, it will appear as a LOW. At the rising edge of BSTROBE, the state of BTRUE is latched into the TRUE flip-flop, and the character counter is incremented. If a mismatch had existed between the data and the programmed character, a LOW would be clocked into the TRUE flip-flop, causing it to latch itself in the false state, resetting the character counter and ignoring all future data comparisons until the RESET key is depressed.

The character counter increments for each sequential successful match until a preprogrammed terminal count is reached. The OPEN signal is activated when the terminal count has been reached, STROBE is LOW, and the data field matches the programmed character associated with terminal count and the selected combination. When the OPEN signal goes LOW, it fires a 10 second interval timer. The HIGH output from the running timer is applied to a power amplifier to drive the user defined actuator. For a safe, or door lock application, the actuator could be a solenoid driven latch pin. For automotive applications, the actuator could be an ignition enable relay. For computer applications, this device could provide restricted use of certain peripherals or of the entire computing mechanism.

Additional outputs are available in the FPLA to provide an alarm function. For example, an alarm bit could be programmed to be activated when an excessive number of characters were detected in a string. In safe door lock applications, a battery back up to the normal supply is recommended to prevent permanently sealing the safe upon loss of power to the unit.

DATA SECURITY ENCODER 54

The security of a data train on a common carrier is enhanced when data is encoded to a particular hard-wired encoded version. This results in the correct detection only in a similarly programmed receiving terminal.

The system shown in Figure 1 seems to have well over 8!=(403020) different combinations without resorting to anything but pulse permutations. In applications for computer systems, the computer could use a random sequence to program devices, as only devices which communicate need to know how they are programmed. Data security is assured, since only the programmer knows the permutation in use.

DATA SYSTEM SECURITY ENCODER

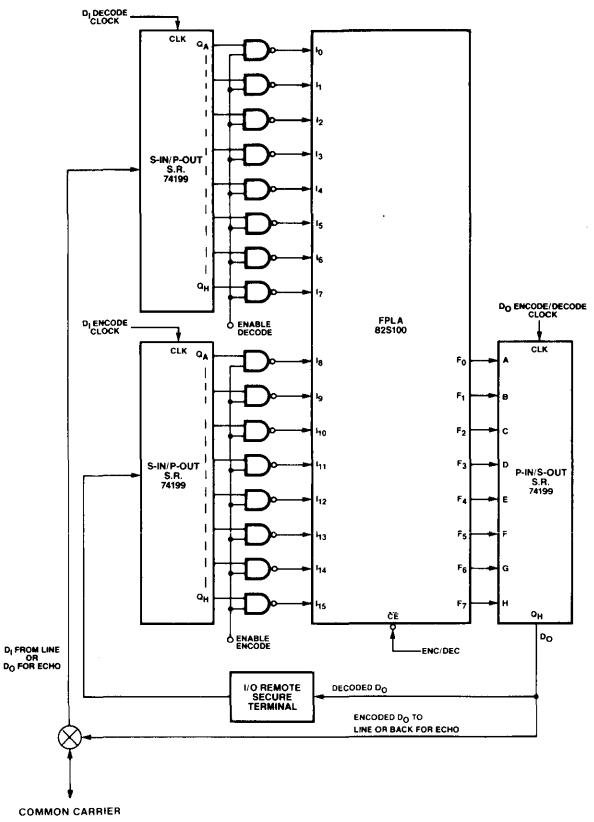


Figure 1.

The calendar clock date reminder decodes 48 unique dates in a calendar year. This would be used to give the owner a reminder of birthdays, anniversaries or any other special day he feels he should remember.

The FPLA is programmed to recognize the dates selected by the owner. Practically, it would be programmed to activate a monitor several days before the actual event, to give the owner time to respond to the reminder.

The circuit in Figure 1 gates the normally multiplexed output of a calendar clock circuit into 11 input lines to the FPLA. To do this, it must demultiplex the BCD output and latch it into registers for a stable input to the FPLA. This is accomplished by the calendar clock's internal digit driver used to gate the appropriate 74175 register. For example, the clock digit driver that is used to display the least significant days digit, D0, is used to latch this data into IC_4 for input into the FPLA as I_0 thru I_3.

When the month and day BCD input match the programmed date, the FPLA outputs F0 - F7 are used to visually indicate the reminder. This can be implemented simply with LEDs driven by the FPLA outputs to indicate various events.

CALENDAR CLOCK DATE REMINDER

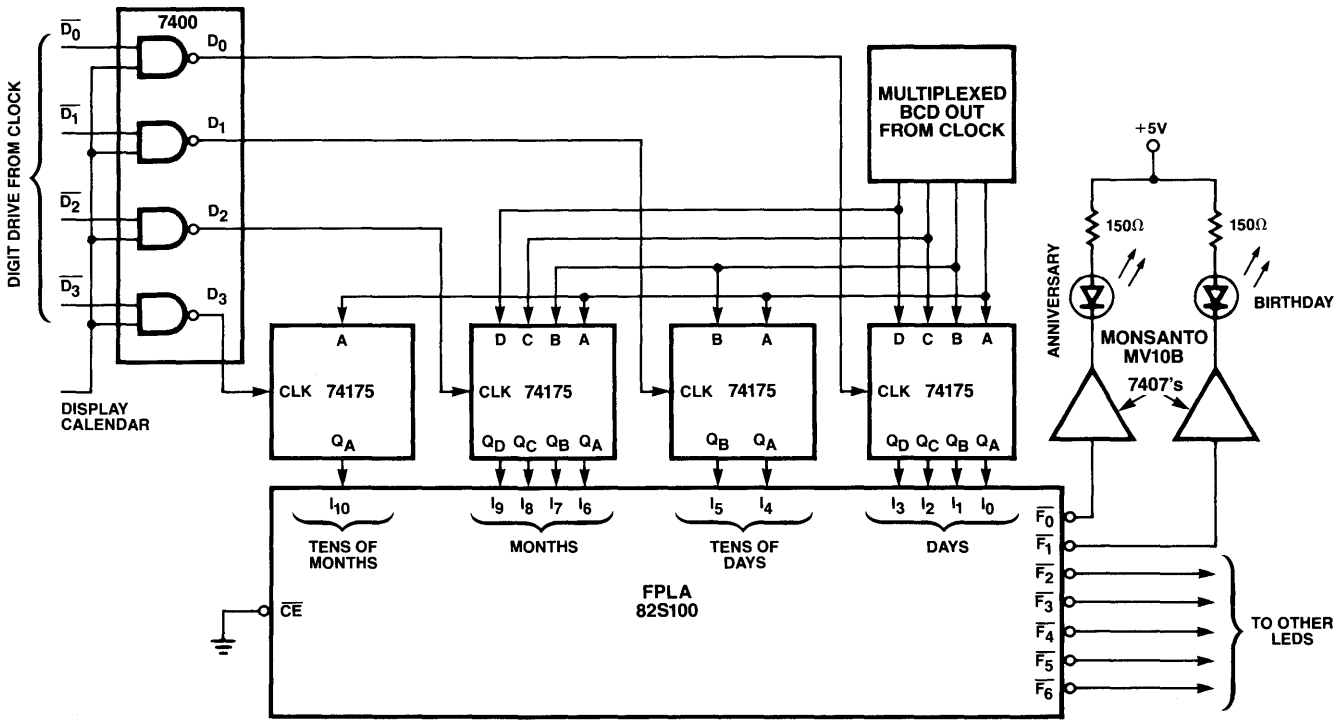


Figure 1.

A preventive Maintenance Monitor can be used to give an accurate visual or audio indication of maintenance required for any type of hardware or equipment at specified time intervals. It could be used with costly machinery that required various maintenance actions at diverse times, without which degradation of equipment would result.

Equipment where such a positive maintenance indication can be utilized range from simple computer hardware such as mag-tape transports and disks, to more complex electro-mechanical machinery and systems such as airplanes, etc.

The circuit shown in Figure 1 utilizes the elapsed time meter usually present on any critical piece of hardware/system. The ETM can be modified, like a thumbwheel switch, to provide BCD outputs to the Preventive Maintenance Monitor.

Although general in principle, the circuit gives an example of a PMM utilized to implement a typical maintenance schedule for an automobile, as recommended by the manufacturer's specifications. The input to the system in this case is the odometer reading. (The odometer has been modified to provide BCD outputs). The outputs of the system are real time indications of which items need maintenance, and when. There are two types of LED indicators: one indicates that some function should be checked, while the other indicates something needs to be replaced or changed.

The inputs to the FPLA are four BCD digits, representing miles in tens, hundreds, thousands, and tens of thousands. Using this scheme, when the odometer value is decoded by the FPLA to indicate a maintenance action, the LED indicators for that action will remain lit for ten miles until the tens BCD changes. By using less inputs the indicators would stay lit longer. For example, 12 inputs, (eliminating tens), would cause the indicators to stay on for a hundred miles, which may be more desirable. Any thing less than 12 inputs (3 BCD's) would make the degree of resolution time when action is required too low.

PREVENTIVE MAINTENANCE MONITOR
(N) FOR A GIVEN F_X DEPENDS ON THE NUMBER OF MAINTENANCE ITEMS DRIVEN BY EACH FPLA OUTPUT. FOR EXAMPLE, F_0 WOULD NEED 6 LED BRANCHES TO DRIVE 1) ENGINE COOLANT CHANGE, 2) PLUG REPLACEMENT 3) CHECK BRAKE LINING AND DRUM, 4) CHANGE BRAKE FLUID, 5) CHECK AND REPLENISH STEERING GEAR BOX, AND 6) REPLACE PCV VALVE

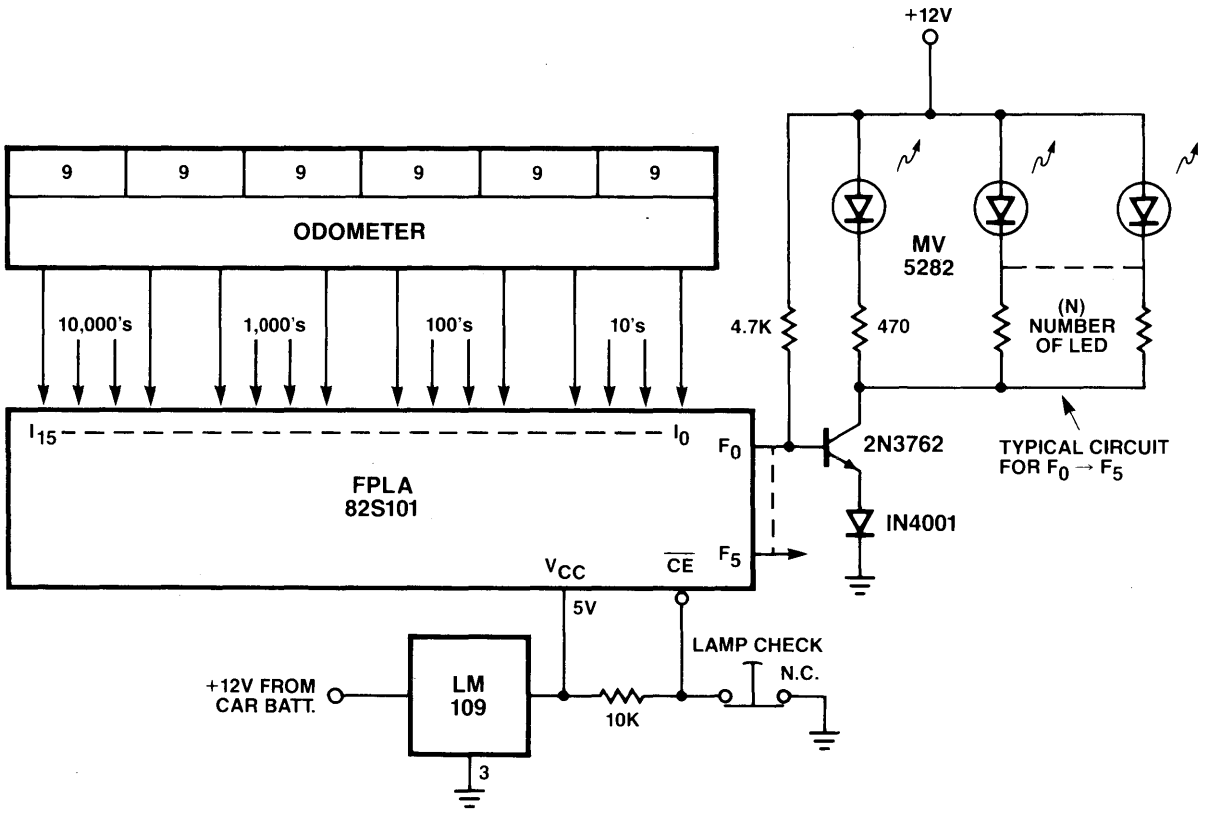


Figure 1.

Just one FPLA easily implements the manufacturer's maintenance schedule outlined in Figure 2. As shown in the Program Table in Figure 3, only 34 product terms are necessary to carry maintenance out to 99,000 miles. After 100,000 miles the PMM system would wraparound and be functional for however long the car lasted. Five outputs (0-4) are used for the four frequency classes of maintenance as shown on the left column of the maintenance schedule.

In addition, F5 is used for indicating replacements or changes needed. F6 and F7 are not used but could be used, along with some of the unused product terms, for some other indication such as a reminder as to when the car should be brought in for its warranty check-ups. The momentary push button switch is used as a lamp-check by forcing all outputs to the default HIGH state via the $\overline{CE}$ input.

TYPICAL AUTOMOBILE MAINTENANCE SCHEDULE
(R) MEANS REPLACE OR CHANGE. F_X ARE FPLA OUTPUTS ACTIVATED AT THE GIVEN MILEAGE INTERVALS

FX ASSIGNMENT	UNIT: 1,000 miles		1	3	6	9	12	15	18	21	24	27	30	33	36	Remarks
0	Change engine coolant						X				X				X	or within 12 months
1	Replace oil filter element		X		X		X		X		X		X		X	
2	Check battery acid level & specific gravity		X	X	X	X	X	X	X	X	X	X	X	X	X	
2	Check & adjust fan and belt tension		X	X	X	X	X	X	X	X	X	X	X	X	X	
2	Check & adjust distributor dwell angle, points & gap		X	X	X	X	X	X	X	X	X	X	X	X	X	
0,1	Check & clean or replace spark plugs		X		X		R		X		R		X		R	
5,2	Clean or replace air cleaner element			X	X	X	X	X	R	X	X	X	X	X	R	
3	Replace fuel filter element										X					
	Tighten nuts & bolts on engine		X													
1	Check & adjust valve clearance		X		X		X		X		X		X		X	
2	Check engine idling speed & carburator condition		X	X	X	X	X	X	X	X	X	X	X	X	X	
1	Check damage of fuel hoses		X		X		X		X		X		X		X	Replace every 48 months
2	Check & adjust ignition timing		X	X	X	X	X	X	X	X	X	X	X	X	X	
4	Check resistive cord resistance (spark plug leads)		X				X				X				X	
	CHASSIS & BODY															
0	Check brake lining & drum						X				X				X	
2	Check brake pad & disc			X	X	X	X	X	X	X	X	X	X	X	X	
1	Check brake booster operation				X		X		X		X		X		X	
2	Check & adjust brake pedal free play & parking brake		X	X	X	X	X	X	X	X	X	X	X	X	X	
1	Check & adjust clutch pedal free play		X		X		X		X		X		X		X	
1	Check steering free play, linkage, front suspension & ball joints		X		X		X		X		X		X		X	
1	Rotate tires				X		X		X		X		X		X	
4	Check front end alignment (side slip)		X				X				X				X	
1	Check leakage of oil, fuel, fluid & water, damage of brake pipes & hoses		X		X		X		X		X		X		X	
4	Tighten nuts & bolts on chassis & body		X				X				X				X	
	LUBRICATION															
2	Change engine oil		X	X	X	X	X	X	X	X	X	X	X	X	X	or within 3 months
0	Change brake fluid (w/disc brake)						X				X				X	
0	Check & replenish steering gear box						X				X				X	
5,1	Check & replenish or change transmission & differential gear oil				X		X		R*		X		X		R*	*or within 24 months
5,2	Check & replenish or change automatic transmission fluid		X	X	X	X	X	X	R*	X	X	X	X	X	R*	*or within 24 months
3	Lubricate ball joints & front wheel bearings										X					or within 24 months
	CRANKCASE & EXHAUST EMISSION CONTROL SYSTEM															
4	Check operation of throttle positioner, speed marker & vacuum switching valve, check hose connections		X				X				X				X	or every 12 months
0	Replace positive crankcase ventilation valve, clean & check connections						X				X				X	or every 12 months

Figure 2.

FPLA PROGRAM TABLE FOR IMPLEMENTING ASSUMED MAINTENANCE SCHEDULE

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL							
																	H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	•
1	H	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
2	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
3	H	H	H	H	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
4	H	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
5	H	H	H	L	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
6	H	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
7	H	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
8	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
9	H	H	L	H	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
10	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
11	H	H	L	L	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
12	H	H	L	L	H	L	L	H	H	H	H	H	H	H	H	H	A	A	A	A	•	A	A	A
13	H	H	L	L	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
14	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
15	H	L	H	H	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
16	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
17	H	L	H	L	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
18	H	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
19	H	L	H	L	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
20	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
21	H	L	L	H	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
22	H	L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
23	H	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
24	H	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	A	A	A	A	A	A	A	A
25	H	L	L	L	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
26	H	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
27	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
28	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
29	L	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
30	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
31	L	H	H	L	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
32	L	H	H	L	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
33	L	H	H	L	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•

Figure 1.

SIGNETICS

HEADQUARTERS

811 East Arques Avenue
Sunnyvale, California 94086
Phone: (408) 739-7700

ARIZONA

Phoenix
Phone: (602) 971-2517

CALIFORNIA

Inglewood
Phone: (213) 670-1101

Irvine
Phone: (714) 833-8980
(213) 924-1668

San Diego
Phone: (714) 560-0242

Sunnyvale
Phone: (408) 736-7565

COLORADO

Parker
Phone: (303) 841-3274

FLORIDA

Pompano Beach
Phone: (305) 782-8225

ILLINOIS

Rolling Meadows
Phone: (312) 259-8300

INDIANA

Noblesville
Phone (317) 773-6770

KANSAS

Wichita
Phone: (316) 683-6035

MASSACHUSETTS

Woburn
Phone: (617) 933-8450

MICHIGAN

Southfield
Phone: (313) 645-1232

MINNESOTA

Edina
Phone: (612) 835-7455

NEW JERSEY

Cherry Hill
Phone: (609) 665-5071

Piscataway
Phone: (201) 981-0123

NEW YORK

Wappingers Falls
Phone: (914) 297-4074

Woodbury, L.I.
Phone: (516) 364-9100

OHIO

Worthington
Phone: (614) 888-7143

TEXAS

Dallas
Phone: (214) 661-1296

REPRESENTATIVES

CALIFORNIA

San Diego
Mesa Engineering
Phone: (714) 278-8021

Sherman Oaks
Astralonics
Phone: (213) 990-5903

CANADA

Calgary, Alberta
Philips Electronics Industries Ltd.
Phone: (403) 243-7737

Montreal, Quebec
Philips Electronics Industries Ltd.
Phone: (514) 342-9180

Ottawa, Ontario
Phillips Electronics Industries Ltd.
Phone: (613) 237-3131

Scarborough, Ontario
Philips Electronics Industries Ltd.
Phone: (416) 292-5161

Vancouver, B.C.
Philips Electronics Industries Ltd.
Phone: (604) 435-4411

COLORADO

Denver
Barnhill Five, Inc.
Phone: (303) 426-0222

CONNECTICUT

Newtown
Kanan Associates
Phone: (203) 426-8157

FLORIDA

Altamonte Springs
Semtronic Associates
Phone: (305) 831-8233

Largo
Semtronic Associates
Phone: (813) 586-1404

ILLINOIS

Chicago
L-Tec Inc.
Phone: (312) 286-1500

KANSAS

Lenexa
Buckman & Associates
Phone: (913) 492-8470

MARYLAND

Glen Burni
Microcomp, Inc.
Phone: (301) 761-4600

MASSACHUSETTS

Reading
Kanan Associates
Phone: (617) 944-8484

MICHIGAN

Bloomfield Hills
Enco Marketing
Phone: (313) 642-0203

MINNESOTA

Edina
Mel Foster Tech. Assoc.
Phone: (612) 835-2254

MISSOURI

St. Charles
Buckman & Associates
Phone: (314) 724-6690

NEW JERSEY

Haddonfield
Thomas Assoc. Inc.
Phone: (609) 854-3011

NEW MEXICO

Albuquerque
The Staley Company, Inc.
Phone: (505) 821-4310/11

NEW YORK

Ithaca
Bob Dean, Inc.
Phone: (607) 272-2187

NORTH CAROLINA

Cary
Montgomery Marketing
Phone: (919) 467-6319

OHIO

Centerville
Norm Case Associates
Phone: (513) 433-0966

Fairview Park
Norm Case Associates
Phone: (216) 333-4120

OREGON

Portland
Western Technical Sales
Phone: (503) 297-1711

TEXAS

Dallas
Cunningham Co.
Phone: (214) 233-4303

Houston
Cunningham Company
Phone: (713) 461-4197

UTAH

West Bountiful
Barnhill Five, Inc.
Phone:(801) 292-8991

WASHINGTON

Bellevue
Western Technical Sales
Phone: (206) 641-3900

WISCONSIN

Greenfield
L-Tec, Inc.
Phone: (414) 545-8900

DISTRIBUTORS

ALABAMA

Huntsville
Hamilton/Avnet Electronics
Phone: (205) 533-1170

ARIZONA

Phoenix
Hamilton/Avnet Electronics
Phone: (602) 275-7851

Liberty Electronics
Phone: (602) 257-1272

CALIFORNIA

Costa Mesa
Avnet Electronics
Phone: (714) 754-6111

Schweber Electronics
Phone: (213) 537-4320

Culver City
Hamilton Electro Sales
Phone:(213) 558-2173

El Segundo
Liberty Electronics
Phone: (213) 322-8100

Mountain View
Elmar Electronics
Phone: (415) 961-3611

Hamilton/Avnet Electronics
Phone: (415) 961-7000

San Diego
Hamilton/Avnet Electronics
Phone: (714) 279-2421

Liberty Electronics
Phone: (714) 565-9171

Sunnyvale
Intermark Electronics
Phone: (408) 738-1111

CANADA

Downsview, Ontario
Cesco Electronics
Phone: (416) 661-0220

Mississauga, Ontario
Hamilton/Avnet Electronics
Phone: (416) 677-7432

Montreal, Quebec
Cesco Electronics
Phone: (514) 735-5511

Zentronics Ltd.
Phone: (514) 735-5361

Ottawa, Ontario
Hamilton/Avnet Electronics
Phone: (613) 226-1700

Zentronics Ltd.
Phone: (613) 238-6411

Toronto, Ontario
Zentronics Ltd.
Phone: (416) 789-5111

Vancouver, B.C.
Bowtek Electronics Co., Ltd.
Phone: (604) 736-1141

Ville St. Laurent, Quebec
Hamilton/Avnet Electronics
Phone: (514) 331-6443

COLORADO

Commerce City
Elmar Electronics
Phone: (303) 287-9611

Denver
Hamilton/Avnet Electronics
Phone: (303) 534-1212

Lakewood
Acacia Sales, Inc.
Phone: (303) 232-2882

CONNECTICUT

Danbury
Schweber Electronics
Phone: (203) 792-3500

Georgetown
Hamilton/Avnet Electronics
Phone: (203) 762-0361

Hamden
Arrow Electronics
Phone: (203) 248-3801

FLORIDA

Ft. Lauderdale
Arrow Electronics
Phone: (305) 776-7790

Hamilton/Avnet Electronics
Phone: (305) 971-2900

Hollywood
Schweber Electronics
Phone: (305) 922-4506

Orlando
Hammond Electronics
Phone: (305) 241-6601

GEORGIA

Atlanta
Schweber Electronics
Phone: (404) 449-9170

Norcross
Hamilton/Avnet Electronics
Phone: (404) 448-0800

ILLINOIS

Elk Grove
Schweber Electronics
Phone: (312) 593-2740

Elmhurst
Semiconductor Specialists
Phone: (312) 279-1000

Schiller Park
Hamilton/Avnet Electronics
Phone: (312) 671-6082

INDIANA

Indianapolis
Semiconductor Specialists
Phone: (317) 243-8271

KANSAS

Lenexa
Hamilton/Avnet Electronics
Phone: (913) 888-8900

MARYLAND

Baltimore
Arrow Electronics
Phone: (301) 247-5200

Gaithersburg
Pioneer Washington Electronics
Phone: (301) 948-0710

Hanover
Hamilton/Avnet Electronics
Phone: (301) 796-5000

Rockville
Schweber Electronics
Phone: (301) 881-2970

MASSACHUSETTS

Waltham
Schweber Electronics
Phone: (617) 890-8484

Woburn
Arrow Electronics
Phone: (617) 933-8130

Hamilton/Avnet Electronics
Phone: (617) 933-8000

MICHIGAN

Livonia
Hamilton/Avnet Electronics
Phone: (313) 522-4700

Troy
Schweber Electronics
Phone: (313) 583-9242

MINNESOTA

Eden Prairie
Schweber Electronics
Phone: (612) 941-5280

Edina
Hamilton/Avnet Electronics
Phone: (612) 941-3801

Minneapolis
Semiconductor Specialists
Phone: (612) 854-8841

MISSOURI

Hazelwood
Hamilton/Avnet Electronics
Phone: (314) 731-1144

NEW MEXICO

Albuquerque
Hamilton/Avnet Electronics
Phone: (505) 765-1500

NEW YORK

Buffalo
Summit Distributors
Phone: (716) 884-3450

East Syracuse
Hamilton/Avnet Electronics
Phone: (315) 437-2642

Farmingdale, L.I.
Arrow Electronics
Phone: (516) 694-6800

Rochester
Hamilton/Avnet Electronics
Phone: (716) 442-7820

Schweber Electronics
Phone: (716) 461-4000

Westbury, L.I.
Hamilton/Avnet Electronics
Phone: (516) 333-5800

Schweber Electronics
Phone: (516) 334-7474

NORTHERN NEW JERSEY

Cedar Grove
Hamilton/Avnet Electronics
Phone: (201) 239-0800

Saddlebrook
Arrow Electronics
Phone: (201) 797-5800

SOUTHERN NEW JERSEY AND PENNSYLVANIA

Cherry Hill, N.J.
Milgray-Delaware Valley
Phone: (609) 424-1300

Moorestown, N.J.
Arrow/Angus Electronics
Phone: (609) 235-1900

Mt. Laurel, N.J.
Hamilton/Avnet Electronics
Phone: (609) 234-2133

CENTRAL NEW JERSEY AND PENNSYLVANIA

Somerset, N.J.
Schweber Electronics
Phone: (201) 469-6008

Horsham, PA
Schweber Electronics
Phone: (215) 441-0600

NORTH CAROLINA

Greensboro
Hammond Electronics
Phone: (919) 275-6391

OHIO

Beechwood
Schweber Electronics
Phone: (216) 464-2970

Cleveland
Arrow Electronics
Phone: (216) 464-2000

Hamilton/Avnet Electronics
Phone: (216) 461-1400

Pioneer Standard Electronics
Phone: (216) 587-3600

Dayton
Arrow Electronics
Phone: (513) 253-9176

Hamilton/Avnet Electronics
Phone: (513) 433-0610

Pioneer Standard Electronics
Phone: (513) 236-9900

TEXAS

Dallas
Component Specialties
Phone: (214) 357-4576

Hamilton/Avnet Electronics
Phone: (214) 661-8661

Schweber Electronics
Phone: (214) 661-5010

Houston
Component Specialties
Phone: (713) 771-7237

Hamilton/Avnet Electronics
Phone: (713) 780-1771

Schweber Electronics
Phone: (713) 784-3600

UTAH

Salt Lake City
Alta Electronics
Phone: (801) 486-7227

Hamilton/Avnet Electronics
Phone: (801) 262-8451

WASHINGTON

Bellevue
Hamilton/Avnet Electronics
Phone: (206) 746-8750

Seattle
Intermark Electronics
Phone: (206) 767-3160

Liberty Electronics
Phone: (206) 763-8200

FOR SIGNETICS PRODUCTS WORLDWIDE:

ARGENTINA

Fapesa I.y.C.
Buenos-Aires
Phone: 652-7438/7478

AUSTRIA

Osterreichische Philips
Wien
Phone: 93 26 11

AUSTRALIA

Philips Industries-ELCOMA
Lane-Cove, N.S.W.
Phone: 421261

BELGIUM

M.B.L.E.
Bruselles
Phone: 523 00 00

BRAZIL

Ibrape, S.A.
Sao Paulo
Phone: 287-7144

CANADA

Philips Electron Devices
Toronto
Phone: 425-5161

CHILE

Philips Chilena S.A.
Phone: 39-4001

DENMARK

Miniwatt A/S
Kobenhavn
Phone: (01) 69 16 22

FINLAND

Oy Philips Ab
Helsinki
Phone: 1 72 71

FRANCE

R.T.C.
Paris
Phone: 355-44-99

GERMANY

Valvo
Hamburg
Phone: (040) 3296-1

HONG KONG

Philips Hong Kong, Ltd.
Kwuntong
Phone: 3-427232

INDIA

Semiconductors, Ltd.
(REPRESENTATIVE ONLY)
Bombay
Phone: 293-667

INDONESIA

P.T. Philips-Ralin Electronics
Jakarta
Phone: 581058

IRAN

Berkeh Company, Ltd.
Tehran
Phone: 831564

ISRAEL

Rapac Electronics, Ltd.
Tel Aviv
Phone: 477115-6-7

ITALY

Philips S.p.A.
Milano
Phone: 2-6994

JAPAN

Signetics Japan, Ltd.
Tokyo
Phone: (03) 230-1521

KOREA

Philips Elec. Korea Ltd.
Seoul
Phone: 44-4202

MEXICO

Electronica S.A. de C.V.
Mexico D.F.
Phone: 533-1180

NETHERLANDS

Philips Nederland B.V.
Eindhoven
Phone: (040) 79 33 33

NEW ZEALAND

E.D.A.C., Ltd.
Wellington
Phone: 873 159

NORWAY

Electronica A.S.
Oslo
Phone: (02) 15 05 90

PHILIPPINES

Philips Industrial Dev., Inc.
Makata-Rizal
Phone: 868951-9

SINGAPORE/MALAYSIA

Philips Singapore Pte., Ltd.
Toa Payoh
Phone: 538811

SOUTH AFRICA

E.D.A.C. (PTY), Ltd.
Johannesburg
Phone: 24-6701-3

SPAIN

Copresa S.A.
Barcelona
Phone: 329 63 12

SWEDEN

Elcoma A.B.
Stockholm
Phone: 08/67 97 80

SWITZERLAND

Philips A.G.
Zurich
Phone: 01/44 22 11

TAIWAN

Philips Taiwan, Ltd.
Taipei
Phone: (02) 551-3101-5

THAILAND/LAOS

Saeng Thong Radio, Ltd.
Bangkok
Phone: 527195, 519763

UNITED KINGDOM

Mullard, Ltd.
London
Phone: 01-580 6633

UNITED STATES

Signetics International Corp.
Sunnyvale, California
Phone: (408) 739-7700

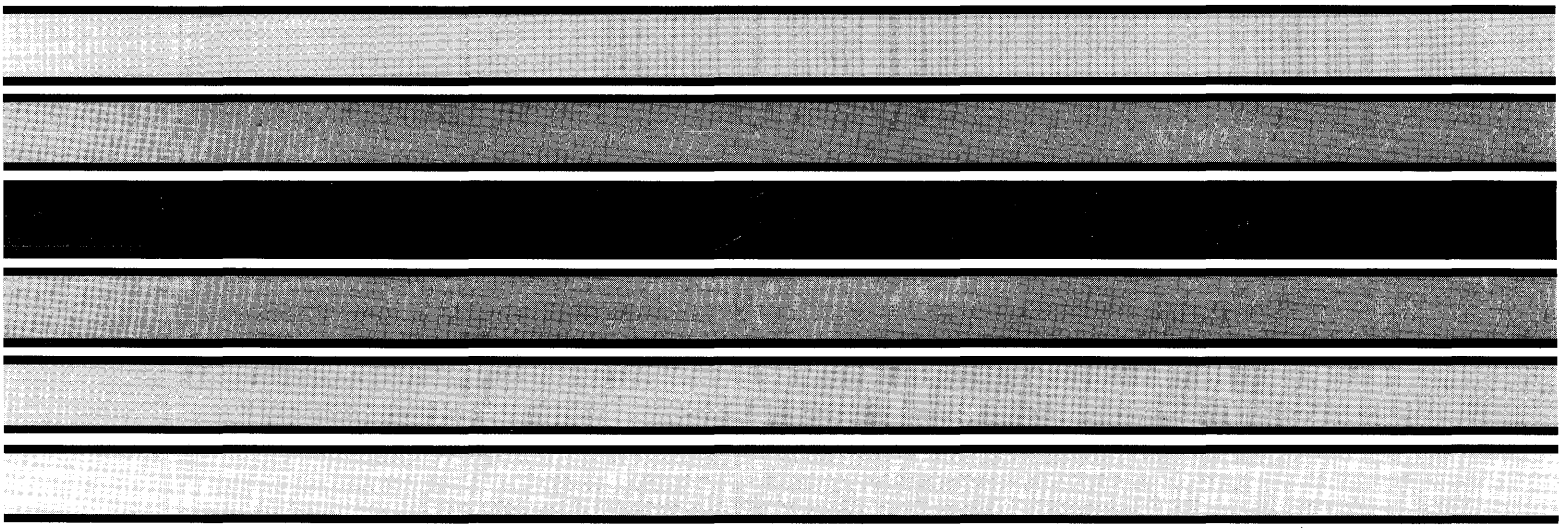
VENEZUELA, PANAMA, ARUBA, TRINIDAD

Instrulab C.A.
Caracas
Phone: 614138

signetics

a subsidiary of **U.S. Philips Corporation**

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700



Printed in USA Mar. 1977