1

CULLER/HARRISON, INC.
150-A Aero Camino
Goleta, California 93017
(805) 968-1064

September 1978

STRUCTURED DESIGN FOR AN LPC ARRAY PROCESSOR

Quarterly Technical Report

Research on Speech Compression Prototype Development

May 1978 - August 1978

PROJECT DIRECTOR: Dr. Glen J. Culler

This research was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No.: 3625
Contract No.: MDA903-78-C-0313
Expiration Date: March 20, 1979
Monitored by: DCASMA Oxnard, Calif.

CONTENTS

STRUCTURED DESIGN FOR AN LPC ARRAY PROCESSOR

1. INTRODUCTION

The research activities of the several ARPA contractors in the NSC network speech compression program have reached a stage of success in LPC algorithm development to merit real consideration of hardware economy and efficiency. To be sure, there are many other contributors to the body of research supporting the LPC technique; and most all of it taken together forms a successful whole that needs to be supported by better hardware to facilitate its expression. To this end, ARPA has funded a modest effort at CHI to design an array processor that is particularly well suited to perform the important steps of the family of LPC algorithms and at the same time to be amenable to LSI implementations. What follows is our first quarterly report on this design effort.

A. Summary of Contract Intent

The design team at CHI has been engaged in developing programmable special purpose computers for a number of years. Much of this work has been done for ARPA, principally in the direction of on-line research systems for signal processing in the acoustical frequency domain.

Our past work has a certain architectural stamp or style that results from the way we have approached the design process, the blending of very different technical backgrounds, and the techniques we have we settled on for solving internal structure problems that seem to have resulted in surprising efficiency and reliability of our devices. In our past efforts we have generally been confronted with an applicational design goal and with a fixed technology* in terms of which the design must be realized. With the rapid change of technology, we have found ourselves designing the same internal structures in different ways to take advantage of the new technology. This has led us to ask how may we represent our designs so they can be independent of the technology used to realize them. The task of designing an LPC array

*In this instance, what we mean by a technology is a set of available digital circuits and associated connectors and construction techniques.

processor that can be prototyped from existing components and then later realized as a single (or a few) LSI chips, brings this technology invariance question forward as a primary issue to be dealt with.

As a result of this, the intent of our work on contract MDA903-78-C-0313 is not only to interpret the basic operations occurring in various LPC analysis and synthesis algorithms and to design an array processor that efficiently performs them, but also to carry out the research and development required to produce a design system that will make it possible to produce technology-invariant designs. We are not suggesting that this can be done "untouched by human hands," for this is not in the nature of architecture; but rather that the human input to the design system will be primarily limited to providing creative organizational expression while the computer part of the system responds to this input by generating a number of important design outputs such as a logic simulator that realizes an equivalent design, and after geometrical specification and primitive module specification, a connector sort, a logic sort, and ultimately, diagnostic software. Of course, all of this points in the direction of how to use a computer system to help us establish how successful a design is before we build it and how to prove it works correctly after it's built.

B. Current Stage of Development

By the time contract work began we had come to understand in qualitative terms the two principal issues involved:

1. What do the LPC equations indicate about the choice of array processor architecture and how does this relate to earlier array processors.

2. What are the important software objects that could be expected to work together to achieve a structured design system that would help us solve the reconveyance problem.

Consequently, we were able to allocate tasks that could proceed in parallel. These turned out to be of two kinds -- those that were clearly understood and would play a definite roll in the work to be accomplished and those that would gather information that might help us find out how to deal with the intangibles of our problem. In the latter category we had to:

a. Compare the equations arising in representative LPC algorithms and establish performance criteria at the array process level.

b. Review the steps of our prior design efforts and the associated documentation and seek a restructuring of both these steps and the documentation to make the whole of it computationally derivable from a structured design document.

c. Study potential data structures and associated control functions that could lead to a logical simulator that is computationally derivable from a structured design document.

At present, we can report on successful accomplishment of these tasks and are well along in implementing their consequences. More specifically:

1. We have a design for an LPC array processor ready for testing.

2. A user input language for specifying a logical design has been devised and a program has been written -- called the logic assembler -- that deduces a data structure which represents the logical design and, as such, provides the basis for logic simulation.

3. We have established the concept of module substitution within the logic simulator and have designed a module analyzer and its interface to the ARPA signal processing system. By specifying to the logic assembler that a certain module is to be replaced during simulation by an actual hardware module,we can cause the logic simulator to control the module analyzer and capture the hardware module's outputs for further use in the simulation. What this means is that with the device represented by software, we can test a given hardware module "in situ" for the whole device without having the rest of the device in hand.

4. We have restructured the nature of our hardware documentation to achieve a form that can be automatically generated and yet be of similar use to engineers in both checkout and maintenance. Indeed, an engineer has to become familiar with its use, but its simplicity appears to make this no great difficulty.

5. Programs that produce design outputs of this selected documentation format are being written and some intermediate results are ready for use.

6. We have begun specifying the test programs for the LPC array processor.

The work remaining to be done on the contract consists of building and checking out the prototype, finishing the software for the structured design system and demonstrating the performance of the prototype on representative LPC algorithms.

In discussions leading up to this contract work it was anticipated that after the array processor design was appropriately in hand, we would be in a position to cooperate with some selected LSI house as an aid in the technology transfer. Our estimate is that by the end of the second quarter we will be in a position to do this.

2. A PRELIMINARY DESIGN FOR AN LPC ARRAY PROCESSOR

In J. Makhoul's paper "Stable and Efficient Lattice Methods for Linear Prediction," the best known methods of LPC analysis are brought together in a simple common framework. We choose this as our principal reference, partly because of its clarity, but mostly because it reveals a wide range of computational complexities engaged in the LPC techniques. Boiling these down to their essences results in only a few prime considerations, nevertheless arranging for their adequate implementation brings us as close to a general purpose array processor as we dare risk. The point of concern is this -- when a family of tasks are to be carried out in a single special purpose processor, they must possess some underlying simplicity that results in design advantage or we are forced to a general purpose design and the power of our approach is lost. Under the latter circumstances we must restrict the family before we can achieve a successful design.

The input to the analysis process comes from an A/D converter which may as well be taken in integer format. The dynamic range of voice signals is such that at least 8 bits/sample are required to perform any sensible analysis and 10 bits are as low as most implementers are willing to consider. In fact, most systems of reasonable quality either use 12-bit samples or incorporate some form of secondary gain ranging. In what follows we will assume that the input data is 12 bits/sample and in integer format. As to conversion rates, the NSC conferencing system used 150 μs/sample corresponding to 3.2 Khz Nyquist frequency which is marginally acceptable for male voices but unfortunately low for some female users with very small vocal tracts. A sample interval of 50 μs guarantees good sibilant data for practically all users, so we assume that the sample interval ΔT is in the range $50\ \mu s \leq \Delta T \leq 150\ \mu s$. In order to set up an array oriented process, we pick a data block size long enough to get statistical benefit in estimating the vocal tract area function and short enough to get good resolution of the change of this shape with time. Accordingly, we can expect analysis block sizes to be in the neighborhood of 100 samples with corresponding time bases in the range of 5 to 15 milliseconds.

In keeping with Makhoul's notation, let f and b represent forward and backward residuals and K denote reflection coefficient. Since $-1 < K < 1$, the direct lattice filter equations:

$$\begin{aligned} f_{m+1}(n) &= f_m(n) + K_{m+1}b_m(n-1) \\ b_{m+1}(n) &= K_{m+1}f_m(n) + b_m(n-1) \end{aligned} \qquad 1)$$

requires us to perform:

$$(\text{Fraction x Integer})_{\text{RND}} + \text{Integer} \rightarrow \text{Integer} \qquad 2)$$

as an array process. The inverse filter:

$$\begin{aligned} f_m(n) &= f_{m+1}(n) - K_{m+1}b_m(n-1) \\ b_{m+1}(n) &= K_{m+1}f_m(n) + b_m(n-1) \end{aligned} \qquad 3)$$

incorporates feedback and for small amplitudes engenders a limit cycle pathology. To handle these properly, we propose to incorporate an integer by fraction multiplication with rounding and to overlap the add (or subtract) operation. Additionally, to guard against overflow, we will arrange that only 11 of the 12 bits of integer data will be significant going into the above equations. Also, as a means to stay away from the limit cycle pathology, we will sense the output size information on the fly. That is, when the input data is too large, we will use a built-in right shift in the array process and when the output data is too small, we will insert an auxiliary left shift array process.

a) If the 12th bit of the input arrays is significant, right shift on read.

b) If the output array has no more than 8 significant bits, left shift it 3 places before proceeding.

In the computation of inner products of integer arrays like:

$$f \cdot f,\ b_{-1} \cdot b_{-1},\ f \cdot b_{-1} \qquad \text{where } b_{-1}(n) = b(n-1)$$

we need to use integer-by-integer multiplication, and since our arrays have about 100 elements, we need an integer multiply overlapped with addition into an extended accumulator.

The noncorrective methods require much more precision than the corrective ones; for this purpose, we provide programmed floating point and double precision operations. We are all aware of the much lower operation count of these methods compared to the reduction of residual corrective methods, but to be fair, we should count them in at least the double precision sense. In many cases, not having the residuals as an output of the computations creates a need for extra processes in the overall algorithm, thereby further reducing their apparent advantage.

In this section we present our current results in the development of an array processor tailored to the requirements of real time LPC analysis and synthesis. In keeping with the contract intent discussed in section 1, we will later express the design in such a manner that it can be equivalently realized in different technologies. To do this we will use the structured design approach presented in section 3 with our overall design goals established as a creative interpretation of the requirements set forth through the review of typical LPC algorithms.

The highest level top-down subdivision of the LPCAP splits it into four submodules: APU, CPU, PS, and IO.

Their overall functions can be described as follows:

1. The APU module carries out all the arithmetic operations on data as specified by the right-hand part of the instruction lines (INSTRR). Its registers provide source data for output to the IO module. Certain characteristic bits provide status information to the CPU that may be used in CPU conditionals.

2. The CPU module performs program sequencing, data addressing, indexing, APU and IO control. Its operations are specified by the left-hand part of the instruction lines (INSTRL). In run mode these instruction lines are supplied by some of the outputs of PS. When a HOST is present and when the CPU is halted, these instruction lines are supplied by the IO module, thus providing the HOST with single step control of the CPU.

3. The PS module is a program source module addressed by the CPU. When PS is a RAM (or contains a subset that is a RAM) it can be loaded by a sequence of outputs from the IO module under control of the HOST.

4. The IO module provides an interface between the LPCAP and the external world. Aside from whatever complexities are introduced by the presence of a HOST machine, it must manage voice A/D and D/A conversion and receive and transmit voice parameters to some communication link.

The main substance of this section is the treatment of each of these modules in turn.

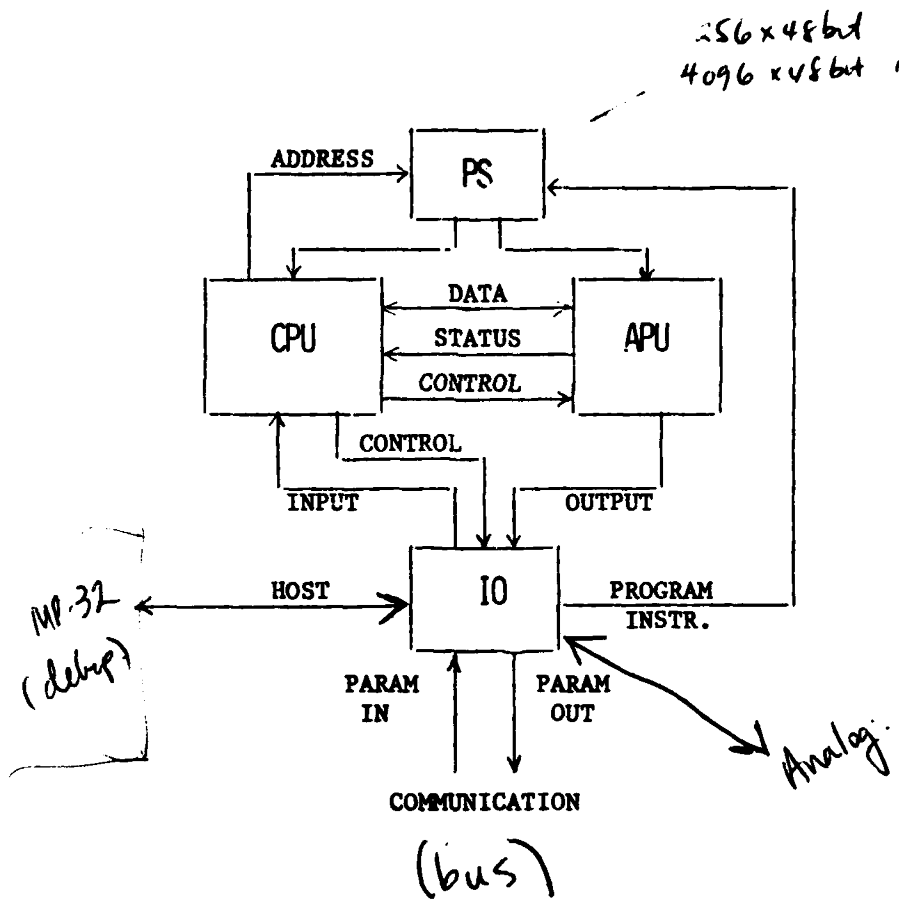


Figure 2.1 Block diagram showing the relationship of the four principal submodules of the LPCAP.

A. The APU Module

The arithmetic requirements on the APU module imposed by the form of the equations used in LPC analysis and synthesis may be summarized as follows:

1. The dynamic range of voice signals are best represented by 12-bit samples.
2. Precise inner products of sample vectors with at least 128 components are required for the noncorrective methods of computing reflection coefficients.
3. Fraction by integer multiplication is required for forming linear combinations of signals.
4. Both down scaling to guard against overflow and upscaling to guard against limit cycle pathologies of digital filtering are required.
5. The same inputs must apply to both sides of the multiplier so that the sum of squares can be computed without copying a vector.
6. The quotients count in LPC algorithms is so low that a programmed divide will suffice.

Since the LPC algorithms have a very concentrated requirement for multiplies and since most of these are associated with adds, we should focus our attention on multipliers that have configurations with both integer-by-integer full precision and fraction-by-integer rounded outputs. Such multipliers are already available with execution timesin the neighborhood of microprocessor instruction times so the architectural considerations in loading and unloading data is of paramount importance. A powerful illustration of how to do this right and the unfortunate consequences of not doing it right is provided by comparing multiply-add throughputs of the TRW chips MP12AJ and MP16AJ. The first can be used with no loss time in a looped multiply-add and the latter has a loss essentially matching its execution time. Our point is not to flamboyantly snipe at the MP16AJ but rather to emphasize the value of the architecture of the MP12AJ. Even though LSI technology will soon support the complete LPCAP on one chip, the internal multiply submodule should have the architecture of the MP12AJ.

With the architecture of the multiply nailed down, we can extend the structure to account for the requirements listed above.

a. Since the forward and backward waves are the primary inputs to the multiplier in a Burg-type analysis, and since the sum of squares of each is to be computed, we must provide a 2-to-1 multiplexer for each of the multiplier inputs (cf figure 2.2); then 5) is satisfied.

b. In order to provide precise inner products of 12-bit data in 128 word blocks, we use the full integer product with sign extension to 32 bits and use this as the A inputs to a 32-bit adder, with a 32-bit accumulator attached to the B side; this takes care of 2).

c. Since the reflection coefficients are fractions between -1 and +1, when they multiply a vector with integer components to give a vector of the same type we must use the fraction-by-integer product with rounding. To add this product to another such vector, as in the lattice filter computation, we also need a 12-bit accumulator.

d. The scaling problems intrinsic to LPC analysis and synthesis are not very severe, yet we must have some means for handling them. One simple way to do this is to maintain two flag bits under program control, call them Big and Small. Let Big selectively monitor the data written in pad and express the or of the exclusive ors of the left-hand two bits of each array value. Then Big is a 1 if any array datum is of full scale. Now let Small monitor the left-hand 4 bits in the same manner. Then Small is a 0 if no array datum has more than 8 significant bits. The capability of clearing and testing these flag bits and maintaining a scale word in pad to account for change of scale, provides a software overview of the scaling relations. We include downscaling hardware on the output from pad and will use the multiply hardware in a special pass to upscale the data when required.

e. One of the aspects of the special forms of the equations relating the forward and backward waves coupled with the direct association of adders and accumulators is that we sometimes need to exchange outputs. Perhaps a stronger way to say this is that we must be able to move data between scratch pad arrays. To satisfy this desire, we include a pair of 2-to-1 output multiplexers.

f. The final step in this overview of the APU consists of providing means for assembling and disassembling byte-oriented data sets. To this end we have selected two-way-in registers for the accumulators with interconnections to an 8-bit accumulator accepting an external input.

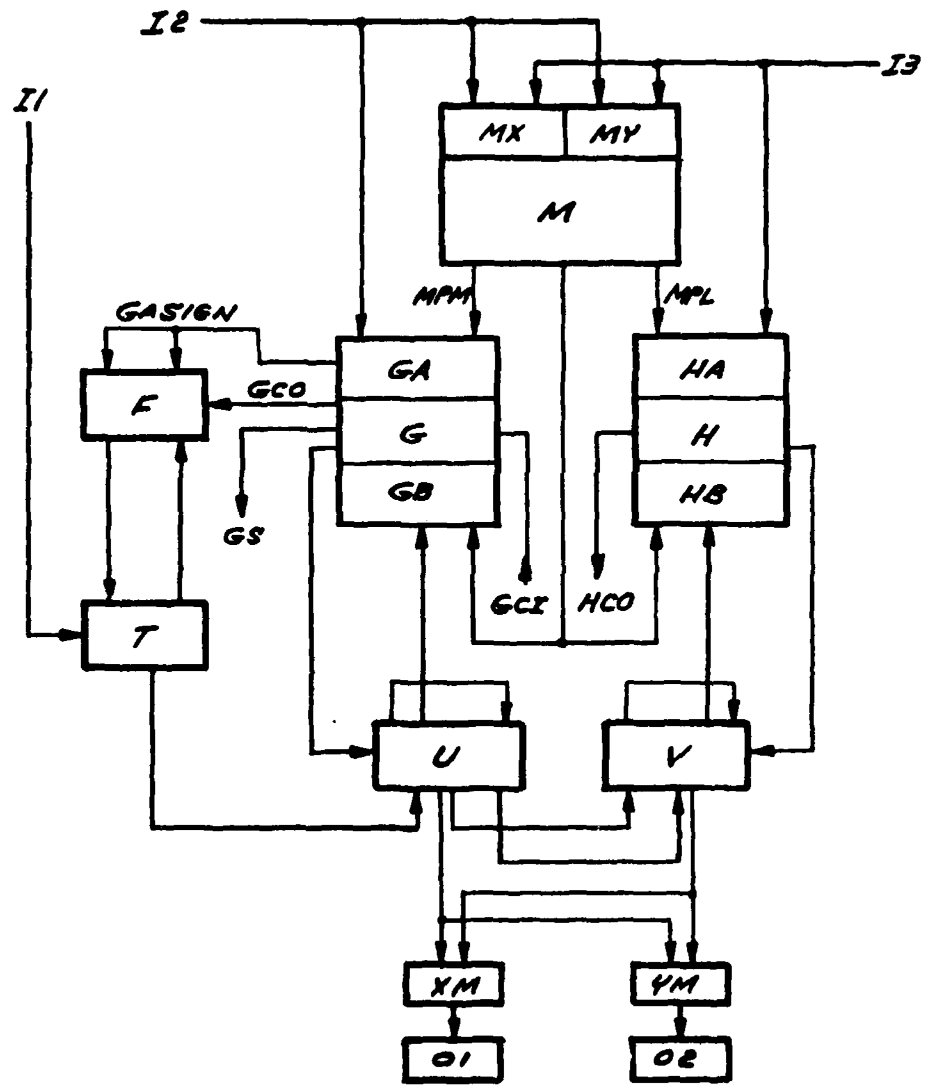
In figure 2.2 below, we give a module schematic for the APU together with a decoding chart with the module operation code definitions.

B. The CPU Module

The APU module discussed above carries with it all of the significance of design relating to LPC specializations. For the CPU we are after the simplest sufficiently rapid control to get the job done. Whereas a 12-bit structure was used in the APU module, an 8-bit structure is the proper choice for the CPU module. The two scratch pads (X and Y) of the APU need only be 256 words long to provide adequate data buffering, so corresponding to these the CPU contains two 8-bit address registers (XA and YA). Present estimated program size indicates that with an instruction page size of 256 instructions, we can expect the subroutines for analysis and synthesis to each fit within a page. For prototyping purposes we will use a 4096 instruction program source (the PS module discussed in C) with 48-bit instructions and 256 per page. In a real device this will, of course, be limited to the size determined by programming the algorithms.

The CPU can now be summarized as follows. Starting with three address counters (PSA, XA, YA) of 8 bits and an 8-bit tally register for array loop control we bus these together with 8 bits from program source (the PS value field). Branching according to the tally register, the sign bit of the APU adder and IO status bits are then provided.

In setting up an array process, the bus connection between the PS value field and the internal registers of the CPU is used to initialize them one at a time. In the array process, all of the registers can be simultaneously controlled in each micro instruction. In order to provide control for the IO module, the bus mentioned above is extended to the IO module. In the bus control fields (TRN and REC) shown in figure 2.4 below, we can address a device and then input or output as required.



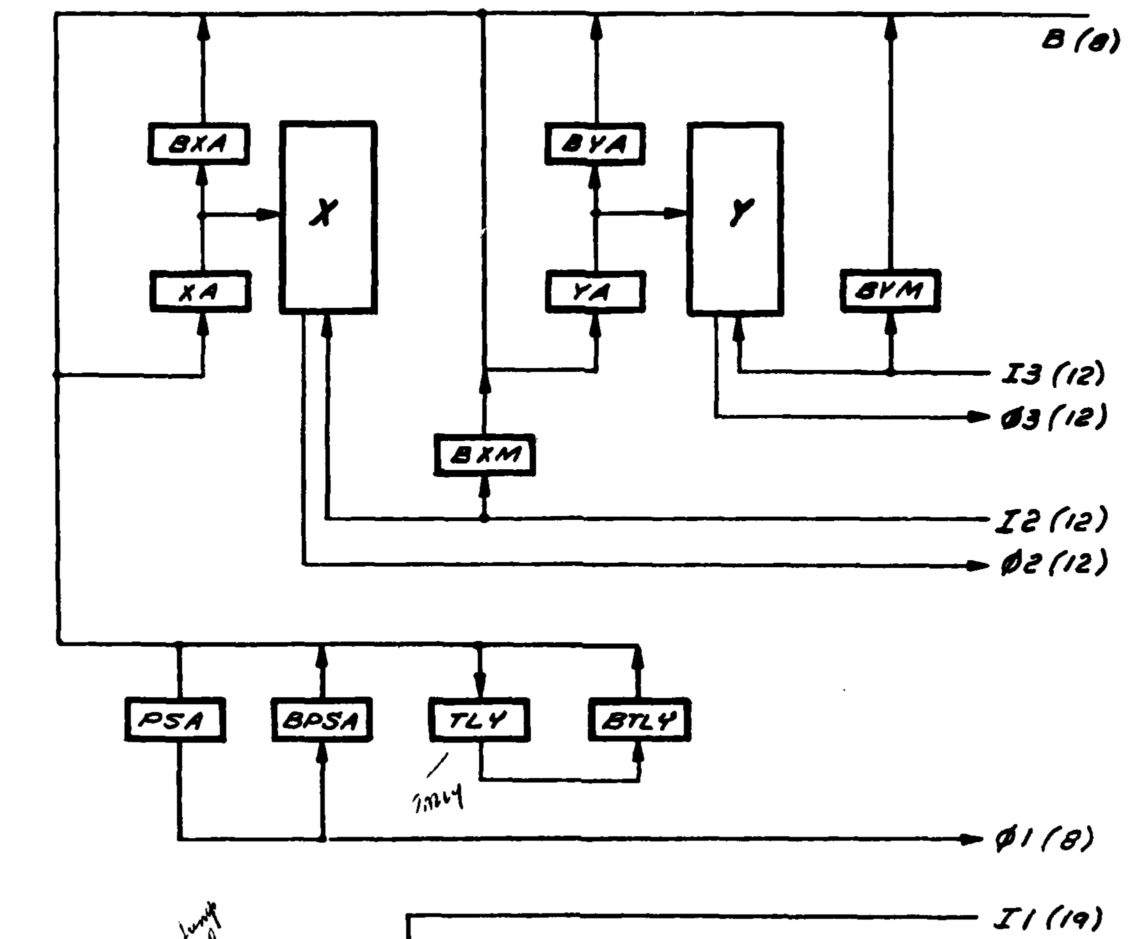
4 4 7 7 5

21 + 27 = 48 bits

	U	V	G			H			M
CODE	UEN	VEN	GOP	GA	GB	HOP	HA	HB	MULT
0	NOOP	NOOP	0	MPM	MPF	0	MPL	MPF	
1	NOOP	NOOP	SUBT	X	U	SUBT	V	V	
2	G→U	H→V	RSUBT	0	0	RSUBT	0	0	
3	RSTU	RSTV	ADD	0	0	ADD	0	0	
4			EOR			EOR			
5			OR			OR			
6			AND			AND			
7			-1			-1			

RND
MXCLK
MYCLK
MX SEL
MY SEL

Figure 2.2 APU Module Schematic



Jump Sel

+ 8 bits of PSVAL

	1	1	1	~~2~~	2	2	2	2	~~3~~ 8	3
CODE	JS	U	V	TLY	XA	YA	X	Y	TRN	REC
0	NC	NC	NC	DEC	DEC	DEC	RDX	RDY	PSVAL	NC
1	GS	LDU	LDV	INC	INC	INC	RDX	RDY	BXA	XA
2				NC	NC	NC	WRT U	WRTU	BYA	YA
3				NC	NC	NC	WRTY	WRT V	BPSA	PSA B
4									BXM	T
5									BTLY	TLY
6									BYM	DEY
7									INPUT	OUTPUT

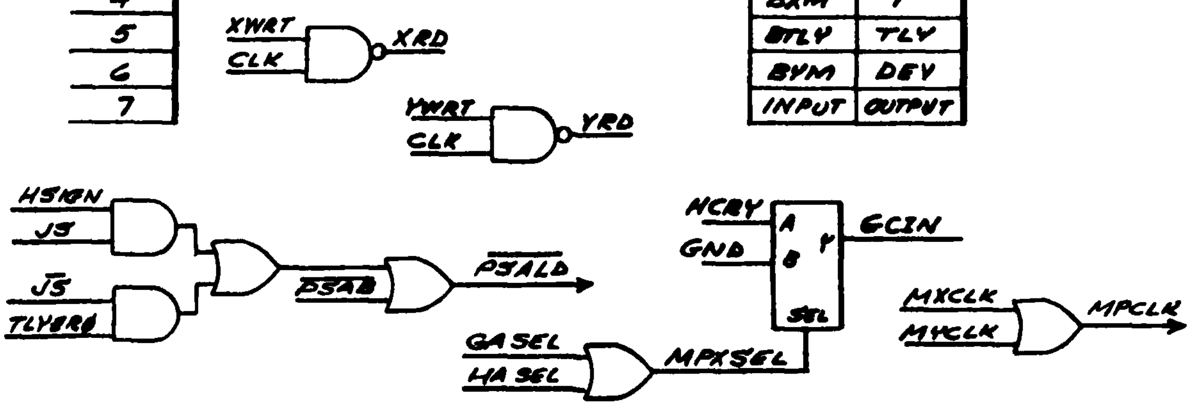


Figure 2.3 CPU Module Schematic

C. The PS Module

The program source module has been designed to allow full convenience in implementing trial LPC algorithms that will need to assess the real time performance of the LPC AP. The design is elementary and merits no special discussion.

D. The IO Module

The design of the IO module has been specialized to provide an interface between the LPC AP and the Serial Three signal processing research system. Our approach here has been to take advantage of the module analyzer -- one of the key elements in the structured design system discussed in section 3. The module analyzer is directly interfaced to the host machine and can be used as a programmable test fixture to check out the individual modules of the LPC AP as well as the entire device.

The IO module provides a means for loading programs into PS, transferring sample data blocks and parameter blocks to and from the Host. Additionally, it provides the logic of a pseudo panel that permits the Host to single step operations of the LPC AP.

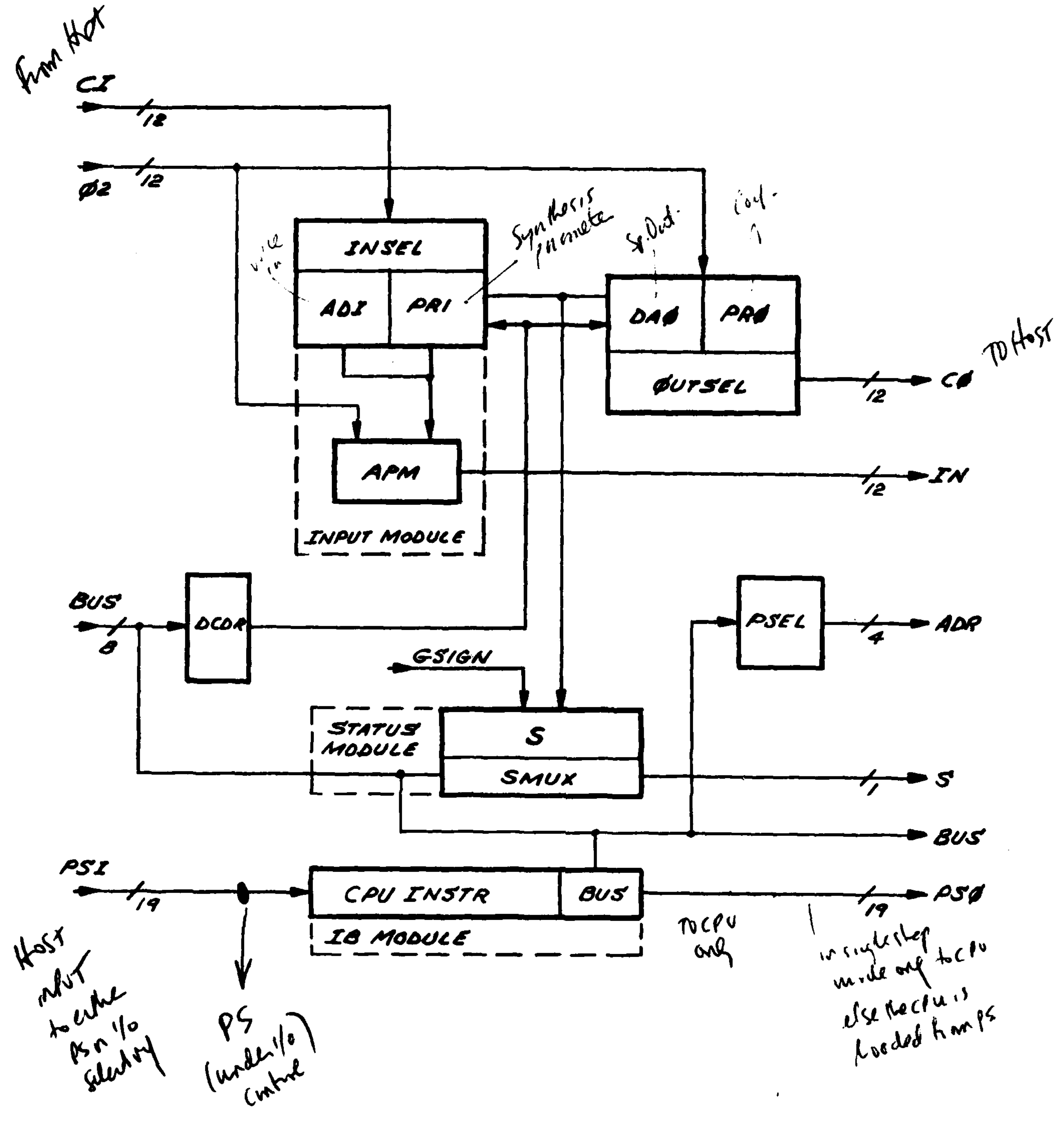


Figure 2.4 IO Module Schematic

3. THE STRUCTURED DESIGN TECHNIQUE

Logical design can be thought of as a creative process in which larger modules are defined as combinations of smaller, simpler ones in such a way as to fulfill basic design goals.

The approach can be "bottom up." In this case the designer visualizes combinations of available parts, and combinations of these, always working towards accomplishing the desired goals.

An alternative is a "top down" approach in which the design is subdivided into sections which, if implementable, would achieve the basic goals. These sections are again decomposed, and so on, until the resulting blocks are seen to be buildable from available parts. The subdivisions represent intermediate design targets.

A third approach works from both directions simultaneously, hoping to meet in the middle. At that point, sections which are known to accomplish the design goals are seen to be realizable as combinations of combinations of available hardware.

In any of these approaches the final design can be conceived as a single entity or expressed in modular terms. However, *modular design simplifies* checkout, allows for consideration of more design alternatives, and facilitates making changes.

A structured design is a tree-like hierarchy having the following characteristics.

1. For each module in the structure, the outputs are uniquely determined by the inputs, the controls and the status of memory.

2. Each module in the hierarchy is either primitive (not broken down further) or composite (a combination of lower level modules called the submodules of the module).

3. The definition of a composite module involves only its submodules. No reference to submodules of the submodules is allowed.

This means that from the point of view of a module, there is only one level below it. All strata beneath are invisible to the module, and each module acts as a primitive relative to higher level design.

In a hardware realization of the design, the primitives might be chips, and lower level, intermediate modules might be groups of chips wired together. A higher level intermediate module might be realized as a board or a group of boards connected by cables.

A. Reconveyance of Design

Assume that a design has been created which accomplishes the original goals. Of course, many alternative designs could be produced fulfilling the same criteria. In particular, one of the intermediate modules (along with its substructure) could be replaced by a different combination of submodules (along with their substructures) which performed the same function. If all of the relationships among other elements of the hierarchy were preserved, the resultant design would fulfill the original goals. Similarly, substitution for an intermediate module in the new design would lead to an additional acceptable design.

Thus, one design can be reconveyed by a whole family of equivalent designs. The highest level subdivision of the original design is of critical importance since all members of the family inherit the top structure.

If an intermediate module has been thoroughly checked out, either by theoretical simulation or by prototyping and hardware checkout, the design can be reconveyed into one in which the module in question is a primitive. The design is simplified by removing the substructure of the module from the hierarchy.

This is of practical as well as theoretical interest. For, at any stage, from design through prototyping or even production, technological advances can be easily incorporated into the existing design. If a function which was previously computed in a composite module is found to be performable by a new device (for example, a chip) the original design can be reconveyed into one where that module is a primitive. It would not be necessary to redesign the whole system, which would be required if there were no modularity.

B. Computer Involvement in the Design Process

A computer can be brought into the design process in several ways. Elements of a computer aided design system are given below.

Given an inventory of primitives and a language suitable for describing composite modules, a logic assembler can produce a data structure encapsulating the design. From this structure, appropriate programs can generate the tedious documentation necessary to implement the design -- for example, connector sort and wire list. This frees the designer to concentrate on more creative aspects such as alternative designs of intermediate targets.

An editor which enables changes to be made in documents defining the design facilitates consideration of alternatives.

A logical simulator can be used to calculate the response of any module in the system to particular sets of inputs and controls (not, of course, in real time). The simulator works on the data structure set up by the logic assembler. At each stage, the inputs and controls are passed on to submodules until they reach a primitive module. Then a subroutine pointed to in the primtive's description is invoked. The outputs thus calculated are passed back to the higher level module. Eventually, when the actions of all submodules of a module have been simulated, the outputs of that module are implicitly determined. In this way, the design can be exercised, while still in theoretical form, to make sure that each module conforms to design goals.

When a module has been realized by engineering design and prototyping, its adequcny as part of the overall design can be tested by integrating the action of a module analyzer with the simulator. The hardware realization of the module is plugged into the analyzer and switches are set to indicate which connector elements represent controls inputs and outputs.

At the stage in the simulation when execution of the module is required, inputs and controls are supplied to the external device through the module analyzer. The resulting outputs produced by the device become the values which are used at the next stage of the simulation.

C. A Mathematical Model for Computer Design

In order to implement a computer aided, design facility it is necessary to have an underlying mathematical model of a computer system and the process by which the system is constructed from components. This 'theoretical' computer, as well as its components, will be called a logical module. The model should be such that its features and functions can be translated into a computer acceptable language.

The purpose of this section is to provide basic definitions and clarify the synthesis process by revealing rules of combination which result in logical modules. We must also develop a language that provides precise expression of one logical module in terms of others. These concerns motivate the definitions which follow.

The definitions are in terms of vector spaces, so as a preliminary step, we indicate notations which will be used elsewhere in this section.

Let X and Y be finite dimensional, vector spaces over finite fields.

1. $X \times Y$, the cross-product of X and Y, is the set of pairs of vectors (x, y) where $x \in X$ and $y \in Y$.

2. If X and Y are over the same field, then $X \oplus Y$, the direct sum of X and Y, is the set of vectors $x|y$. If

$$x = (x_1, \ldots, x_n) \in X \text{ and } y = (y_1, \ldots, y_m) \in Y$$

then $x|y$ has coordinate expression

$$x|y = (x_1, \ldots, x_n, y_1, \ldots, y_m)$$

3. If X and Y are of the same dimension, n, and over the same field, they are isomorphic and can be identified. This identification will be denoted $X \leftrightarrow Y$.

4. The <u>dimension</u> of X will be denoted by <u>dim X</u>

Logical Modules

Defn 1: A Logical Module L is an ordered sextuple

$$L = \langle C, I, IO, M, O, F \rangle$$

where C, I, IO, M, O are finite dimensional vector spaces over finite fields and F is a function from C x I x IO x M into O such that, for each set of vectors $c \in C$, $i \in I$, $io \in IO$, $m \in M$ there is a unique vector $o \in O$ for which

$$F(c, i, io, m) = o$$

The notation used here has the following mneumonic relations:

C is the control space of the module
I is the input space of the module
IO is the input/output space of the module
M is the memory space of the module
O is the output space of the module
F is the function of the module

IO and/or M may be empty. The vectors of IO represent either input or output depending upon the control vector.

The outputs can be 'switchable,' in which case the field associated with O has at least three elements.

For simplicity O is presented as a single space; however, it some but not all outputs are switchable, O is really a pair of vector spaces.

The crux of the definition dictates that for every combination of control, input, IO and memory vectors, there is a unique output vector 'computed' by the function F.

The concept of a logical module applies over a broad range of complexities, extending from 'chips' to an entire computer system.

Defn 2: Let L be a logical module. The boundary of L, denoted by δL, is the quadruple

$$\delta L = \langle C, I, IO, O \rangle$$

The quantity

$$\dim \delta L = \dim C + \dim I + \dim IO + \dim O$$

will be called the *dimension* of the boundary.

The boundary is not itself a vector space, since the underlying fields of the constituent spaces may be different. However, the individual coordinates of the constituent spaces can be ordered to run from 1 to $\dim \delta L$ (as for $x|y$). This will be called an ordering of the *elements* of the boundary.

Defn 3: A logical module L is *primitive* if its function F is explicitly given as part of the module's definition.

Defn 4: A logical module L is *composite* if it is defined by a combination of logical modules

$$L_i \quad L_1, L_2, \ldots , L_n$$

The L_i are called *submodules* of the module L. A submodule can be either primitive or itself a composite module. Only primitive modules can have memory.

In order to combine logical modules and form more comprehensive structures which are themselves logical modules, we must specify a *boundary correspondence* that defines a boundary δL for a composite module L in terms of the boundaries δL_i of the constituent submodules. The boundary elements of δL must be associated with boundary elements of submodules. A single element of δL may be associated with elements in more than one submodule or even with more than one boundary element of a single module. A further restriction on the boundary correspondence is that output elements of the composite module boundary be associated with output elements of the submodules. (It is not required that, for example, each input to the composite module be an input to some submodule.)

It will usually be the case that there are some submodule boundary elements which are not associated with elements of the composite boundary. Mappings may be specified among such elements.

These mappings are called interconnects.

From the definitions above, it seems clear that the function of a composite module is derivable from the structural relationships of the boundaries of its submodules and the submodule functions. In fact, the boundary correspondences and interconnects define the function F of the composite module L implicitly in terms of the functions F_i of the submodules L_i.

To aid us in giving some specific examples of combinations, we introduce the following notions.

Defn 5: Two logical modules are equivalent if there exists a one-one correspondence of their boundaries which results in their functions becoming identical.

Defn 6: A composite logical module is homogeneous if all of its submodules are equivalent.

Simple Composite Modules

As a rule it is difficult to desicribe the functional relationships between the function F of the composite module L and the functions F_i of the submodules. In such a case, the boundary correspondences and interconnects can be listed element by element. However, when the boundary mappings can be described in vector terms, F often has a simple representation in terms of the F_i. We detail a few such situations below.

Parallel Modules

In the design of a computer system, we frequently encounter logical modules that occur in parallel configuration; that is, their inputs, outputs, memories and controls are respectively independent and consequently their functions can be simultaneously used.

For example, consider an 'or' chip with 4 'or' gates which can be used independently. In this case the individual gates, not the chip, are the primitives; the chip represents a composite module with parallel submodules.

Let

$$L_1 = <C_1, I_1, IO_1, M_1, O_1, F_1> \quad \text{and}$$

$$L_2 = <C_2, I_2, IO_2, M_2, O_2, F_2>$$

be logical modules; let

$$C = C_1 + C_2 \quad I = I_1 + I_2 \quad IO = IO_1 + IO_2$$

$$M = M_1 + M_2 \quad O = O_1 + O_2 \quad \text{and define}$$

$$F(c, i, io, m) = F_1(c_1, i_1, io_1, m_1)|F_2(c_2, i_2, io_2, m_2).$$

for each combination of control input IO and memory vectors.

Then $L = <C, I, IO, M, O, F>$ is a logical module with parallel submodules. The boundary of L is $<C_1 + C_2, I_1 + I_2, IO_1 + IO_2, O_1 + O_2>$. $\dim \delta L = \dim \delta L_1 + \dim \delta L_2$. There are no interconnects.

Widened Modules

Using homogeneous submodules it is easy to define a wider module with the same controls. Since homogeneous modules are logically equivalent, we may use the ordering assigned by the equivalence and identify corresponding controls. This, in effect, gangs the controls of the submodules together. The inputs, outputs and memories are combined in parallel as above.

Let

$$C = C_1 \leftrightarrow C_2 \quad I = I_1 + I_2 \quad IO = IO_1 + IO_2$$

$$M = M_1 + M_2 \quad O = O_1 + O_2 \quad \text{and define}$$

$$F(c, i, io, m) = F\ (c,\ i_1, io_1, m_1)|\ F_2(c,\ i_2, io_2, m_2)$$

for each combination of control, input, IO and memory vectors.

$$\dim \delta L = \dim C_1 + 2 \dim I_1 + 2 \dim IO_1 + 2 \dim O_1 < \dim \delta L_1 + \dim \delta L_2$$

There are no interconnects.

An example of the realization of a widened module is a 12-bit register made from three 4-bit register chips.

Composition of Modules

Logical modules may be combined by composition in two important ways, viz the output of one may provide inputs or controls for another. In either case, the composition mapping represents interconnects.

Function Composition

If the dimension of the output space of the first module agrees with the dimension of the input space of the second module, then the composition of these is a logical module. Let

$$C = C_1 + C_2 \quad I = I_1 \quad IO = IO_1 + IO_2$$

$$M = M_1 + M_2 \quad O = O_2 \quad \text{and define}$$

$$F(c, i, io, m) = F_2(c_2, F_1(c_1, i_1, io_1, m_1), io_2, m_2)$$

for each combination of control, input, IO, and memory vectors.

Such modules are, in effect, cascaded and, in the case that L_1 can be preparing its new outputs without modifying its prior outputs, this composition is said to form a pipeline.

Control Composition

If the output space of one module coincides with the control space of another module, the composition of these modules is a logical module.

Let $\dim O_1 = \dim C_2$,

$$C = C_1 \quad I = I_1 + I_2 \quad IO = IO_1 + IO_2$$

$$M = M_1 + M_2 \quad O = O_2 \text{ and define}$$

$$F(c, i, io, m) = F_2(F_1(c_1, i_1, io_1, m_1), i_2, io_2, m_2)$$

for each combination of control, input, IO and memory vectors.

In such a case as this we say the module L_1 has provided the control for module L_2. Such a module is called a control module.

In the various composite modules treated above, we have limited ourselves to simple combinations, yet combinations of these combinations will provide most of what is required to treat significant computer systems.

D. Defining Logical Modules

The definitions of the previous sections were of conceptual intent; that is, they essentially assigned formal names to intuitive concepts. In order to translate these concepts into somethings a computer can get its teeth into, we must present a module definition language and a corresponding data structure. Since module combinations can be completely described in terms of the module boundary and the boundaries of the submodules, the language is a vehicle for specifications of boundaries.

Three kinds of documents suffice for logical module definitions:

1) definition document for a primitive module
2) definitive document for a composite module
3) definition document for a module which is a copy of a previously defined module (either primitive or composite).

Definition document for a primitive module

The boundary elements of the module must be ordered and assigned 'local' names. For example, if the module is to be realized as a 'chip,' the names given to the pins by the manufacturer will suffice. The document then consists of a list of those names, together with a description of the use made of each boundary element (control, input, IO, output). The amount of memory required (if any) must be detailed.

In addition, pointers must be provided to subroutines which compute the module's function. Up to three such subroutines are allowed. This allows control, input, IO and output vectors to be divided into convenient subsets, and enables subsets of outputs to be computed from subsets of inputs.

Definition document for a composite module

The document contains a list of submodules of the module. Then, for each element of the composite module boundary, two items of information must be given: the use of this element (input, control, etc.) and the signal name associated with that element. Each signal name must appear in the definition document of at least one submodule.

The signal names in the document definitions for the module and its submodules determine the boundary correspondences and the interconnects between the submodules. The major design step, then, is encompassed in the assignment of signal names.

Definition document for a copy of a previously defined module

In such a document, the name given in the original definition document must be provided. Then the boundary elements of the module are assigned new signal names. The elements are listed in the same order as in the original module definition document. For each boundary element, its type (control, input, etc.) as well as its assigned (new) signal name is designated.

Using such documents allows many copies to be made from one definition document. In the case of primitives, this avoids proliferation of copies of the subroutine for the module. In the case of composite modules, it enhances modularity by allowing one set of signal names to be associated with boundary elements within a module, and a different set to be assigned to those same elements when the module is used as a component of a larger module.

If the definition document for a submodule L_1 of a module L is itself a composite module, definition document, then the signal names connecting boundary elements of L with boundary elements of L_1 will continue into the interior of L_1. Such a signal connects at least three levels of the structure. This runs counter to modularity but is allowed in our system for flexibility.

When the design documents for all submodules of a module are copies of previously defined modules, all signals among the boundary elements of the module and the submodules stop at the boundaries of the submodules, preserving modularity. That is, wiring between modules is independent of wiring within those modules.

Examples of definition documents

We illustrate the concepts of the previous sections by including printouts of design documents.

Document 1 is a composite module, definition document for an 8-bit counter, CNTR8. It is composed of the submodules CNT1, CNT2, CNTB, and has 24 boundary elements. Signal names on the same line represent elements of the same type -- control, input, etc. The signal names listed here appear in the design definition documents for the modules CNT1, CNT2 and CNT8.

Document 2 defines a module ADDR8 which is a copy of the CNTR8 module. ADDR8 is combined with another module to form a module PAD. The definition document for PAD is shown in document 3.

Those signal names in document 2 which also appear in document 3, (e.g. BUS 7) detail part of the boundary correspondence for the composite module PAD. Names which appear in document 2 but not in document 3 (e.g. A7) will occur in the definition document for the other submodule of PAD (RAM12). These represent interconnects betwen the submodules of PAD.

```
DØCUMENT CNTR8           PAGE       1    LINE       1
*CNTR8 8 bit counter module with bussed and regular
outputs
SUBMØDS    CNT1,CNT2,CNTB
CØNTRØL    LDn,INC,BA,BAn,CLK,G1n,G2n
IØ         BUS7,BUS6,BUS5,BUS4,BUS3,BUS2,BUS1,BUS0
ØUTPUT     CNT7,CNT6,CNT5,CNT4,CNT3,CNT2,CNT1,CNT0
ØUTPUT     CNTZRØn
END
```

Document 1. Definition document for a composite module

```
DØCUMENT ADDR8          PAGE        1    LINE        1
*ADDR8      8 bit ADDRESS REGISTER/CNTR
TYPE        CNTR8
CØNTRØL     LDn,INC,BA,BAn,CLK,DC,GND
BUS         BUS7,BUS6,BUS5,BUS4,BUS3,BUS2,BUS1,BUS0
ØUTPUT      A7,A6,A5,A4,A3,A2,A1,A0
ØUTPUT      ØPEN
            END
```

Document 2. Definition document for a 'copy' of the module definition document 1

```
DØCUMENT PAD            PAGE        1   LINE       1
*PAD         256 word by 12 bit RAM with address register
counter
SUBMØDS      ADDR8,RAM12
INPUT        IN11,IN10,IN09,IN08,IN07,IN06
INPUT        IN05,IN04,IN03,IN02,IN01,IN00
CØNTRØL      RD,LDn,DC,INC,BA,BAn,CLK
BUS          BUS7,BUS6,BUS5,BUS4,BUS3,BUS2,BUS1,BUS0
ØUTPUT       ØUT11,ØUT10,ØUT09,ØUT08,ØUT07,ØUT06
ØUTPUT       ØUT05,ØUT04,ØUT03,ØUT02,ØUT01,ØUT00
END
```

Document 3. Definition document for a composite module on the next higher level. The module defined in document 2 is a submodule of this module.

E. The Integrated Logical Simulator

The logical module simulator permits verification of the logical design of a module prior to its construction. It also can be used to compute expected results for comparison with values measured using the module analyzer. In addition, algorithm development for a device can be carried out using the simulator.

As part of an integrated logical design facility, the simulator uses the common logical module structure for description of the module being simulated. The module is defined in terms of its boundary elements and its constituent submodules. These submodules may themselves be composite modules, or they may be uses of previously defined modules. At the lowest level, these are primitive modules with associated programs to perform their function. It is also possible to replace a module in the simulation with the physical device using the module analyzer. In this mode, signals to and from the module analyzer are used with those produced through simulation of the remaining modules.

The simulation is performed at discrete time steps. For each step, the processing consists of computing the outputs of the module from given inputs and controls. All output computation actually takes place at the lowest, or primitive level. Input and control signals are distributed from higher level modules down through submodules until primitive modules are reached. Signals are collected at the boundary of each primitive module until those needed to compute outputs are available; then outputs are computed. These outputs are then distributed throughout the module structure wherever they are needed as inputs or control signals. The processing for a step is complete when all primitive modules have computed their outputs.

A logical module is described to the simulator by specifying its boundary elements and either identifying the module type from a library of previously defined modules or by listing the constituent submodules. The submodule descriptions are in the same form: either uses of previously defined modules or sets of interconnected modules. The boundary specification for each module is ordered by position. The use of the boundary element (Input, Output, Control, Bus) is given explicitly. The signal name given for the element defines the interconnection structure between submodules and to the boundary of the composite module.

Primitive modules are defined for simulation by providing an executable subroutine which computes the outputs of the module from its inputs and controls. In addition, a table must be provided which describes the use made of each boundary element of the module, together with the amount of memory which the module contains. Each such module can have up to three separate subroutines. This permits computation of some outputs based on the presence of only a partial subset of the inputs. This simplifies the processing of modules containing memory elements whose state is to be used during the current step and updated for use in the following step. One subroutine makes their outputs available as soon as the appropriate control signals are present, another updates their internal state when the input data is available.

A two-part, parallel data structure is used for the logic simulation. The first part, the interconnect table (Table 3.1), represents the logical interconnections of the boundary of a module with the boundaries of its constituent submodules. This structure provides the paths for transfer of logic values from their source to all modules where they are used. Each module entry in this structure has a set of pointers to the entries for each of its submodules and a pointer entry for each boundary element of the module and of each of its submodules. These pointer entries are linked together in circular lists connecting all boundary points which share a common signal name. Primitive module entries in this structure contain the subroutines for computing their outputs. If the same module is used more than once, only one copy of it is needed in this structure since its connections to other modules for each use are described at the next higher level.

The second structure used for simulation is a state table which maintains the record of the current state of the module (Table 3.2). This table includes the memory for each primitive module and the current state of all its boundary elements. At higher levels, this table contains only a list of pointers to the entries for each submodule. Since identical modules need not have identical states, this table has separate entries for each use of a module.

These data structures are created from the text description of a module in a two-stage process. Each composite module is first processed by the logic simulation assembler to produce its interconnect table, which shows all logic variables present on its boundary as well as all interconnections between the submodules which make up this module. The output of the logic assembler for a module is this interconnect table and a list of the submodules referenced. When a module is to be simulated, the interconnect tables for all modules contained in it are loaded along with primitive module code and linked together. At the same time, the state table is constructed and the initial variable state filled in. This completes the second stage of the creation of the data structures needed and is followed by the actual simulation steps.

Table 3.1 Interconnect Table Formats for a Module

a. Nonprimitive Module

Word #	Contents
0	# submodules + 1 (M+1)
1-M	Pointer to each submodule entry (*-displacements)
M+1	# boundary elements for module (N)
M+2 - M+N+2	Pointer entry for each boundary element
M+N+3 - END	Entries for submodules A (interconnect table for submodule) filled in during second phase. One-word pointer entry for each boundary element of submodule, contains submodule index/pin index of next use of signal. Each element entry, whether on the boundary of the module or some submodule, is a circular list link of the form: submodule index/element index where submodule index 0 refers to the boundary of the module. All elements with the same signal name are linked together in a ring.

b. Primitive Module

Word 0:	length of state table entry for module
Word 1:	A (source for state table entry)
Word 2:	A (primary processing code)
Word 3:	control entry 1 A(first control processing code)
Word 4:	control entry 2 A(second control processing code)

A primitive module has up to three processing sections. Each section has its own entry point and is executed according to the following rules:

1. Entry point given at module base + 2. Standard processing. Executed when the specified number of scheduling inputs and controls are valid.
2. Entry at module base + 3: Executed when control with subtype 1 is valid.
3. Entry at module base + 4: Executed when a control with subtype 2 is valid and low or a control with subtype 3 is valid and high.

Table 3.2 State Table Formats

a. State Table Nonprimitive Module Entry

Words 1 to N-1:	Address of state table entry for each submodule.
Word N:	#1 address of state table entry for last submodule.

b. State Table Primitive Module Entry

Word 0: #2

S	# needed	R	# received
15	14 8	7	6 0

R and S are set when control entry 1 or 2 respectively can be executed.

# received is a count of the number of signals which have their schedule bit set which have been received.

# needed is the number of these signals which must be present before the primary entry can be executed.

Words 1 to $\frac{N}{2}$: [#1]

T	ST	S	V	N	C	T	ST	S	V	N	C
15 14	13 12	11	10	9	8	7 6	5 4	3	2	1	0

One byte for each signal:

T = Type	00 = Control
	01 = Output
	10 = Input
	11 = Bus

ST = Subtype	Output Bus	Control
	00 = Combinatorial/Ungated	00 = Other
	10 = Latched/Ungated	01 = Immediate entry 1
	01 = Combinatorial/enabled	10 = Active low - entry 2
	11 = Latched/enabled	11 = Active high - entry 2

S = Schedule bit - increment received count when this signal is received

V = Valid - set when signal is valid

N = Next Value - retains value for use in clocked outputs

C = Current Value - state of signal if V bit is set.

The last signal word has its flag 1 set. If the module requires memory, it will be located immediately after the signal entries.

F. The Integrated Module Analyzer

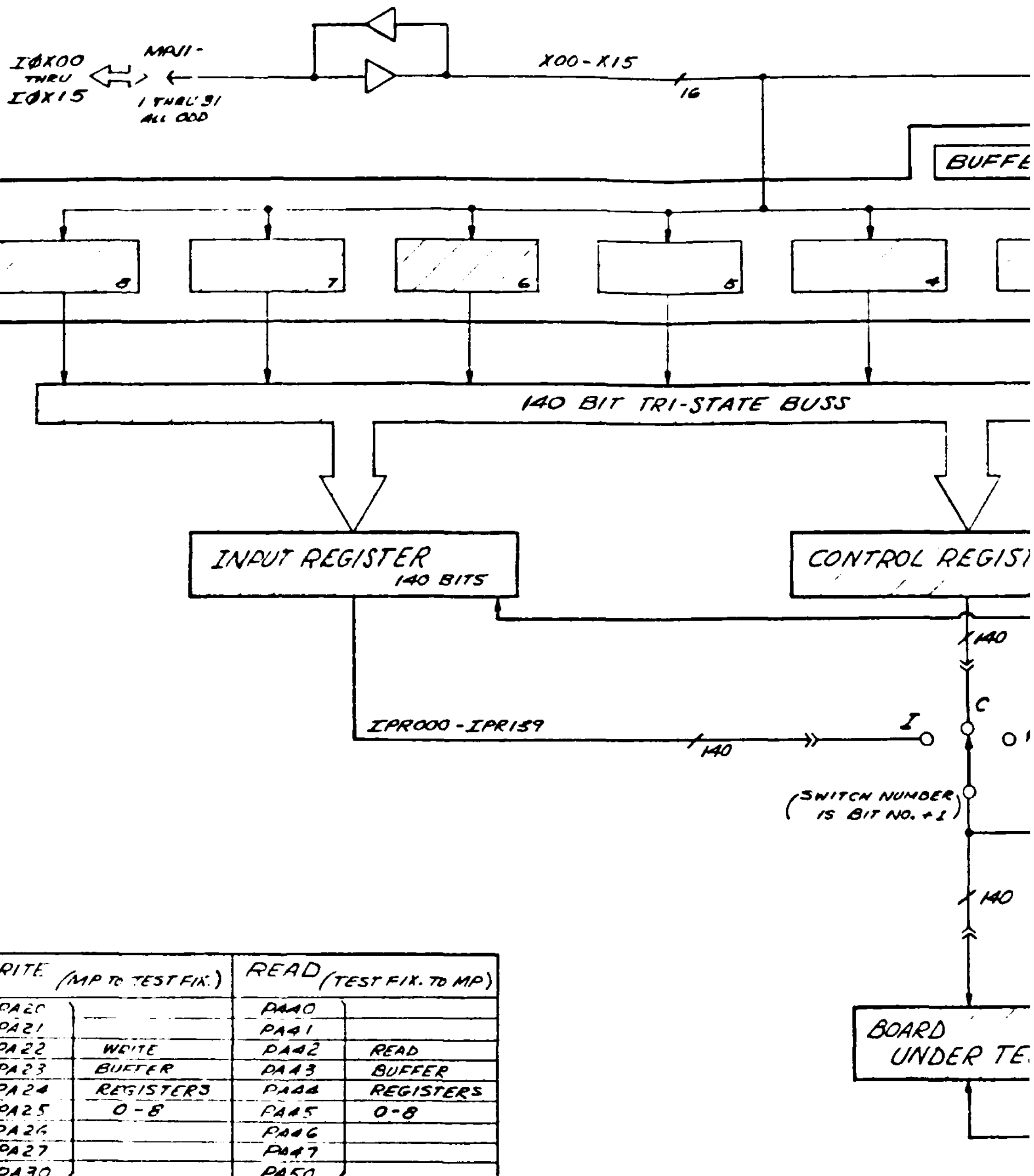
The Module Analyzer (MA) performs post construction verification of the logical and engineering design. At this stage in the design the modules involved should be logically correct. However, engineering considerations may affect the performance and implementation of the logical design. Timing characteristics, signal transmission paths, power requirements, and noise environment are among the possible engineering design considerations that may affect the final form of the hardware.

The realization of the prototype MA was influenced by two factors: (1) the desire to have its elements under direct external software control, and (2) to have a working prototype quickly available. These two factors resulted in a device that has overall structural simplicity. The final implementation of the MA may involve more sophistication that could allow more automation of hardware checkout. For example, the manual switches used for sampling selection could be electronic switches under program control. Nevertheless the prototype MA has proven itself a useful tool. The remainder of this section documents the MA and contains the following paragraphs: Paragraph 1 is a functional description of the entire unit including each of its major registers. Paragraph 2 describes how to operate the MA using the manual controls, external connections, and programmatic capabilities. Para. 3 is a detailed circuit description. Appendices include sample microprograms, and a complete set of drawings.

1. Functional Description

The main features of the MA are depicted in the block diagram of Figure 3.1. Generally, the purpose of the MA is to provide both static and dynamic inputs to a board under test and then to sample the board's outputs after a specified time period. This time period is specified by external control, so that a sequence of tests can determine when an output changes within 5 ns. External addresses and data are supplied on PAX00* to PAX05* and IØX00 to IØX15, respectively. The internal registers are addressed according to the decoding scheme included in Figure 3.1.

The Buffer Register (BR) has several purposes. It serves as a converter from (or to) external 16-bit data to (or from) internal 140-bit data. This is accomplished by nine separately addressable 16-bit registers within BR. After data has been transferred into BR, the information may then be loaded into either the Input Register or the Control Register, both 140 bits.



WRITE (MP TO TEST FIX.)		READ (TEST FIX. TO MP)	
PA20		PA40	
PA21		PA41	
PA22	WRITE	PA42	READ
PA23	BUFFER	PA43	BUFFER
PA24	REGISTERS	PA44	REGISTERS
PA25	0 - 8	PA45	0 - 8
PA26		PA46	
PA27		PA47	
PA30		PA50	
PA31	LOAD COUNTER	PA51	READ STATUS
PA32	LOAD INPUT R		
PA33	LOAD CNTRL R		
PA34	RESET		

BRUNING 40-21

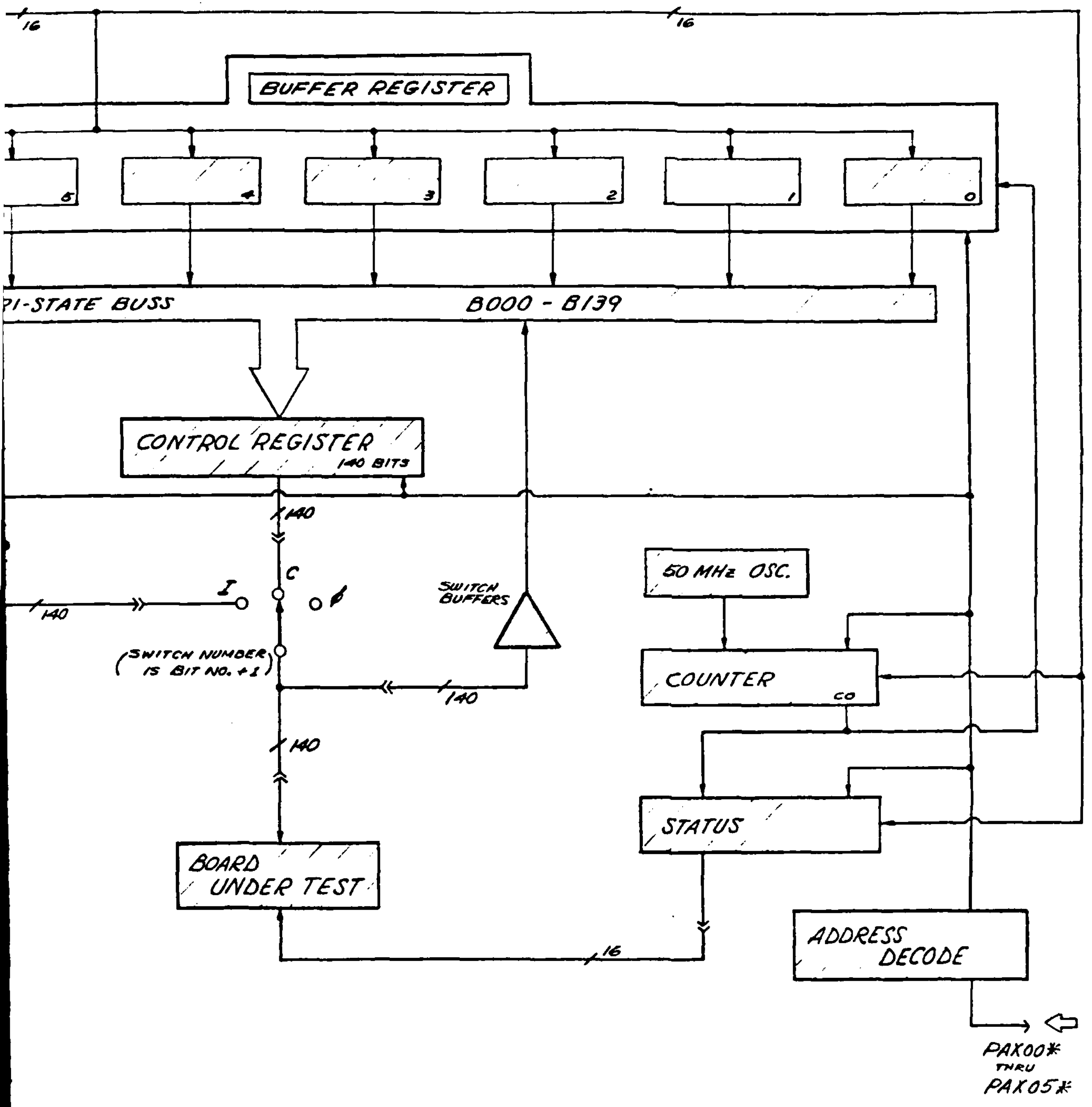


Figure 3.1

TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC		
DECIMAL .	MODULE ANALYZER	SCALE —	DRAWN BY [illegible] / APPROVED BY
FRACTIONAL ±	TITLE BLOCK DIAGRAM - MODULE ANALYZER		
ANGULAR ±	DATE 7/29/80	DRAWING NUMBER MA01	

BR also serves as the data storage register when the board under test is sampled.

The Input Register (IPR) provides static input data to the board under test. It is a 140-bit register that is loaded from BR. Its outputs may be connected to the board under test by switch settings.

The Control Register (CØN) provides dynamic input data to the board under test. The clocking of CØN provides an initiate pulse to the MA that begins the test sequence. This initiate pulse defines the time of change of any bit in CØN and also provides a start pulse to the counter.

The Counter consists of two elements, the Count Save (CS) register and a 50 Mhz up-counting register (CNT). The counter has two functions: It generates timing strobes for the various data transfer operations to, from, and within the MA. It also provides the sample strobe that defines when data from the board under test is loaded into BR and the Status Register. The length of time between the initiate pulse of CØN and the sample pulse of CNT is determined by the number loaded into CS. This time may be practically varied from 30 ns to 327 μs in 5 ns intervals.

The Status Register (STAT) is a 16-bit register whose inputs are directly attached to external probes. These probes may be connected to any signal on the board under test; this provides the capability to sample 16 internal states in addition to the I/O pins. Data from STAT is directly read from the X-Bus since it is only 16 bits wide.

Switches on the MA are set to correspond to the function of each pin. Input pins are switched to "I" or "C" depending on whether the input is a static or dynamic one. At sample time the state of these inputs is loaded into BR. Output pins are switched to "Ø" so that at sample time the state of the output pin is recorded. Power and ground pins are switched to "Ø"; power wiring is not a logic input so it is wired separately to the board connector.

2. Operation

The operation of the MA is governed by three conditions: (1) External connections, (2) switch settings, and (3) control of external logic signals.

The external connections must be made prior to powering on. The power cable must be plugged to both the MA and an external 5V power supply. A 50-pin flat cable must be connected to Connector MP1. This provides external input/output; pin assignments are given in Table 3.3. The opposite end is connected to the device providing control. The board to be tested should be

Table 3.3 MP1 Pin Assignments

Pin	Signal	Pin	Signal
1	IØX00*	26	IØX12*R
2	IØX00*R	27	IØX13*
3	IØX01*	28	IØX13*R
4	IØX01*R	29	IØX14*
5	IØX02*	30	IØX14*R
6	IØX02*R	31	IØX15*
7	IØX03*	32	IØX15*R
8	IØX03*R	33	IØXENA*
9	IØX04*	34	IØXENA*R
10	IØX04*R	35	UNUSED
11	IØX05*	36	UNUSED
12	IØX05*R	37	PAX00*
13	IØX06*	38	PAX00*R
14	IØX06*R	39	PAX01*
15	IØX07*	40	PAX01*R
16	IØX07*R	41	PAX02*R
17	IØX08*	42	PAX02*R
18	IØX08*R	43	PAX03*
19	IØX09*	44	PAX03*R
20	IØX09*R	45	PAX04*
21	IØX10*	46	PAX04*R
22	IØX10*R	47	PAX05*
23	IØX11*	48	PAX05*R
24	IØX11*R	49	UNUSED
25	IØX12*	50	UNUSED

43

placed in the edge connector provided on the top of the MA. When these connections have been made the power-on switch may be enabled.

Before programmatic manipulation is begun, the desired switch settings should be made. Output pins of the board under test should have their corresponding switches set to "Ø". Inputs should be set to "I" or "C" depending on whether the pin will have a static or dynamic input. In addition to switch settings the user may desire to attach any or all of the 16 probes to internal points on the board.

Programmatic control of the MA is achieved by manipulation of the address (PAX00* to PAX15*), data (IØX00 to IØX15), and enable (IØXENA*) signals. Generally the procedure to perform a test sequence is as follows:

(1) Reset the MA (PA = 34_8).

(2) Load CS with the desired 16-bit count (PA = 31_8).

(3) Load BR with the desired data for IPR (PA 20 to 30).

(4) Transfer BR to IPR (PA32).

(5) Load BR with the desired data for CØN (PA 20 to 30).

(6) Transfer BR to CØN (PA33); this initiates the count previously loaded into CS.

(7) Wait for sample pulse.

(8) Read BR (PA 40 to 50).

(9) Read STAT (PA 51).

These data transfers are controlled by internally generated timing strobes. These strobes define the times during which commands and data are to be supplied to the MA, and they define when MA output data is be externally sampled. The timing of input to the MA is given in Figure 3.2. Both IØX (data) and address (PA*) should be changed a minimum of 100 ns before making IØXENA* active. IØXENA* enables the X-Bus, and 820 ns later the timing pulse, STRØBE1*, is activated. In Figure 3.2 the signal REG DECODE represents the register enable signal that is a decode of PA 20-34. The combination of decode and STRØBE1* generates the appropriate register clock. This latches the data in the corresponding register. The timing is the same for clocking external data into a 16-bit segment of BR, transferring 140 bits from BR to IPR or CØN, and generating a reset pulse. The user must simply keep IØXENA* active for more

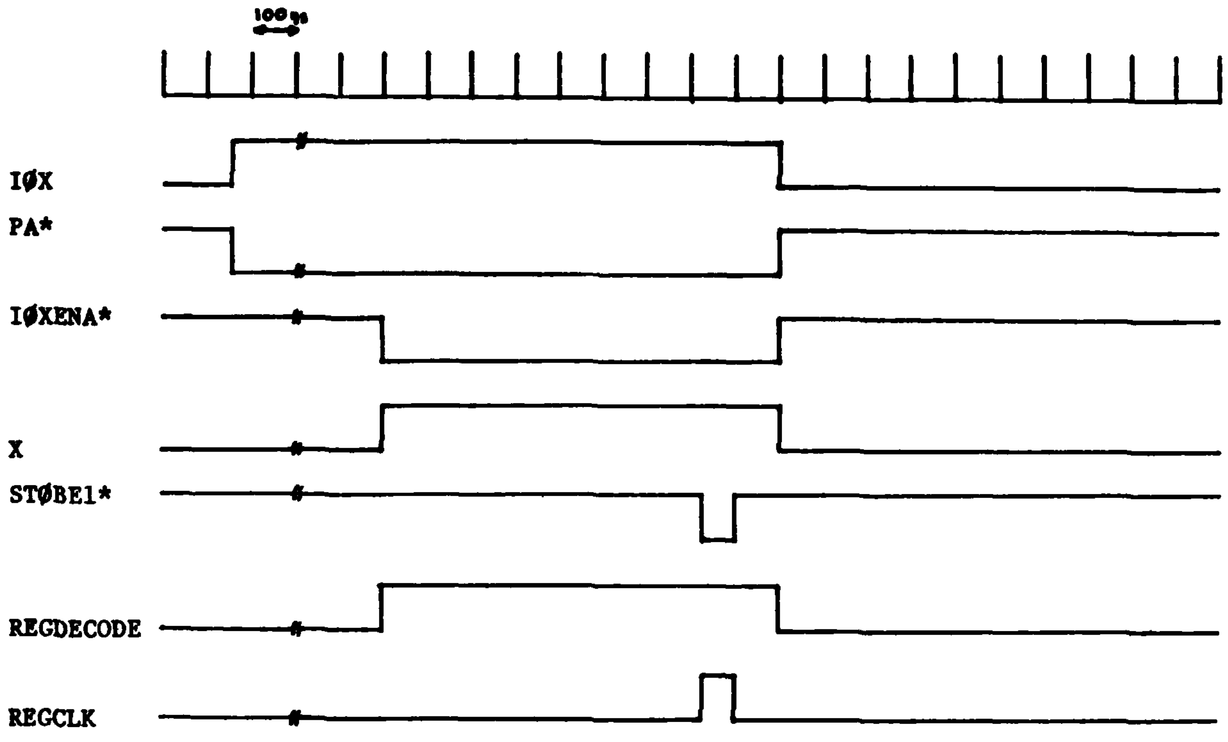


Figure 3.2 Input Timing Diagram

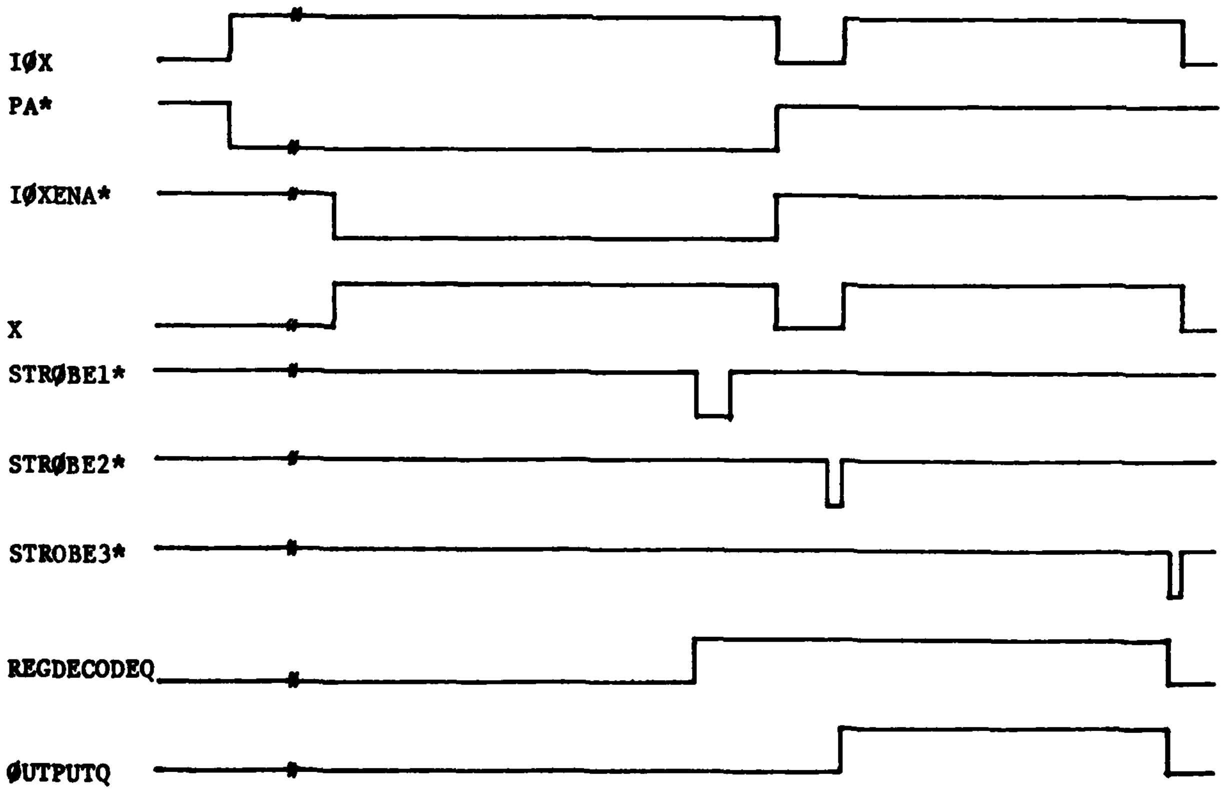


Figure 3.3 Output Timing Diagram

than 900 ns and then data, address and IØXENA* may be made inactive simultaneously.

The timing for output from the MA is similar to the input timing with the addition that the output data must be sampled during a specified time period. Output timing is given in Figure 3.3. The timimg for output is the same as input thru the generation of STRØBE1*. This allows the MA to decode the address of the register being selected for output. However, after STRØBE1* there are additional timing rules. The user must stop driving the enable, address, and data signals before STRØBE2* which occurs at 1140 ns. At this time the MA will begin driving the IØX lines and will continue until STRØBE3* at 1920 ns. Data must be sampled between 1140 and 1920 ns.

Appendix 1 gives sample micro programs for controlling the MA. They are written for a CHI MP32A Macro Processor. The instruction ØUTPUT CLR provides proper timing to IØXENA*, and INPUT samples IØX during the time it is driven by the MA. These micro routines assume that PD (which drives IØX) has been previously set up with the desired data.

3. Circuit Description

This section will first describe each of the major registers within the MA and then describe the control circuits that generate the proper timing and event sequence. Refer to the logic drawings in Appendix 2.

The Buffer Register (BR) is the only reigster that functions as both an array of 16-bit registers and as a 140-bit register. BR is made of DM8542 4-bit three-state registers. External input data (IØX00 to IØX15) is received by 8T95 three-state drivers whose outputs are connected to the A-Bus of the 8542. The 140 bits are segmented into 16-bit groups, each group having a separate input and separate output enable controls. These controls are direct decodes of the PA* signals. Although the entire 140-bit register must have a common clock to allow loading of 140 data bits from the B-Bus, sixteen bit loading is accomplished by the A-Bus controls. Thus BR is externally read in 16-bit segments.

When data is to be transferred from BR to either the Input or Control Registers the B-Bus output enable control is activated. Data is clocked into these registers at STRØBE1* time. Both the Input and Control Registers (IPR and CØN) are made of SN74LS174 6-bit registers.

The outputs of IPR and CØN may be switched to the board under test and in this case they are buffered from the B-Bus of the 8542 by three-state buffers (SN74367). When data is to be loaded into BR from this side the input control LDBR* activates both the three-state buffers and the B-Bus input of BR. In this case BRCLK is generated from the counter.

The Status Register is made of AM25S18 4-bit three-state registers. Its inputs are directly attached to the external probes. It is clocked at the sample time generated by the counter. Its three-state outputs are enabled onto the X-Bus when its address has been decoded. It is read in the same manner as any of the 16-bit segments of BR.

Timing pulses and control signals are generated by a 16-bit up counter. The basic clock of the counter is supplied by a 50 Mhz oscillator. The outputs of the counter are used in two functionally different ways and divide into two phases. The first phase generates all the timing and control signals; the second phase generates the data sample pulse which may be varied between 30 ns and 327 μs in 5 ns intervals.

The first phase generally begins with the activation of IØXENA* and lasts until STRØBE3*. The only exception to this is when the data sampling pulse is to be generated (see below).

When IØXENA* becomes active it clcoks CNTENAQ which enables the 16-bit counter (AM93S16). CNTENAQ is also wired to the reset input of each stage of the counter to ensure the count begins at zero. The counter is a synchronous counter, but the carryout of a stage is clocked into the next stage with a delay of one clock. This was necessary because the delay time from clock to carryout plus the setup time from carry in to clock is longer than the 20 ns clock period generated by the oscillator. Thus it was necessary to stretch the carryout by two inverters and an ØR gate in order to meet the setup requirements.

The strobe pulses, STRØBE1-3* are generated by simply using an 8-input AND gate with the appropriate inputs. STROBE1* generates all the 140-bit register clocks: CØUNTCLK, IPRCLK, CØNCLK, and BRCLKO. STRØBE2* clocks ØUTENAQ, the enable pulse for MA output. STRØBE3* is generally used to reset control flip flops to their original state so that each time the MA is addressed it begins operation from the same state. It resets ØUTPUTQ, ØUTENAQ, LDBR and CNTENAQ.

The exception to this sequence is when the counter is used to generate the sample pulse, TO. The decode of PA33 (Load Control Register) generates the signal CØNCLK at STRØBE1* time. CØNCLK alters the usual sequence of events. Referring to Figure 3.4 the first events are the activation of STRINHQ* and LDBR*. STRINHQ* disables STROBE2-3*; LDBR* enables the B-Bus input of BR. CØNCLK* is used to enable the parallel load feature of the 93S16 counter. So the contents of CS are loaded into the counter at every DELCLK until CØNCLK* goes inactive. At this trailing edge of CØNCLK the counter begins to count and the Control Register is clocked. Hence the dynamic inputs to the board under test change at the same time the counter begins to count.

When the counter reaches a count of all ones the sample pulse, TO, is generated. TO generates a BRCLK, so the data from the board under test (or the contents of other registers as determined by the switch settings) is loaded into BR. TO also clocks data into the Status Register. Additionally TO resets STRINHQ*, so that the counter now begins counting again from zero. Therefore, the generation of the STRØBE1-3* pulses proceeds as previously described with STRØBE3* being the only important event. STRØBE3* resets any remaining active pulses so that the MA is returned to its idle state.

The implementation of selectable 5 ns delays is accomplished by using a delay line with 2-1/2 ns delay taps. Delays of 5, 10, 15, and 20 ns are used to provide four selectable clocks. DELO is selected at all times except when LDBR is active. When LDBR is active the delay is selected by the two least significant bits of CS.

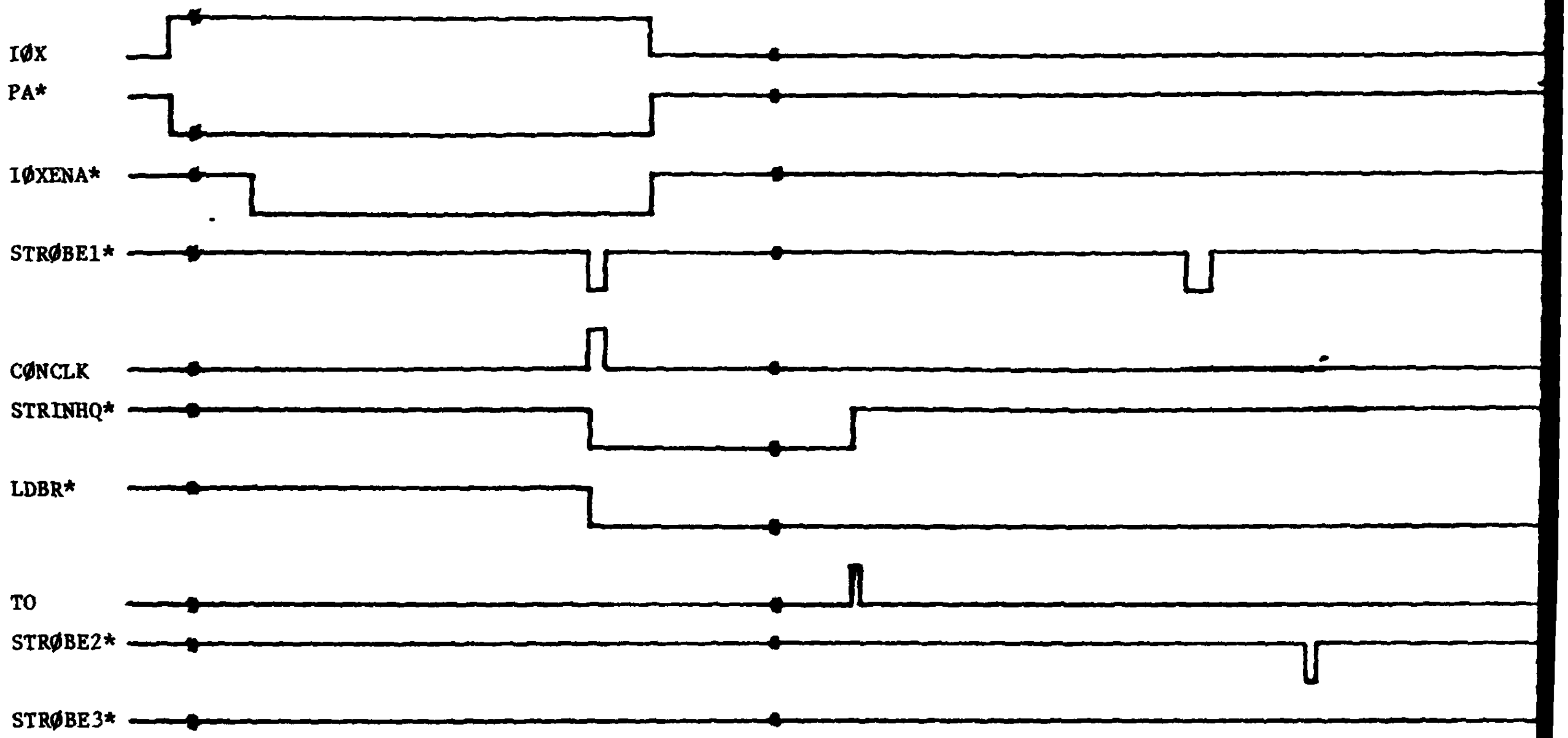


Figure 3.4 Test Sequence Timing

APPENDIX A: SAMPLE OUTPUTS FROM THE STRUCTURED DESIGN SYSTEM

```
DOCUMENT connAPU          PAGE      1   LINE      1
*connAPU
P               *Part name
1               TSel
2               TCLK
3               XMSel
4               XMClr
5               Ø100
6               Ø101
7               Ø102
8               Ø103
9               Ø104
10              Ø105
11              Ø106
12              Ø107
13              Ø108
14              Ø109
15              Ø110
16              Ø111
17              GASel
18              GAClr
19              GC0
20              GC1
21              GC2
22              GBSel
23              GBClr
24              UCLK
25              USel
26              MPCLK
27              MXCLK
28              MYCLK
29              MPRND
30              MXSel
31              MXClr
32              MYSel
33              MYClr
34              CLK
35              YMSel
36              YMClr
37              Ø200
38              Ø201
39              Ø202
40              Ø203
41              Ø204
42              Ø205
43              Ø206
44              Ø207
45              Ø208
46              Ø209
47              Ø210
48              Ø211
49              HASel
50              HAClr
51              HC0
52              HC1
53              HC2
54              HBSel
55              HBClr
56              VCLK
57              VSel
58              NC
```

```
DOCUMENT connAPU          PAGE     1   LINE    61
59          G11
60          GCIN
61          HCRY
62          GND
63          GND
64
65          I100
66          I101
67          I102
68          I103
69          I104
70          I105
71          I106
72          I107
73          I200
74          I201
75          I202
76          I203
77          I204
78          I205
79          I206
80          I207
81          I208
82          I209
83          I210
84          I211
85          I300
86          I301
87          I302
88          I303
89          I304
90          I305
91          I306
92          I307
93          I308
94          I309
95          I310
96          I311
97          M0
98          M1
99          M2
100         M3
101         M4
102         HB0
103         HB1
104         HA0
105         HA1
106         HØP0
107         HØP1
108         HØP2
109         GB0
110         GB1
111         GA0
112         GA1
113         GØP0
114         GØP1
115         GØP2
116         U0
117         V0
118
```

```
DOCUMENT connAPU          PAGE      1   LINE     121
119       NC
120
121       VCC
122       VCC
123
124
125
126
127
128
$
```

```
DØCUMENT menuAPU            PAGE       1   LINE       1
'menuAPU
'Submod.   Physical  Display
'Name      Location  Location
M          J19       0,0
MX1        J10       0,0
MX3        N10       30,25          MX2       L10        0,40
MY1        J1        0,0
MY3        N1        30,25          MY2       L1         0,40
T1         C10       0,0
F1         A10       30,0           T2        G10        0,40
G1         C19       0,0            F2        E10        30,40
G3         C41       30,25          G2        C30        0,40
GA1        A21       0,0
GA3        A43       30,25          GA2       A32        0,40
GB1        E21       0,0
GB3        F43       30,25          GB2       E32        0,40
H1         R19       0,0
H3         R41       30,25          H2        R30        0,40
HA1        N21       0,0
HA3        N43       30,25          HA2       N32        0,40
HB1        T21       0,0
HB3        T43       30,25          HB2       T32        0,40
U1         G21       0,0
U3         G43       30,25          U2        G32        0,40
V1         V21       0,0
V3         V43       30,25          V2        V32        0,40
XM1        C1        0,0
XM3        G1        30,25          XM2       E1         0,40
YM1        R1        0,0
YM3        V1        30,25          YM2       T1         0,40
DCD4       A1        0,0
DCD6       T10       30,0           DCD5      R10        0,40
resistAPU  G42       0,0            DCD7      V10        30,40
$
```

```
*M
MPY12AJ    64 3*Part name
I   X4     1  MX07    J19
I   X5     2  MX06    J20
I   X6     3  MX05    J21
I   X7     4  MX04    J22
I   X8     5  MX03    J23
I   X9     6  MX02    J24
I   X10    7  MX01    J25
I   X11    8  MX00    J26
Ø   PR22   9  MPL00   J27
Ø   PR21   10 MPL01   J28
Ø   PR20   11 MPL02   J29
Ø   PR19   12 MPL03   J30
Ø   PR18   13 MPL04   J31
Ø   PR17   14 MPL05   J32
Ø   PR16   15 MPL06   J33
Ø   PR15   16 MPL07   J34
Ø   PR14   17 MPL08   J35
Ø   PR13   18 MPL09   J36
Ø   PR12   19 MPL10   J37
Ø   SGNL   20 ØPEN    J38
C   TRIL   21 TRUE    J39
C   TRIM   22 TRUE    J40
G   GND    23 GND     J41
G   GND    24 GND     J42
G   GND    25 GND     J43
G   GND    26 GND     J44
C   CLKL   27 MPCLK   J45
C   CLKM   28 MPCLK   J46
Ø   PR11   29 MPF00   J47
Ø   PR10   30 MPF01   J48
Ø   PR9    31 MPF02   J49
Ø   PR8    32 MPF03   J50
Ø   PR7    33 MPF04   L50
Ø   PR6    34 MPF05   L49
Ø   PR5    35 MPF06   L48
Ø   PR4    36 MPF07   L47
Ø   PR3    37 MPF08   L46
Ø   PR2    38 MPF09   L45
Ø   PR1    39 MPF10   L44
Ø   SGNM   40 MPF11   L43
    NC     41 ØPEN    L42
I   YSGN   42 MY11    L41
I   Y1     43 MY10    L40
I   Y2     44 MY09    L39
I   Y3     45 MY08    L38
I   Y4     46 MY07    L37
I   Y5     47 MY06    L36
V   VCC    48 VCC     L35
V   VCC    49 VCC     L34
V   VCC    50 VCC     L33
I   Y6     51 MY05    L32
I   Y7     52 MY04    L31
I   Y8     53 MY03    L30
```

```
*MX1
25LS157     16 2*Part  name
C  Sel       1 MXSel   J10
I  1A        2 I208    J11
I  1B        3 I308    J12
Ø  1Y        4 MX08    J13
I  2A        5 I209    J14
I  2B        6 I309    J15
Ø  2Y        7 MX09    J16
G  GND       8 GND     J17
Ø  3Y        9 MX10    K17
I  3B       10 I310    K16
I  3A       11 I210    K15
Ø  4Y       12 MX11    K14
I  4B       13 I311    K13
I  4A       14 I211    K12
C  STRØBE   15 MXClr   K11
V  VCC      16 VCC     K10
```

```
*MX2
25LS157     16 2*Part  name
C  Sel       1 MXSel   L10
I  1A        2 I204    L11
I  1B        3 I304    L12
Ø  1Y        4 MX04    L13
I  2A        5 I205    L14
I  2B        6 I305    L15
Ø  2Y        7 MX05    L16
G  GND       8 GND     L17
Ø  3Y        9 MX06    M17
I  3B       10 I306    M16
I  3A       11 I206    M15
Ø  4Y       12 MX07    M14
I  4B       13 I307    M13
I  4A       14 I207    M12
C  STRØBE   15 MXClr   M11
V  VCC      16 VCC     M10
```

```
*MX3
25LS157     16 2*Part  name
C  Sel       1 MXSel   N10
I  1A        2 I200    N11
I  1B        3 I300    N12
Ø  1Y        4 MX00    N13
I  2A        5 I201    N14
I  2B        6 I301    N15
Ø  2Y        7 MX01    N16
G  GND       8 GND     N17
Ø  3Y        9 MX02    P17
I  3B       10 I302    P16
I  3A       11 I202    P15
Ø  4Y       12 MX03    P14
I  4B       13 I303    P13
I  4A       14 I203    P12
C  STRØBE   15 MXClr   P11
V  VCC      16 VCC     P10
```

IO = Selectable In Ø

C = ~~CLOCK~~ CONTROL

I = INPUT

Ø = OUTPUT

G = GROUND

V = Voltage

```
*MY1
25LS157      16 2*Part  name
C  Sel       1  MYSel   J1
I   1A       2  I208    J2
I   1B       3  I308    J3
Ø   1Y       4  MY08    J4
I   2A       5  I209    J5
I   2B       6  I309    J6
Ø   2Y       7  MY09    J7
G  GND       8  GND     J8
Ø   3Y       9  MY10    K8
I   3B       10 I310    K7
I   3A       11 I210    K6
Ø   4Y       12 MY11    K5
I   4B       13 I311    K4
I   4A       14 I211    K3
C  STRØBE    15 MYClr   K2
V  VCC       16 VCC     K1
```

```
*MY2
25LS157      16 2*Part  name
C  Sel       1  MYSel   L1
I   1A       2  I204    L2
I   1B       3  I304    L3
Ø   1Y       4  MY04    L4
I   2A       5  I205    L5
I   2B       6  I305    L6
Ø   2Y       7  MY05    L7
G  GND       8  GND     L8
Ø   3Y       9  MY06    M8
I   3B       10 I306    M7
I   3A       11 I206    M6
Ø   4Y       12 MY07    M5
I   4B       13 I307    M4
I   4A       14 I207    M3
C  STRØBE    15 MYClr   M2
V  VCC       16 VCC     M1
```

```
*MY3
25LS157      16 2*Part  name
C  Sel       1  MYSel   N1
I   1A       2  I200    N2
I   1B       3  I300    N3
Ø   1Y       4  MY00    N4
I   2A       5  I201    N5
I   2B       6  I301    N6
Ø   2Y       7  MY01    N7
G  GND       8  GND     N8
Ø   3Y       9  MY02    P8
I   3B       10 I302    P7
I   3A       11 I202    P6
Ø   4Y       12 MY03    P5
I   4B       13 I303    P4
I   4A       14 I203    P3
C  STRØBE    15 MYClr   P2
V  VCC       16 VCC     P1
```

```
*T1
25LS09      16 2*Part name
C  S        1  TSel   C10
Ø  Q0       2  T04    C11
I  D0A      3  F04    C12
I  D0B      4  I104   C13
I  D1B      5  I105   C14
I  D1A      6  F05    C15
Ø  Q1       7  T05    C16
G  GND      8  GND    C17
C  CP       9  TCLK   D17
Ø  Q2       10 T06    D16
I  D2A      11 F06    D15
I  D2B      12 I106   D14
I  D3B      13 I107   D13
I  D3A      14 F07    D12
Ø  Q3       15 T07    D11
V  VCC      16 VCC    D10
```

```
*T2
25LS09      16 2*Part name
C  S        1  TSel   G10
Ø  Q0       2  T00    G11
I  D0A      3  F00    G12
I  D0B      4  I100   G13
I  D1B      5  I101   G14
I  D1A      6  F01    G15
Ø  Q1       7  T01    G16
G  GND      8  GND    G17
C  CP       9  TCLK   H17
Ø  Q2       10 T02    H16
I  D2A      11 F02    H15
I  D2B      12 I102   H14
I  D3B      13 I103   H13
I  D3A      14 F03    H12
Ø  Q3       15 T03    H11
V  VCC      16 VCC    H10
```

```
*F1
25LS2517    20 2
I  A1       1  GA11   A10
I  B1       2  T05    A11
I  A0       3  GA11   A12
I  B0       4  T04    A13
C  S0       5  GC0    A14
C  S1       6  GC1    A15
C  S2       7  GC2    A16
Ø  F0       8  F04    A17
Ø  F1       9  F05    A18
G  GND      10 GND    A19
Ø  F2       11 F06    B19
Ø  F3       12 F07    B18
Ø  ØVR      13 ØPEN   B17
Ø  CØ       14 ØPEN   B16
I  Cn       15 F2CØ   B15
I  B3       16 T07    B14
I  A3       17 GA11   B13
I  B2       18 T06    B12
I  A2       19 GA11   B11
V  VCC      20 VCC    B10
```

```
*F2
25LS2517    20 2*Part name
I  A1       1  GA11   E10
I  B1       2  T01    E11
I  A0       3  GA11   E12
I  B0       4  T00    E13
C  S0       5  GC0    E14
C  S1       6  GC1    E15
C  S2       7  GC2    E16
Ø  F0       8  F00    E17
Ø  F1       9  F01    E18
G  GND      10 GND    E19
Ø  F2       11 F02    F19
Ø  F3       12 F03    F18
Ø  ØVR      13 ØPEN   F17
Ø  CØ       14 F2CØ   F16
I  Cn       15 G1CØ   F15
I  B3       16 T03    F14
I  A3       17 GA11   F13
I  B2       18 T02    F12
I  A2       19 GA11   F11
V  VCC      20 VCC    F10
```

```
*G1
25LS2517    20 2*Part name
I  A1       1  GA09   C19
I  B1       2  GB09   C20
I  A0       3  GA08   C21
I  B0       4  GB08   C22
C  S0       5  GC0    C23
C  S1       6  GC1    C24
C  S2       7  GC2    C25
Ø  F0       8  G08    C26
Ø  F1       9  G09    C27
G  GND      10 GND    C28
Ø  F2       11 G10    D28
Ø  F3       12 G11    D27
Ø  ØVR      13 ØPEN   D26
Ø  CØ       14 G1CØ   D25
I  Cn       15 G2CØ   D24
I  B3       16 GB11   D23
I  A3       17 GA11   D22
I  B2       18 GB10   D21
I  A2       19 GA10   D20
V  VCC      20 VCC    D19
```

```
*G2
25LS2517    20 2*Part name
I  A1       1  GA05   C30
I  B1       2  GB05   C31
I  A0       3  GA04   C32
I  B0       4  GB04   C33
C  S0       5  GC0    C34
C  S1       6  GC1    C35
C  S2       7  GC2    C36
Ø  F0       8  G04    C37
Ø  F1       9  G05    C38
G  GND      10 GND    C39
Ø  F2       11 G06    D39
Ø  F3       12 G07    D38
Ø  ØVR      13 ØPEN   D37
Ø  CØ       14 G2CØ   D36
I  Cn       15 G3CØ   D35
I  B3       16 GB07   D34
I  A3       17 GA07   D33
I  B2       18 GB06   D32
I  A2       19 GA06   D31
V  VCC      20 VCC    D30
```

```
*G3
25LS2517    20 2*Part name
I  A1       1  GA01   C41
I  B1       2  GB01   C42
I  A0       3  GA00   C43
I  B0       4  GB00   C44
C  S0       5  GC0    C45
C  S1       6  GC1    C46
C  S2       7  GC2    C47
Ø  F0       8  G00    C48
Ø  F1       9  G01    C49
G  GND      10 GND    C50
Ø  F2       11 G02    D50
Ø  F3       12 G03    D49
Ø  ØVR      13 ØPEN   D48
Ø  CØ       14 G3CØ   D47
I  Cn       15 GCIN   D46
I  B3       16 GB03   D45
I  A3       17 GA03   D44
I  B2       18 GB02   D43
I  A2       19 GA02   D42
V  VCC      20 VCC    D41
```

```
*GA1                                        *GA2
25LS157    16 2*Part  name                  25LS157    16 2*Part  name
C  Sel     1  GASel   A21                   C  Sel     1  GASel   A32
I  1A      2  MPF09   A22                   I  1A      2  MPF05   A33
I  1B      3  I208    A23                   I  1B      3  I204    A34
Ø  1Y      4  GA08    A24                   Ø  1Y      4  GA04    A35
I  2A      5  MPF10   A25                   I  2A      5  MPF06   A36
I  2B      6  I209    A26                   I  2B      6  I205    A37
Ø  2Y      7  GA09    A27                   Ø  2Y      7  GA05    A38
G  GND     8  GND     A28                   G  GND     8  GND     A39
Ø  3Y      9  GA10    B28                   Ø  3Y      9  GA06    B39
I  3B      10 I210    B27                   I  3B      10 I206    B38
I  3A      11 MPF11   B26                   I  3A      11 MPF07   B37
Ø  4Y      12 GA11    B25                   Ø  4Y      12 GA07    B36
I  4B      13 I211    B24                   I  4B      13 I207    B35
I  4A      14 MPF11   B23                   I  4A      14 MPF08   B34
C  STRØBE  15 GAClr   B22                   C  STRØBE  15 GAClr   B33
V  VCC     16 VCC     B21                   V  VCC     16 VCC     B32
```

```
*GA3
25LS157    16 2*Part  name
C  Sel     1  GASel   A43
I  1A      2  MPF01   A44
I  1B      3  I200    A45
Ø  1Y      4  GA00    A46
I  2A      5  MPF02   A47
I  2B      6  I201    A48
Ø  2Y      7  GA01    A49
G  GND     8  GND     A50
Ø  3Y      9  GA02    B50
I  3B      10 I202    B49
I  3A      11 MPF03   B48
Ø  4Y      12 GA03    B47
I  4B      13 I203    B46
I  4A      14 MPF04   B45
C  STRØBE  15 GAClr   B44
V  VCC     16 VCC     B43
```

```
'GB1
25LS157      16 2'Part  name
C  Sel        1  GBSel   E21
I   1A        2  MPF08   E22
I   1B        3  U08     E23
Ø   1Y        4  GB08    E24
I   2A        5  MPF09   E25
I   2B        6  U09     E26
Ø   2Y        7  GB09    E27
G  GND        8  GND     E28
Ø   3Y        9  GB10    F28
I   3B       10  U10     F27
I   3A       11  MPF10   F26
Ø   4Y       12  GB11    F25
I   4B       13  U11     F24
I   4A       14  MPF11   F23
C  STRØBE    15  GBClr   F22
V  VCC       16  VCC     F21
```

```
'GB2
25LS157      16 2'Part  name
C  Sel        1  GBSel   E32
I   1A        2  MPF04   E33
I   1B        3  U04     E34
Ø   1Y        4  GB04    E35
I   2A        5  MPF05   E36
I   2B        6  U05     E37
Ø   2Y        7  GB05    E38
G  GND        8  GND     E39
Ø   3Y        9  GB06    F39
I   3B       10  U06     F38
I   3A       11  MPF06   F37
Ø   4Y       12  GB07    F36
I   4B       13  U07     F35
I   4A       14  MPF07   F34
C  STRØBE    15  GBClr   F33
V  VCC       16  VCC     F32
```

```
'GB3
25LS157      16 2'Part  name
C  Sel        1  GBSel   E43
I   1A        2  MPF00   E44
I   1B        3  U00     E45
Ø   1Y        4  GB00    E46
I   2A        5  MPF01   E47
I   2B        6  U01     E48
Ø   2Y        7  GB01    E49
G  GND        8  GND     E50
Ø   3Y        9  GB02    F50
I   3B       10  U02     F49
I   3A       11  MPF02   F48
Ø   4Y       12  GB03    F47
I   4B       13  U03     F46
I   4A       14  MPF03   F45
C  STRØBE    15  GBClr   F44
V  VCC       16  VCC     F43
```

```
*H1                                     *H2
25LS2517   20 2*Part name               25LS2517   20 2*Part name
I  A1      1  HA09   R19                I  A1      1  HA05   R30
I  B1      2  HB09   R20                I  B1      2  HB05   R31
I  A0      3  HA08   R21                I  A0      3  HA04   R32
I  B0      4  HB08   R22                I  B0      4  HB04   R33
C  S0      5  HC0    R23                C  S0      5  HC0    R34
C  S1      6  HC1    R24                C  S1      6  HC1    R35
C  S2      7  HC2    R25                C  S2      7  HC2    R36
Ø  F0      8  H08    R26                Ø  F0      8  H04    R37
Ø  F1      9  H09    R27                Ø  F1      9  H05    R38
G  GND     10 GND    R28                G  GND     10 GND    R39
Ø  F2      11 H10    S28                Ø  F2      11 H06    S39
Ø  F3      12 H11    S27                Ø  F3      12 H07    S38
Ø  ØVR     13 ØPEN   S26                Ø  ØVR     13 ØPEN   S37
Ø  CØ      14 HCRY   S25                Ø  CØ      14 H2CØ   S36
I  Cn      15 H2CØ   S24                I  Cn      15 H3CØ   S35
I  B3      16 HB11   S23                I  B3      16 HB07   S34
I  A3      17 HA11   S22                I  A3      17 HA07   S33
I  B2      18 HB10   S21                I  B2      18 HB06   S32
I  A2      19 HA10   S20                I  A2      19 HA06   S31
V  VCC     20 VCC    S19                V  VCC     20 VCC    S30
```

```
*H3
25LS2517   20 2*Part name
I  A1      1  HA01   R41
I  B1      2  HB01   R42
I  A0      3  HA00   R43
I  B0      4  HB00   R44
C  S0      5  HC0    R45
C  S1      6  HC1    R46
C  S2      7  HC2    R47
Ø  F0      8  H00    R48
Ø  F1      9  H01    R49
G  GND     10 GND    R50
Ø  F2      11 H02    S50
Ø  F3      12 H03    S49
Ø  ØVR     13 ØPEN   S48
Ø  CØ      14 H3CØ   S47
I  Cn      15 GND    S46
I  B3      16 HB03   S45
I  A3      17 HA03   S44
I  B2      18 HB02   S43
I  A2      19 HA02   S42
V  VCC     20 VCC    S41
```

```
"HA1
25LS157      16 2"Part name
C   Sel      1  HASel  N21
I   1A       2  MPL08  N22
I   1B       3  I308   N23
Ø   1Y       4  HA08   N24
I   2A       5  MPL09  N25
I   2B       6  I309   N26
Ø   2Y       7  HA09   N27
G   GND      8  GND    N28
Ø   3Y       9  HA10   P28
I   3B       10 I310   P27
I   3A       10 MPL10  P26
Ø   4Y       12 HA11   P25
I   4B       13 I311   P24
I   4A       14 MPF00  P23
C   STRØBE   15 HAClr  P22
V   VCC      16 VCC    P21
```

```
"HA2
25LS157      16 2"Part name
C   Sel      1  HASel  N32
I   1A       2  MPL04  N33
I   1B       3  I304   N34
Ø   1Y       4  HA04   N35
I   2A       5  MPL05  N36
I   2B       6  I305   N37
Ø   2Y       7  HA05   N38
G   GND      8  GND    N39
Ø   3Y       9  HA06   P39
I   3B       10 I306   P38
I   3A       10 MPL06  P37
Ø   4Y       12 HA07   P36
I   4B       13 I307   P35
I   4A       14 MPL07  P34
C   STRØBE   15 HAClr  P33
V   VCC      16 VCC    P32
```

```
"HA3
25LS157      16 2"Part name
C   Sel      1  HASel  N43
I   1A       2  MPL00  N44
I   1B       3  I300   N45
Ø   1Y       4  HA00   N46
I   2A       5  MPL01  N47
I   2B       6  I301   N48
Ø   2Y       7  HA01   N49
G   GND      8  GND    N50
Ø   3Y       9  HA02   P50
I   3B       10 I302   P49
I   3A       10 MPL02  P48
Ø   4Y       12 HA03   P47
I   4B       13 I303   P46
I   4A       14 MPL03  P45
C   STRØBE   15 HAClr  P44
V   VCC      16 VCC    P43
```

```
"HB1
25LS157    16 2"Part name
C  Sel      1 HBSel  T21
I  1A       2 MPF08  T22
I  1B       3 V08    T23
Ø  1Y       4 HB08   T24
I  2A       5 MPF09  T25
I  2B       6 V09    T26
Ø  2Y       7 HB09   T27
G  GND      8 GND    T28
Ø  3Y       9 HB10   U28
I  3B      10 V10    U27
I  3A      10 MPF10  U26
Ø  4Y      12 HB11   U25
I  4B      13 V11    U24
I  4A      14 MPF11  U23
C  STRØBE  15 HBClr  U22
V  VCC     16 VCC    U21
```

```
"HB2
25LS157    16 2"Part name
C  Sel      1 HBSel  T32
I  1A       2 MPF04  T33
I  1B       3 V04    T34
Ø  1Y       4 HB04   T35
I  2A       5 MPF05  T36
I  2B       6 V05    T37
Ø  2Y       7 HB05   T38
G  GND      8 GND    T39
Ø  3Y       9 HB06   U39
I  3B      10 V06    U38
I  3A      10 MPF06  U37
Ø  4Y      12 HB07   U36
I  4B      13 V07    U35
I  4A      14 MPF07  U34
C  STRØBE  15 HBClr  U33
V  VCC     16 VCC    U32
```

```
"HB3
25LS157    16 2"Part name
C  Sel      1 HBSel  T43
I  1A       2 MPF00  T44
I  1B       3 V00    T45
Ø  1Y       4 HB00   T46
I  2A       5 MPF01  T47
I  2B       6 V01    T48
Ø  2Y       7 HB01   T49
G  GND      8 GND    T50
Ø  3Y       9 HB02   U50
I  3B      10 V02    U49
I  3A      10 MPF02  U48
Ø  4Y      12 HB03   U47
I  4B      13 V03    U46
I  4A      14 MPF03  U45
C  STRØBE  15 HBClr  U44
V  VCC     16 VCC    U43
```

```
*U1                                   *U2
25LS09     16 2*Part name             25LS09     16 2*Part name
C   S      1  USel   G21              C   S      1  USel   G32
Ø   Q0     2  U08    G22              Ø   Q0     2  U04    G33
I   D0A    3  G08    G23              I   D0A    3  G04    G34
I   D0B    4  T04    G24              I   D0B    4  T00    G35
I   D1B    5  T05    G25              I   D1B    5  T01    G36
I   D1A    6  G09    G26              I   D1A    6  G05    G37
Ø   Q1     7  U09    G27              Ø   Q1     7  U05    G38
G   GND    8  GND    G28              G   GND    8  GND    G39
C   CP     9  UCLK   H28              C   CP     9  UCLK   H39
Ø   Q2     10 U10    H27              Ø   Q2     10 U06    H38
I   D2A    11 G10    H26              I   D2A    11 G06    H37
I   D2B    12 T06    H25              I   D2B    12 T02    H36
I   D3B    13 T07    H24              I   D3B    13 T03    H35
I   D3A    14 G11    H23              I   D3A    14 G07    H34
Ø   Q3     15 U11    H22              Ø   Q3     15 U07    H33
V   VCC    16 VCC    H21              V   VCC    16 VCC    H32
```

```
*U3
25LS09     16 2*Part name
C   S      1  USel   G43
Ø   Q0     2  U00    G44
I   D0A    3  G00    G45
I   D0B    4  U08    G46
I   D1B    5  U09    G47
I   D1A    6  G01    G48
Ø   Q1     7  U01    G49
G   GND    8  GND    G50
C   CP     9  UCLK   H50
Ø   Q2     10 U02    H49
I   D2A    11 G02    H48
I   D2B    12 U10    H47
I   D3B    13 U11    H46
I   D3A    14 G03    H45
Ø   Q3     15 U03    H44
V   VCC    16 VCC    H43
```

```
'V1                                    'V2
25LS09     16 2'Part  name             25LS09     16 2'Part  name
C  S       1  VSel    V21              C  S       1  VSel    V32
O  Q0      2  V08     V22              O  Q0      2  V04     V33
I  D0A     3  H08     V23              I  D0A     3  H04     V34
I  D0B     4  U04     V24              I  D0B     4  U00     V35
I  D1B     5  U05     V25              I  D1B     5  U01     V36
I  D1A     6  H09     V26              I  D1A     6  H05     V37
O  Q1      7  V09     V27              O  Q1      7  V05     V38
G  GND     8  GND     V28              G  GND     8  GND     V39
C  CP      9  VCLK    W28              C  CP      9  VCLK    W39
O  Q2      10 V10     W27              O  Q2      10 V06     W38
I  D2A     11 H10     W26              I  D2A     11 H06     W37
I  D2B     12 U06     W25              I  D2B     12 U02     W36
I  D3B     13 U07     W24              I  D3B     13 U03     W35
I  D3A     14 H11     W23              I  D3A     14 H07     W34
O  Q3      15 V11     W22              O  Q3      15 V07     W33
V  VCC     16 VCC     W21              V  VCC     16 VCC     W32
```

```
'V3
25LS09     16 2'Part  name
C  S       1  VSel    V43
O  Q0      2  V00     V44
I  D0A     3  H00     V45
I  D0B     4  V08     V46
I  D1B     5  V09     V47
I  D1A     6  H01     V48
O  Q1      7  V01     V49
G  GND     8  GND     V50
C  CP      9  VCLK    W50
O  Q2      10 V02     W49
I  D2A     11 H02     W48
I  D2B     12 V10     W47
I  D3B     13 V11     W46
I  D3A     14 H03     W45
O  Q3      15 V03     W44
V  VCC     16 VCC     W43
```

```
*XM1                                    *XM2
25LS157    16 2*Part  name              25LS157    16 2*Part  name
C   Sel     1 XMSel   C1                C   Sel     1 XMSel   E1
I   1A      2 U08     C2                I   1A      2 U04     E2
I   1B      3 V08     C3                I   1B      3 V04     E3
Ø   1Y      4 Ø108    C4                Ø   1Y      4 Ø104    E4
I   2A      5 U09     C5                I   2A      5 U05     E5
I   2B      6 V09     C6                I   2B      6 V05     E6
Ø   2Y      7 Ø109    C7                Ø   2Y      7 Ø105    E7
G   GND     8 GND     C8                G   GND     8 GND     F8
Ø   3Y      9 Ø110    D8                Ø   3Y      9 Ø106    F8
I   3B     10 V10     D7                I   3B     10 V06     F7
I   3A     11 U10     D6                I   3A     11 U06     F6
Ø   4Y     12 Ø111    D5                Ø   4Y     12 Ø107    F5
I   4B     13 V11     D4                I   4B     13 V07     F4
I   4A     14 U11     D3                I   4A     14 U07     F3
C   STRØBE 15 XMClr   D2                C   STRØBE 15 XMClr   F2
V   VCC    16 VCC     D1                V   VCC    16 VCC     F1
```

```
*XM3
25LS157    16 2*Part  name
C   Sel     1 XMSel   G1
I   1A      2 U00     G2
I   1B      3 V00     G3
Ø   1Y      4 Ø100    G4
I   2A      5 U01     G5
I   2B      6 V01     G6
Ø   2Y      7 Ø101    G7
G   GND     8 GND     G8
Ø   3Y      9 Ø102    H8
I   3B     10 V02     H7
I   3A     11 U02     H6
Ø   4Y     12 Ø103    H5
I   4B     13 V03     H4
I   4A     14 U03     H3
C   STRØBE 15 XMClr   H2
V   VCC    16 VCC     H1
```

```
*YM1
25LS157    16 2*Part  name
C  Sel      1 YMSel   R1
I  1A       2 U08     R2
I  1B       3 V08     R3
O  1Y       4 O208    R4
I  2A       5 U09     R5
I  2B       6 V09     R6
O  2Y       7 O209    R7
G  GND      8 GND     R8
O  3Y       9 O210    S8
I  3B      10 V10     S7
I  3A      11 U10     S6
O  4Y      12 O211    S5
I  4B      13 V11     S4
I  4A      14 U11     S3
C  STROBE  15 YMClr   S2
V  VCC     16 VCC     S1
```

```
*YM2
25LS157    16 2*Part  name
C  Sel      1 YMSel   T1
I  1A       2 U04     T2
I  1B       3 V04     T3
O  1Y       4 O204    T4
I  2A       5 U05     T5
I  2B       6 V05     T6
O  2Y       7 O205    T7
G  GND      8 GND     T8
O  3Y       9 O206    U8
I  3B      10 V06     U7
I  3A      11 U06     U6
O  4Y      12 O207    U5
I  4B      13 V07     U4
I  4A      14 U07     U3
C  STROBE  15 YMClr   U2
V  VCC     16 VCC     U1
```

```
*YM3
25LS157    16 2*Part  name
C  Sel      1 YMSel   V1
I  1A       2 U00     V2
I  1B       3 V00     V3
O  1Y       4 O200    V4
I  2A       5 U01     V5
I  2B       6 V01     V6
O  2Y       7 O201    V7
G  GND      8 GND     V8
O  3Y       9 O202    W8
I  3B      10 V02     W7
I  3A      11 U02     W6
O  4Y      12 O203    W5
I  4B      13 V03     W4
I  4A      14 U03     W3
C  STROBE  15 YMClr   W2
V  VCC     16 VCC     W1
```

```
*DCD4
*Type      Contacts  Width
25LS07     16 2
C  E       1  GND    A1
Ø  Q0      2  GC2    A2
I  D0      3  GØP2   A3
I  D1      4  U0     A4
Ø  Q1      5  USel   A5
I  D2      6  V0     A6
Ø  Q2      7  VSel   A7
G  GND     8  GND    A8
C  CP      9  CLK    B8
Ø  Q3      10        B7
I  D3      11        B6
Ø  Q4      12        B5
I  D4      13        B4
I  D5      14        B3
Ø  Q5      15        B2
V  VCC     16 VCC    B1
```

```
*DCD5
25LS07     16 2
C  E       1  GND    R10
Ø  Q0      2  GBSel  R11
I  D0      3  GB0    R12
I  D1      4  GB1    R13
Ø  Q1      5  GBClr  R14
I  D2      6  GA0    R15
Ø  Q2      7  GASel  R16
G  GND     8  GND    R17
C  CP      9  CLK    S17
Ø  Q3      10 GAClr  S16
I  D3      11 GA1    S15
Ø  Q4      12 GC0    S14
I  D4      13 GØP0   S13
I  D5      14 GØP1   S12
Ø  Q5      15 GC1    S11
V  VCC     16 VCC    S10
```

```
*DCD6
25LS07     16 2
C  E       1  GND    T10
Ø  Q0      2  HBClr  T11
I  D0      3  HB1    T12
I  D1      4  HA0    T13
Ø  Q1      5  HASel  T14
I  D2      6  HA1    T15
Ø  Q2      7  HAClr  T16
G  GND     8  GND    T17
C  CP      9  CLK    U17
Ø  Q3      10 HC0    U16
I  D3      11 HØP0   U15
Ø  Q4      12 HC1    U14
I  D4      13 HØP1   U13
I  D5      14 HØP2   U12
Ø  Q5      15 HC2    U11
V  VCC     16 VCC    U10
```

```
*DCD7
25LS07     16 2
C  E       1  GND    V10
Ø  Q0      2  MPRND  V11
I  D0      3  M0     V12
I  D1      4  M1     V13
Ø  Q1      5  MXCLK  V14
I  D2      6  M2     V15
Ø  Q2      7  MYCLK  V16
G  GND     8  GND    V17
C  CP      9  CLK    W17
Ø  Q3      10 MXSel  W16
I  D3      11 M3     W15
Ø  Q4      12 MYSel  W14
I  D4      13 M4     W13
I  D5      14 HB0    W12
Ø  Q5      15 HBSel  W11
V  VCC     16 VCC    W10
```

```
*resistAPU
RESISTOR  2  2*Part name
V  VCC    1  VCC    G42
@  True   2  TRUE   H42
```

CLK	34	B8	S17	U17	W17
F00	G12	E17			
F01	G15	E18			
F02	H15	F19			
F03	H12	F18			
F04	C12	A17			
F05	C15	A18			
F06	D15	B19			
F07	D12	B18			
F2CE	B15	F16			
G00	C48	G45			
G01	C49	G48			
G02	D50	H48			
G03	D49	H45			
G04	C37	G34			
G05	C38	G37			
G06	D39	H37			
G07	D38	H34			
G08	C26	G23			
G09	C27	G26			
G10	D28	H26			
G11	59	D27	H23		
G1CE	F15	D25			
G2CE	D24	D36			
G3CE	D35	D47			
GA0	111	R15			

GA00	C43	A46			
GA01	C41	A49			
GA02	D42	B50			
GA03	D44	B47			
GA04	C32	A35			
GA05	C30	A38			
GA06	D31	B39			
GA07	D33	B36			
GA08	C21	A24			
GA09	C19	A27			
GA1	112	S15			
GA10	D20	B28			
GA11	A10 E12	A12 F13	B13 F11	B11 D22	E10 B25
GAClr	18	B22	B33	B44	S16
GASel	17	A21	A32	A43	R16
GB0	109	R12			
GB00	C44	E46			
GB01	C42	E49			
GB02	D43	F50			
GB03	D45	F47			
GB04	C33	E35			
GB05	C31	E38			
GB06	D32	F39			
GB07	D34	F36			
GB08	C22	E24			

GB09	C20	E27			
GB1	110	R13			
GB10	D21	F28			
GB11	D23	F25			
GBClr	23	F22	F33	F44	R14
GBSel	22	E21	E32	E43	R11
GC0	19	A14	E14	C23	C34
	C45	S14			
GC1	20	A15	E15	C24	C35
	C46	S11			
GC2	21	A16	E16	C25	C36
	C47	A2			
GCIN	60	D46			
GND	62	63	J41	J42	J43
	J44	J17	L17	N17	J8
	L8	N8	C17	G17	A19
	E19	C28	C39	C50	A28
	A39	A50	E28	E39	E50
	R28	R39	R50	S46	N28
	N39	N50	T28	T39	T50
	G28	G39	G50	V28	V39
	V50	C8	E8	G8	R8
	T8	V8	A1	A8	R10
	R17	T10	T17	V10	V17
GSP0	113	S13			
GSP1	114	S12			
GSP2	115	A3			
H00	R48	V45			
H01	R49	V48			
H02	S50	W48			
H03	S49	W45			
H04	R37	V34			

H05	R38	V37			
H06	S39	W37			
H07	S38	W34			
H08	R26	V23			
H09	R27	V26			
H10	S28	W26			
H11	S27	W23			
H200	S24	S36			
H300	S35	S47			
HA0	104	T13			
HA00	R43	N46			
HA01	R41	N49			
HA02	S42	P50			
HA03	S44	P47			
HA04	R32	N35			
HA05	R30	N38			
HA06	S31	P39			
HA07	S33	P36			
HA08	R21	N24			
HA09	R19	N27			
HA1	105	T15			
HA10	S20	P28			
HA11	S22	P25			
HAClr	50	P22	P33	P44	T16
HASel	49	N21	N32	N43	T14
HB0	102	W12			

HB00	R44	T46			
HB01	R42	T49			
HB02	S43	U50			
HB03	S45	U47			
HB04	R33	T35			
HB05	R31	T38			
HB06	S32	U39			
HB07	S34	U36			
HB08	R22	T24			
HB09	R20	T27			
HB1	103	T12			
HB10	S21	U28			
HB11	S23	U25			
HBClr	55	U22	U33	U44	T11
HBSel	54	T21	T32	T43	W11
HC0	51	R23	R34	R45	U16
HC1	52	R24	R35	R46	U14
HC2	53	R25	R36	R47	U11
HCRY	61	S25			
HØP0	106	U15			
HØP1	107	U13			
HØP2	108	U12			
I100	65	G13			
I101	66	G14			
I102	67	H14			
I103	68	H13			

I104	69	C13		
I105	70	C14		
I106	71	D14		
I107	72	D13		
I200	73	N11	N2	A45
I201	74	N14	N5	A48
I202	75	P15	P6	B49
I203	76	P12	P3	B46
I204	77	L11	L2	A34
I205	78	L14	L5	A37
I206	79	M15	M6	B38
I207	80	M12	M3	B35
I208	81	J11	J2	A23
I209	82	J14	J5	A26
I210	83	K15	K6	B27
I211	84	K12	K3	B24
I300	85	N12	N3	N45
I301	86	N15	N6	N48
I302	87	P16	P7	P49
I303	88	P13	P4	P46
I304	89	L12	L3	N34
I305	90	L15	L6	N37
I306	91	M16	M7	P38
I307	92	M13	M4	P35
I308	93	J12	J3	N23
I309	94	J15	J6	N26

I310	95	K16	K7	P27	
I311	96	K13	K4	P24	
M0	97	V12			
M1	98	V13			
M2	99	V15			
M3	100	W15			
M4	101	W13			
MPCLK	26	J45	J46		
MPF00	J47	E44	P23	T44	
MPF01	J48	A44	E47	T47	
MPF02	J49	A47	F48	U48	
MPF03	J50	B48	F45	U45	
MPF04	M50	B45	E33	T33	
MPF05	M49	A33	E36	T36	
MPF06	M48	A36	F37	U37	
MPF07	M47	B37	F34	U34	
MPF08	M46	B34	E22	T22	
MPF09	M45	A22	E25	T25	
MPF10	M44	A25	F26	U26	
MPF11	M43	B26	B23	F23	U23
MPL00	J27	N44			
MPL01	J28	N47			
MPL02	J29	P48			
MPL03	J30	P45			
MPL04	J31	N33			
MPL05	J32	N36			

MPL06	J33	P37			
MPL07	J34	P34			
MPL08	J35	N22			
MPL09	J36	N25			
MPL10	J37	P26			
MPRND	29	M25	V11		
MX00	J26	N13			
MX01	J25	N16			
MX02	J24	P17			
MX03	J23	P14			
MX04	J22	L13			
MX05	J21	L16			
MX06	J20	M17			
MX07	J19	M14			
MX08	M19	J13			
MX09	M20	J16			
MX10	M21	K17			
MX11	M22	K14			
MXCLK	27	M23	V14		
MXClr	31	K11	M11	P11	
MXSel	30	J10	L10	N10	W16
MY00	M27	N4			
MY01	M28	N7			
MY02	M29	P8			
MY03	M30	P5			
MY04	M31	L4			

MY05	M32	L7			
MY06	M36	M8			
MY07	M37	M5			
MY08	M38	J4			
MY09	M39	J7			
MY10	M40	K8			
MY11	M41	K5			
MYCLK	28	M24	V16		
MYClr	33	K2	M2	P2	
MYSel	32	J1	L1	N1	W14
NC	58	119			
Ø100	5	G4			
Ø101	6	G7			
Ø102	7	H8			
Ø103	8	H5			
Ø104	9	E4			
Ø105	10	E7			
Ø106	11	F8			
Ø107	12	F5			
Ø108	13	C4			
Ø109	14	C7			
Ø110	15	D8			
Ø111	16	D5			
Ø200	37	V4			
Ø201	38	V7			
Ø202	39	W8			

Q203	40	W5			
Q204	41	T4			
Q205	42	T7			
Q206	43	U8			
Q207	44	U5			
Q208	45	R4			
Q209	46	R7			
Q210	47	S8			
Q211	48	S5			
OPEN	J38 F17 S37	M42 D26 S48	A26 D37	B17 D48	B16 S26
T00	G11	E13	G35		
T01	G16	E11	G36		
T02	H16	F12	H36		
T03	H11	F14	H35		
T04	C11	A13	G24		
T05	C16	A11	G25		
T06	D16	B12	H25		
T07	D11	B14	H24		
TCLK	2	D17	H17		
TRUE	H42	J39	J40		
TSel	1	C10	G10		
U0	116	A4			
U00	E46	G44	V35	G2	V2
U01	E48	G49	V36	G5	V5
U02	F49	H49	W36	H6	W6

U03	F46	H44	W35	H3	W3
U04	E34	G33	V24	E2	T2
U05	E37	G38	V25	E5	T5
U06	F38	H38	W25	F6	U6
U07	F35	H33	W24	F3	U3
U08	E23	G22	G46	C2	R2
U09	E26	G27	G47	C5	R5
U10	F27	H27	H47	D6	S6
U11	F24	H22	H46	D3	S3
UCLK	24	H28	H39	H50	
USel	25	G21	G32	G43	A5
V0	117	A6			
V00	T45	V44	G3	V3	
V01	T48	V49	G6	V6	
V02	U49	W49	H7	W7	
V03	U46	W44	H4	W4	
V04	T34	V33	E3	T3	
V05	T37	V38	E6	T6	
V06	U38	W38	F7	U7	
V07	U35	W33	F4	U4	
V08	T23	V22	V46	C3	R3
V09	T26	V27	V47	C6	R6
V10	U27	W27	W47	D7	S7
V11	U24	W22	W46	D4	S4

```
VCC      G42     121     122     M35     M34
         M33     K10     M10     P10     K1
         M1      P1      D10     H10     B10
         F10     D19     D30     D41     B21
         B32     B43     F21     F32     F43
         S19     S30     S41     P21     P32
         P43     U21     U32     U43     H21
         H32     H43     W21     W32     W43
         D1      F1      H1      S1      U1
         W1      B1      S10     U10     W10

VCLK      56     W28     W39     W50

VSel      57     V21     V32     V43     A7

XMClr     4      D2      F2      H2

XMSel     3      C1      E1      G1

YMClr     36     S2      U2      W2

YMSel     35     R1      T1      V1
DONE
```

APPENDIX B: MICRO PROGRAMS

Micro Programs

```
DOCUMENT  BUFFLD
RA  IA     T M  J   ØP DCBA
                   GLØBAL BUFFLD
 0   C     0 0 60  14 1406  BUFFLD     LØADCD MACRØ 60,L
 2  60     0 2 61   6 0020  B          PA=20
 4  61     7 0 62  14 7123  A          ØUTPUT CLR
 6  62     3 0 63  14 5001             WAIT 4,2
10  63     0 0 64  16 0365             IF ØCT=3 SKIP
12  64     0 0 61   6 0010             INC PA GØTØ A
14  65     0 0  3   6 0000             GØTØ 3
16  66     0 0 60   6 0000  L          GØTØ B
```

```
DOCUMENT  BUFFRD
RA  IA     T M  J   ØP DCBA
                   GLØBAL BUFFRD
 0   C     0 0 60  14 1407  BUFFRD     LØADCD MACRØ 60,10
 2  60     0 2 61   6 0040  B          PA=40
 4  61     7 0 62  14 7123  A          ØUTPUT CLR
 6  62     2 1 63  14 7120             INPUT
10  63     3 0 64   5 0513             MØVE DI+E1+I1+PD T=3
12  64     0 0 65  16 0566             IF ØCT=5 SKIP
14  65     0 0 61   6 0010             INC PA GØTØ A
16  65     0 0  3   6 0000             GØTØ 3
20  67     0 0 60   6 0000             GØTØ B
```

```
DOCUMENT  CNTRØLLD
RA  IA     T M  J   ØP DCBA
                   GLØBAL CNTRØLLD
 0   C     0 0 60  14 1406  CNTRØLLD   LØADCD MACRØ 60,L
 2  60     0 2 61   6 0033  B          PA=33
 4  61     7 0 62  14 7123             ØUTPUT CLR
 6  62     0 2 63   6 0002  A          PA=2
10  63     0 0 61  16 3402             IF S(2)=0 GØTØ A,ØTW C
12  64     0 1 65  16 4500  C          S(PA)=0
14  65     0 0  3   6 0000             GØTØ 3
16  66     0 0 60   6 0000  L          GØTØ B
```

```
DOCUMENT  COUNTLD
RA  IA     T M  J   ØP DCBA
                   GLØBAL COUNTLD
 0   C     0 0 60  14 1403  COUNTLD    LØADCD MACRØ 60,L
 2  60     0 2 61   6 0031  B          PA=31
 4  61     7 0 62  14 7123             ØUTPUT CLR
 6  62     0 0  3   6 0000             GØTØ 3
10  63     0 0 60   6 0000  L          GØTØ B
```

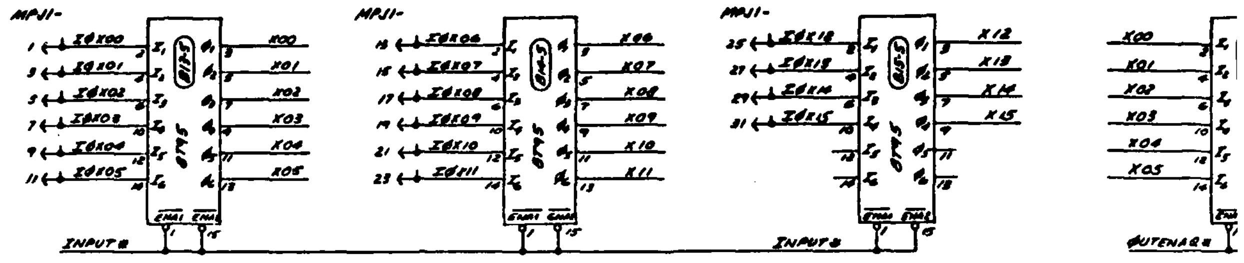
```
DOCUMENT  INPTLD
RA  IA    T M  J   ØP DCBA
                 GLØBAL INPTLD
 0   C    0 0 60  14 1403  INPTLD    LØADCD MACRØ 60,L
 2  60    0 2 61   6 0032  B         PA=32
 4  61    7 0 62  14 7123            ØUTPUT CLR
 6  62    0 0  3   6 0000            GØTØ 3
10  63    0 0 60   6 0000  L         GØTØ B
```

```
DOCUMENT  RESMA
RA  IA    T M  J   ØP DCBA
                 GLØBAL RESMA
 0   C    0 0 60  14 1403  RESMA     LØADCD MACRØ 60,L
 2  60    0 2 61   6 0034  B         PA=34
 4  61    7 0 62  14 7123            ØUTPUT CLR
 6  62    0 0  3   6 0000            GØTØ 3
10  63    0 0 60   6 0000  L         GØTØ B
```

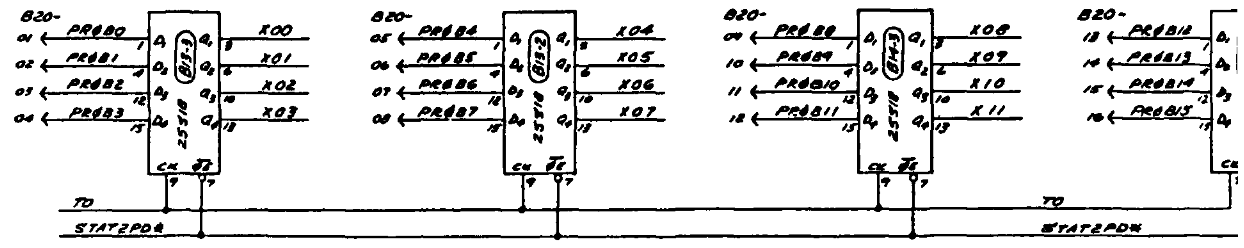
```
DOCUMENT  STATRD
RA  IA    T M  J   ØP DCBA
                 GLØBAL STATRD
 0   C    0 0 60  14 1405  STATRD    LØADCD MACRØ 60,L
 2  60    0 2 61   6 0051  B         PA=51
 4  61    7 0 62  14 7123  A         ØUTPUT CLR
 6  62    2 1 63  14 7120            INPUT
10  63    3 0 64   5 0513            MØVE DI+F1+I1+P1) T=3
12  64    0 0  3   6 0000            GØTØ 3
14  65    0 0 60   6 0000  L         GØTØ B
```

APPENDIX C: MODULE ANALYZER LOGICAL DRAWINGS

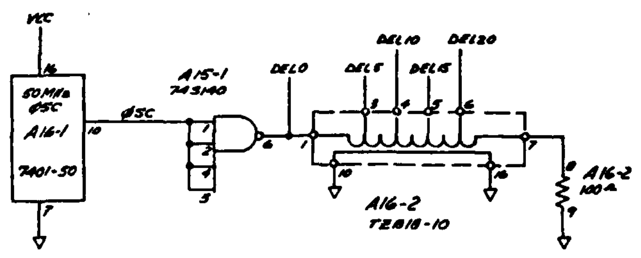
IØX BUSS



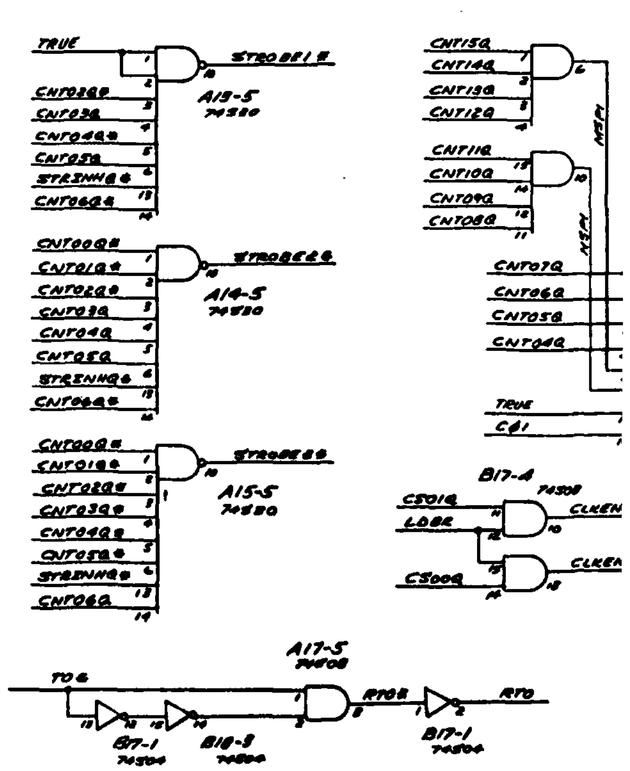
STATUS REGISTER



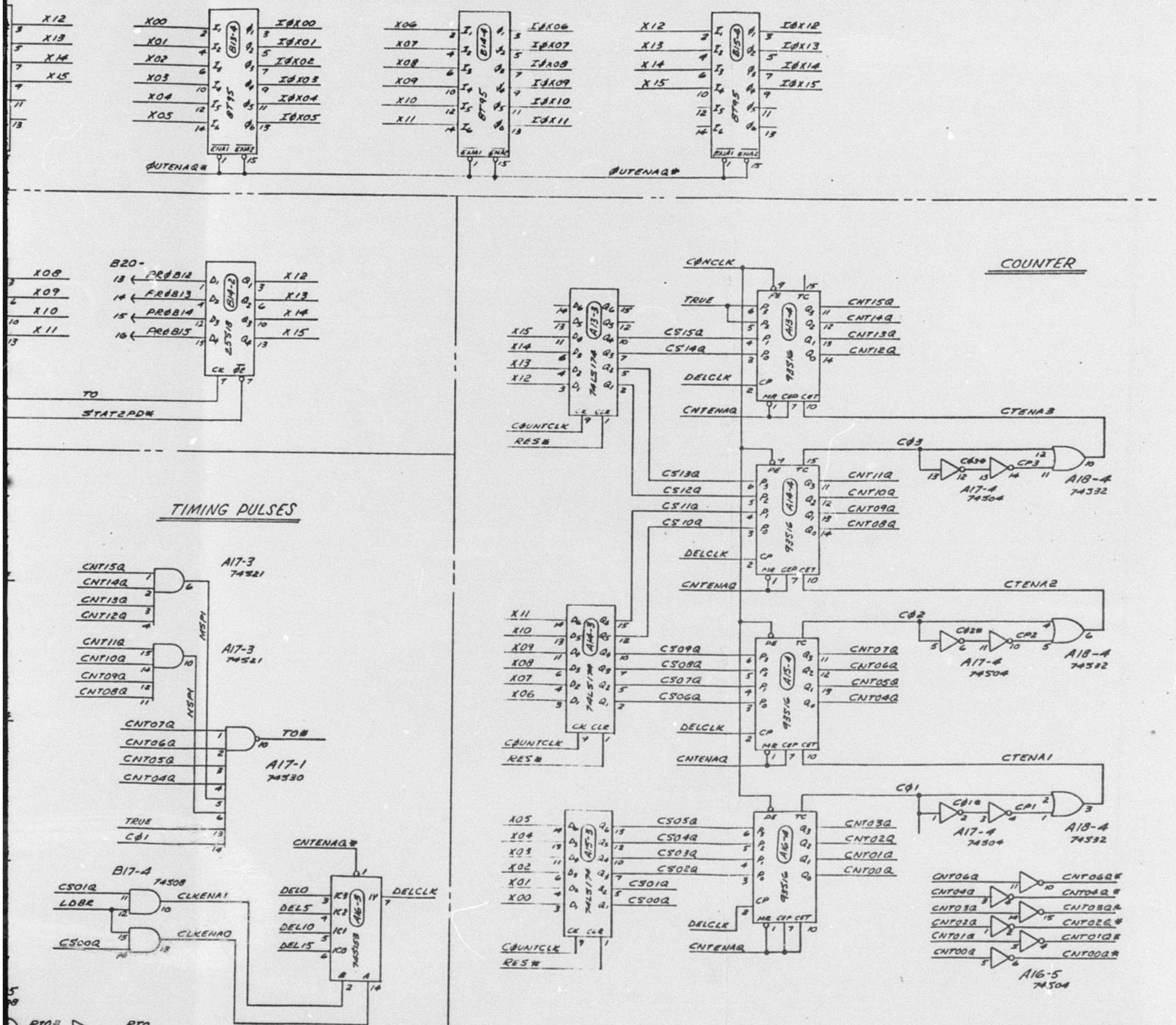
CLOCK



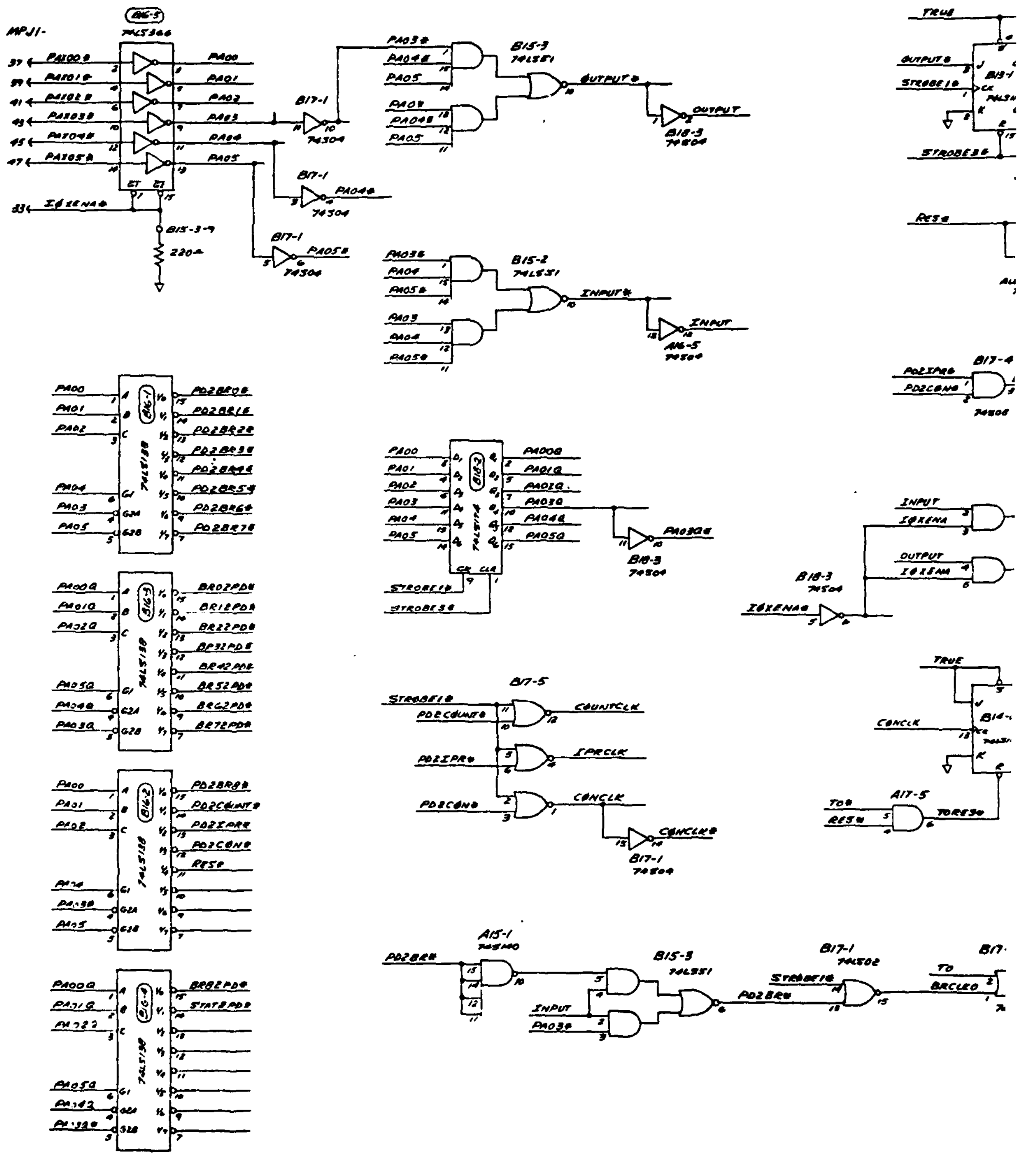
TIMIN



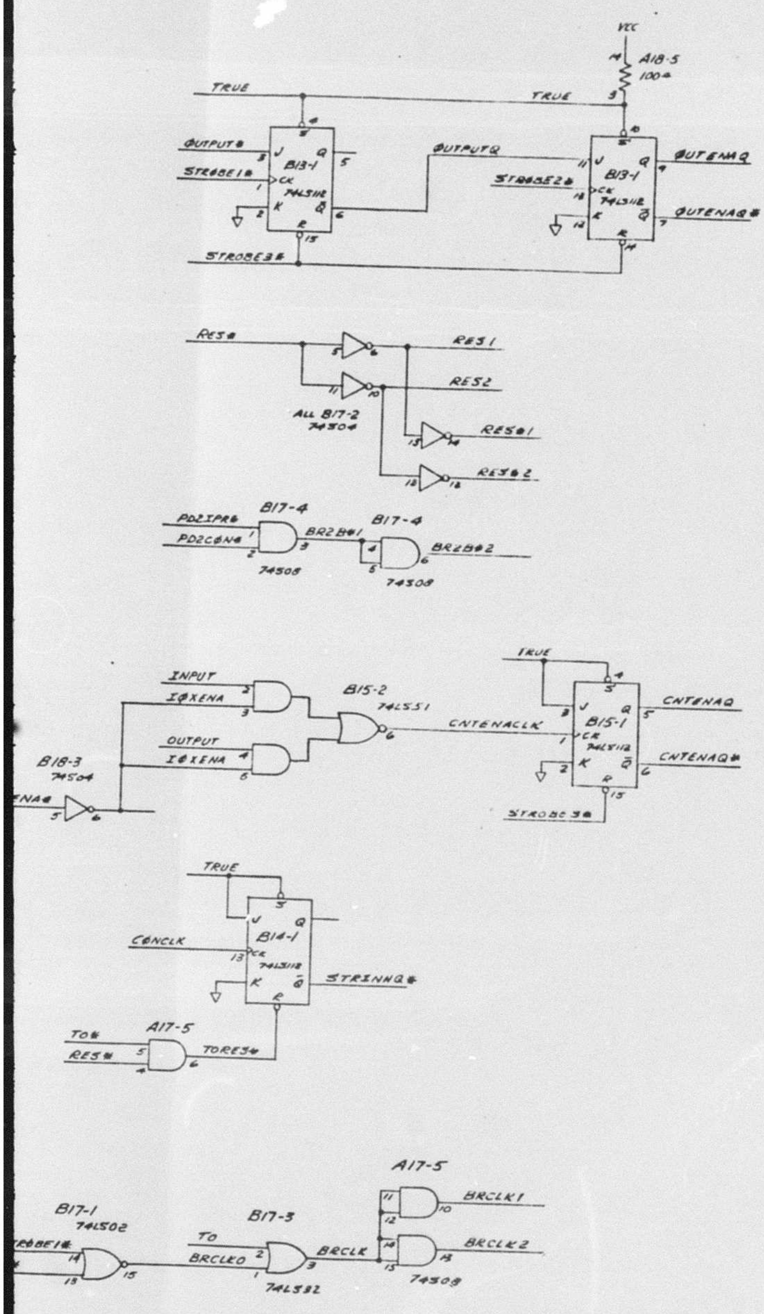
DATE	CHK	REVISION RECORD



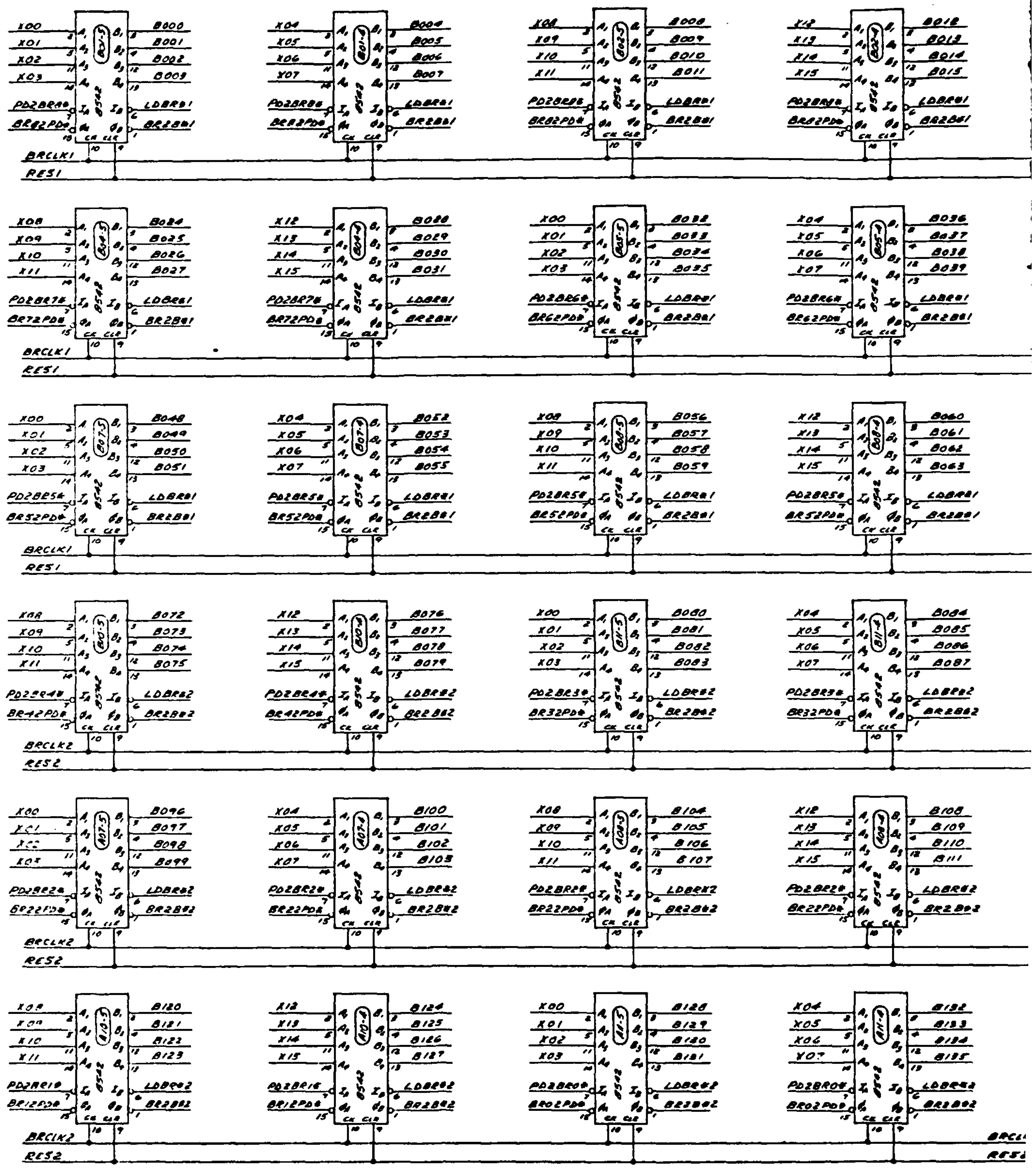
TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC.		
DECIMAL ±	MODULE ANALYZER	SCALE	DRAWN BY Duri S
			APPROVED BY
FRACTIONAL ±	TITLE TIMING LOGIC, X-BUS, STA REG		
ANGULAR ±	DATE 7/2?/79	DRAWING NUMBER MA02	



MPJ1-
B16-5
74LS366
PAI00
PAI01
PAI02
PAI03
PAI04
PAI05
PA00
PA01
PA02
PA03
PA04
PA05
IØXENA
B15-3-9
220Ω
B17-1
74S04
PA04
PA05
B15-3
74LS51
OUTPUT
B18-3
B15-2
74LS51
INPUT
A16-5
74LS138
PD2BR0
PD2BR1
PD2BR2
PD2BR3
PD2BR4
PD2BR5
PD2BR6
PD2BR7
BR0 2PD
BR1 2PD
BR2 2PD
BR3 2PD
BR4 2PD
BR5 2PD
BR6 2PD
BR7 2PD
PD2BR8
PD2COUNT
PD2IPR
PD2CON
RES
BR8 2PD
STAT 2PD
74LS174
PA00Q
PA01Q
PA02Q
PA03Q
PA04Q
PA05Q
STROBE1
STROBE3
B17-5
COUNTCLK
IPRCLK
CONCLK
A15-1
74S140
B15-3
74LS51
B17-1
74S02
STROBE1
PD2BR
TO BRCLEO
TRUE
STROBE2
B17-4
74S08
INPUT
IØXENA
B14
A17-5
TO
RES
TORES



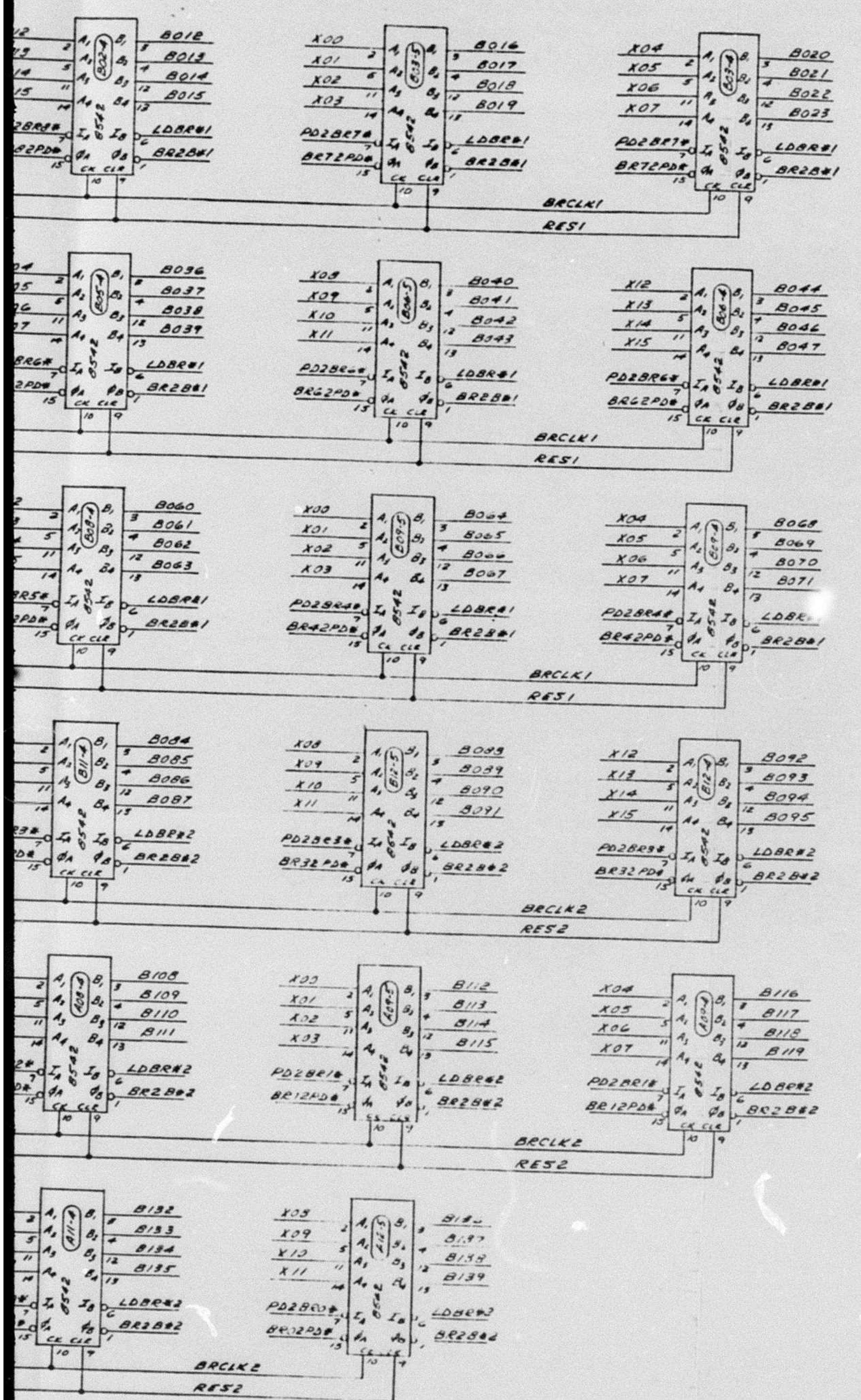
TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC•		
DECIMAL ±	MODULE ANALYZER	SCALE —	DRAWN BY Don S[illegible] APPROVED BY
FRACTIONAL ±	TITLE CONTROL LOGIC		
ANGULAR ±	DATE 7/24/79	DRAWING NUMBER MA03	



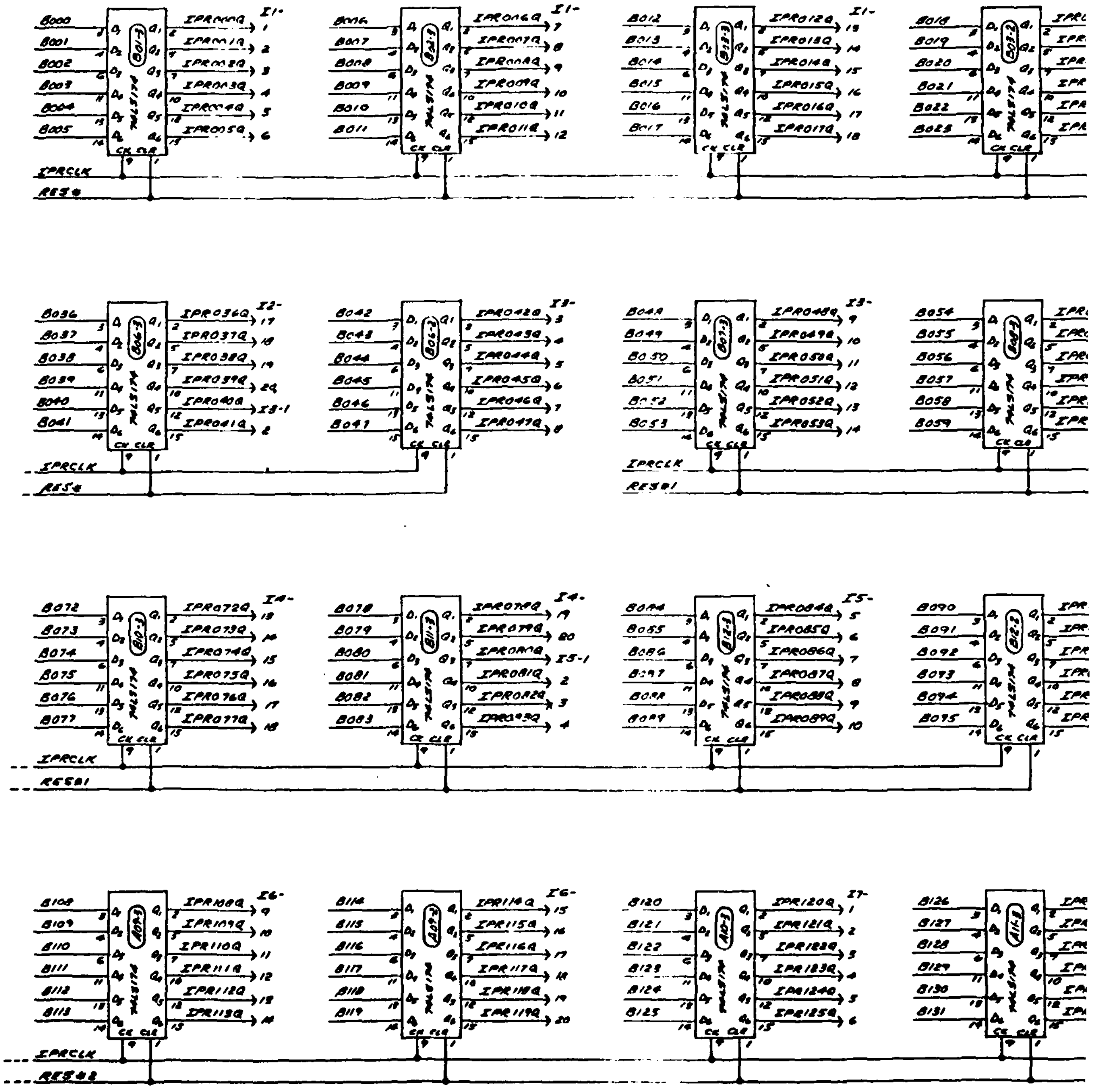
X00
X01
X02
X03
PD2BR00
BR02PD0
B000
B001
B002
B003
LDBR01
BR2B01
8542
BRCLK1
RES1
BRCLK2
RES2
B096
B097
B098
B099
B100
B101
B102
B103
B104
B105
B106
B107
B108
B109
B110
B111
B120
B121
B122
B123
B124
B125
B126
B127
B128
B129
B130
B131
B132
B133
B134
B135
LDBR02
BR2B02

1

2



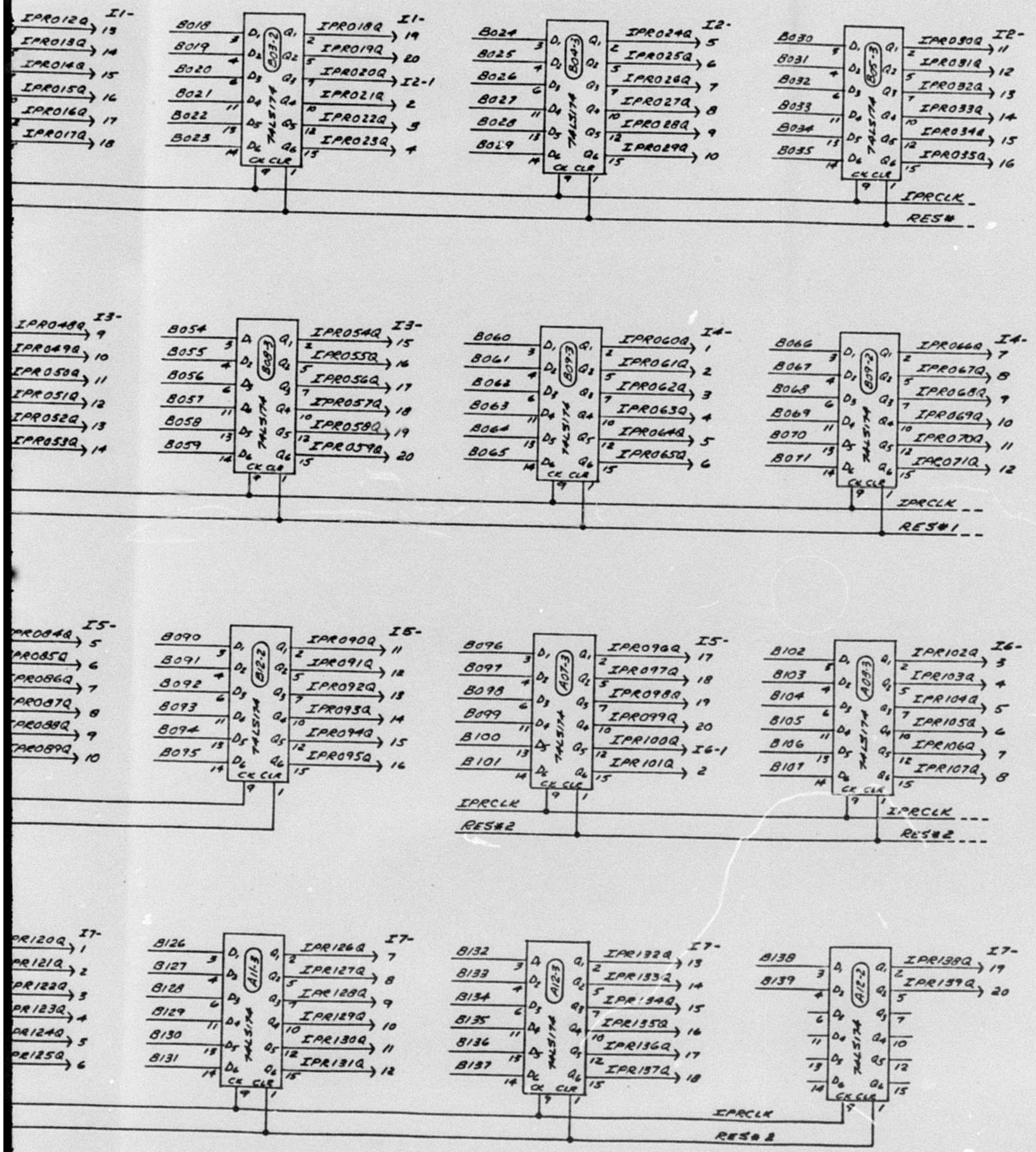
TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC.		
DECIMAL ±	MODULE ANALYZER	SCALE –	DRAWN BY D. Smooth APPROVED BY
FRACTIONAL ±	TITLE BUFFER REGISTER		
ANGULAR ±	DATE 7/24/78	DRAWING NUMBER MA04	


74LS174
IPRCLK
RES01
RES02
B000
B001
B002
B003
B004
B005
IPR000Q
IPR001Q
IPR002Q
IPR003Q
IPR004Q
IPR005Q
I1-
I2-
I3-
I4-
I5-
I6-
I7-
CK CLR

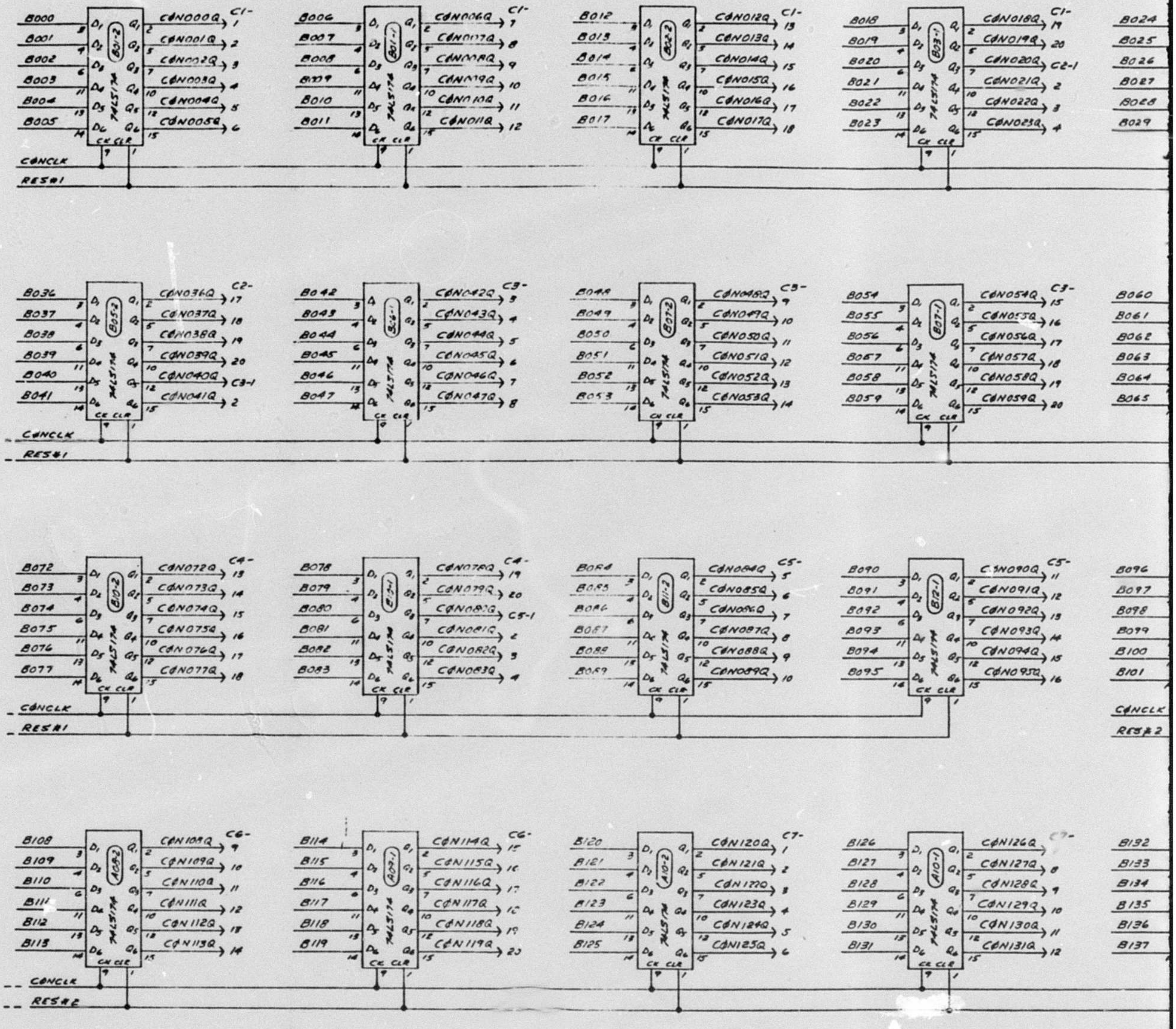
1

2

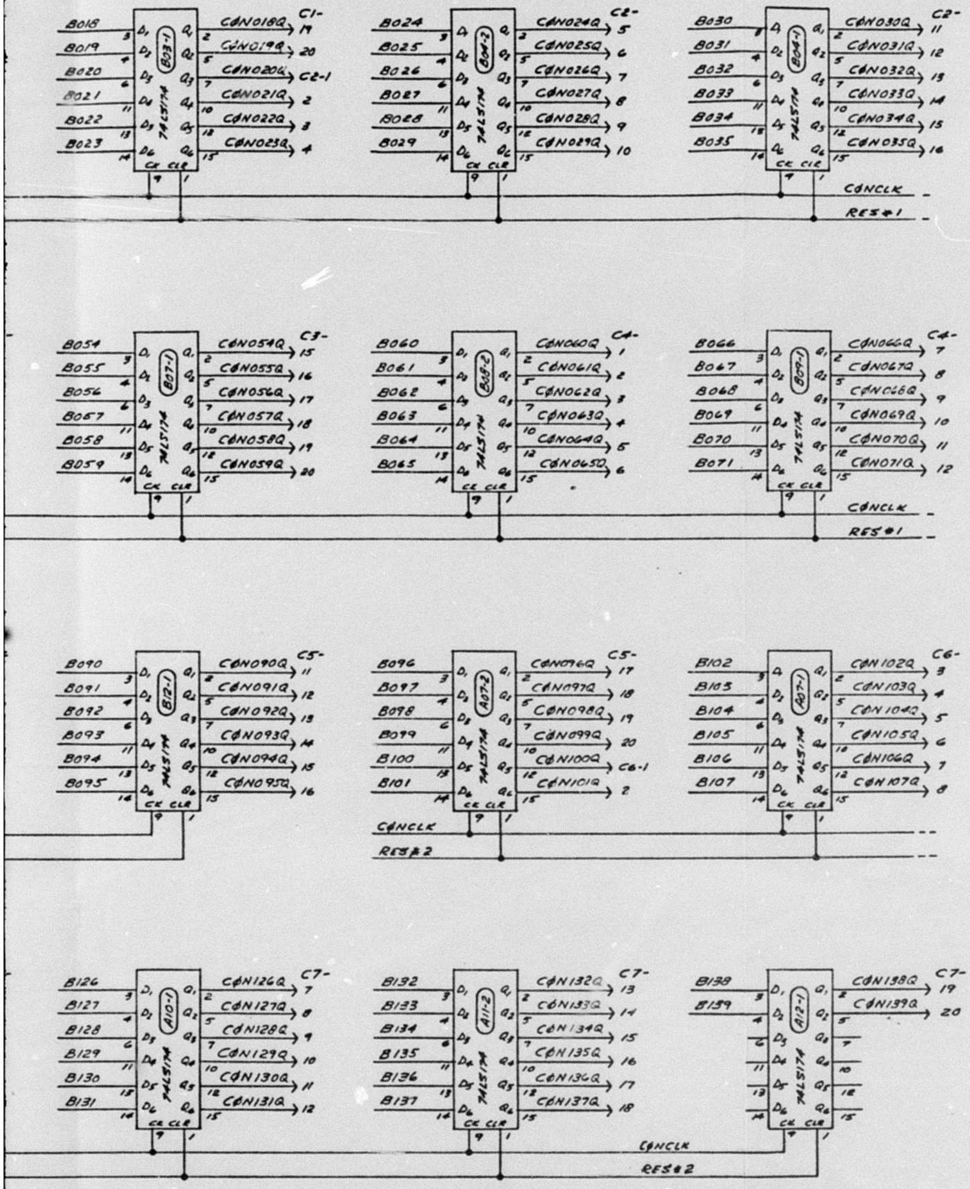
DATE	SYM	REVISION RECORD	DR	CK



TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC.			
DECIMAL ±	MODULE ANALYZER	SCALE —	DRAWN BY [illegible]	APPROVED BY
FRACTIONAL ±	TITLE INPUT REGISTER			
ANGULAR ±	DATE 7/26/78	DRAWING NUMBER MA05		


B000 B001 B002 B003 B004 B005
74LS174 B01-2
C0N000Q C0N001Q C0N002Q C0N003Q C0N004Q C0N005Q
C1- 1 2 3 4 5 6
B006 B007 B008 B009 B010 B011
74LS174 B01-1
C0N006Q C0N007Q C0N008Q C0N009Q C0N010Q C0N011Q
C1- 7 8 9 10 11 12
B012 B013 B014 B015 B016 B017
74LS174 B02-2
C0N012Q C0N013Q C0N014Q C0N015Q C0N016Q C0N017Q
C1- 13 14 15 16 17 18
B018 B019 B020 B021 B022 B023
74LS174 B03-1
C0N018Q C0N019Q C0N020Q C0N021Q C0N022Q C0N023Q
C1- 19 20 C2-1 2 3 4
B024 B025 B026 B027 B028 B029
C0NCLK
RES#1
B036 B037 B038 B039 B040 B041
74LS174 B05-2
C0N036Q C0N037Q C0N038Q C0N039Q C0N040Q C0N041Q
C2- 17 18 19 20 C3-1 2
B042 B043 B044 B045 B046 B047
74LS174 B06-1
C0N042Q C0N043Q C0N044Q C0N045Q C0N046Q C0N047Q
C3- 3 4 5 6 7 8
B048 B049 B050 B051 B052 B053
74LS174 B07-2
C0N048Q C0N049Q C0N050Q C0N051Q C0N052Q C0N053Q
C3- 9 10 11 12 13 14
B054 B055 B056 B057 B058 B059
74LS174 B07-1
C0N054Q C0N055Q C0N056Q C0N057Q C0N058Q C0N059Q
C3- 15 16 17 18 19 20
B060 B061 B062 B063 B064 B065
C0NCLK
RES#1
B072 B073 B074 B075 B076 B077
74LS174 B10-2
C0N072Q C0N073Q C0N074Q C0N075Q C0N076Q C0N077Q
C4- 13 14 15 16 17 18
B078 B079 B080 B081 B082 B083
74LS174 B13-1
C0N078Q C0N079Q C0N080Q C0N081Q C0N082Q C0N083Q
C4- 19 20 C5-1 2 3 4
B084 B085 B086 B087 B088 B089
74LS174 B11-2
C0N084Q C0N085Q C0N086Q C0N087Q C0N088Q C0N089Q
C5- 5 6 7 8 9 10
B090 B091 B092 B093 B094 B095
74LS174 B12-1
C0N090Q C0N091Q C0N092Q C0N093Q C0N094Q C0N095Q
C5- 11 12 13 14 15 16
B096 B097 B098 B099 B100 B101
C0NCLK
RES#1
RES#2
B108 B109 B110 B111 B112 B113
74LS174 A08-2
C0N108Q C0N109Q C0N110Q C0N111Q C0N112Q C0N113Q
C6- 9 10 11 12 13 14
B114 B115 B116 B117 B118 B119
74LS174 A09-1
C0N114Q C0N115Q C0N116Q C0N117Q C0N118Q C0N119Q
C6- 15 16 17 18 19 20
B120 B121 B122 B123 B124 B125
74LS174 A10-2
C0N120Q C0N121Q C0N122Q C0N123Q C0N124Q C0N125Q
C7- 1 2 3 4 5 6
B126 B127 B128 B129 B130 B131
74LS174 A10-1
C0N126Q C0N127Q C0N128Q C0N129Q C0N130Q C0N131Q
C7- 7 8 9 10 11 12
B132 B133 B134 B135 B136 B137
C0NCLK
RES#2
D1 D2 D3 D4 D5 D6 Q1 Q2 Q3 Q4 Q5 Q6 CK CLR

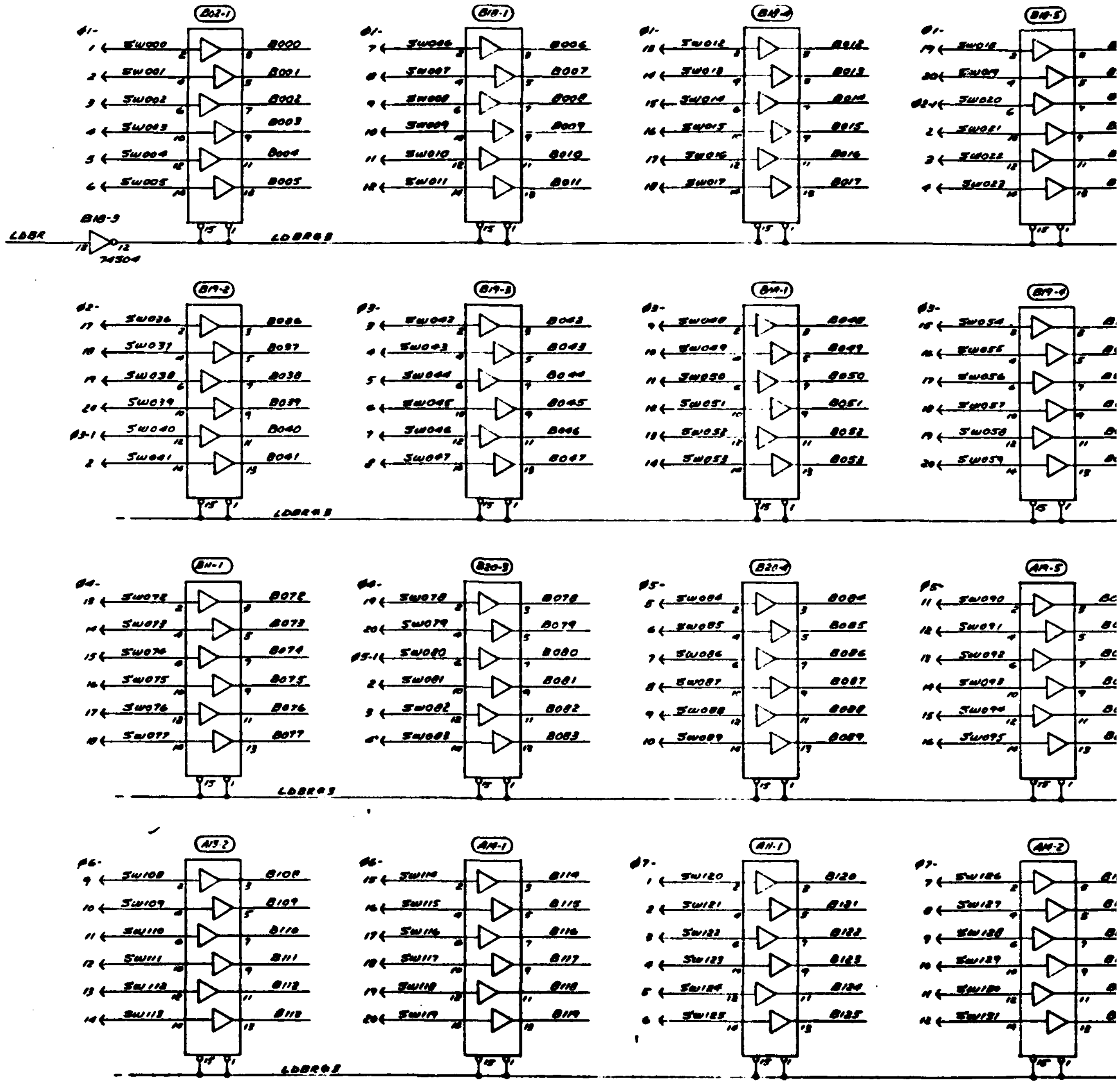
DATE	SYM	REVISION RECORD	DR	CK



B018 B019 B020 B021 B022 B023
B03-1 74LS174
C∅N018Q C∅N019Q C∅N020Q C∅N021Q C∅N022Q C∅N023Q
C1-19 20 C2-1 2 3 4
B024 B025 B026 B027 B028 B029
B04-2 74LS174
C∅N024Q C∅N025Q C∅N026Q C∅N027Q C∅N028Q C∅N029Q
C2-5 6 7 8 9 10
B030 B031 B032 B033 B034 B035
B04-1 74LS174
C∅N030Q C∅N031Q C∅N032Q C∅N033Q C∅N034Q C∅N035Q
C2-11 12 13 14 15 16
C∅NCLK
RES#1
B054 B055 B056 B057 B058 B059
B07-1 74LS174
C∅N054Q C∅N055Q C∅N056Q C∅N057Q C∅N058Q C∅N059Q
C3-15 16 17 18 19 20
B060 B061 B062 B063 B064 B065
B08-2 74LS174
C∅N060Q C∅N061Q C∅N062Q C∅N063Q C∅N064Q C∅N065Q
C4-1 2 3 4 5 6
B066 B067 B068 B069 B070 B071
B09-1 74LS174
C∅N066Q C∅N067Q C∅N068Q C∅N069Q C∅N070Q C∅N071Q
C4-7 8 9 10 11 12
C∅NCLK
RES#1
B090 B091 B092 B093 B094 B095
B12-1 74LS174
C∅N090Q C∅N091Q C∅N092Q C∅N093Q C∅N094Q C∅N095Q
C5-11 12 13 14 15 16
B096 B097 B098 B099 B100 B101
A07-2 74LS174
C∅N096Q C∅N097Q C∅N098Q C∅N099Q C∅N100Q C∅N101Q
C5-17 18 19 20 C6-1 2
B102 B103 B104 B105 B106 B107
A07-1 74LS174
C∅N102Q C∅N103Q C∅N104Q C∅N105Q C∅N106Q C∅N107Q
C6-3 4 5 6 7 8
C∅NCLK
RES#2
B126 B127 B128 B129 B130 B131
A10-1 74LS174
C∅N126Q C∅N127Q C∅N128Q C∅N129Q C∅N130Q C∅N131Q
C7-7 8 9 10 11 12
B132 B133 B134 B135 B136 B137
A11-2 74LS174
C∅N132Q C∅N133Q C∅N134Q C∅N135Q C∅N136Q C∅N137Q
C7-13 14 15 16 17 18
B138 B139
A12-1 74LS174
C∅N138Q C∅N139Q
C7-19 20
C∅NCLK
RES#2
D1 D2 D3 D4 D5 D6 Q1 Q2 Q3 Q4 Q5 Q6 CK CLR

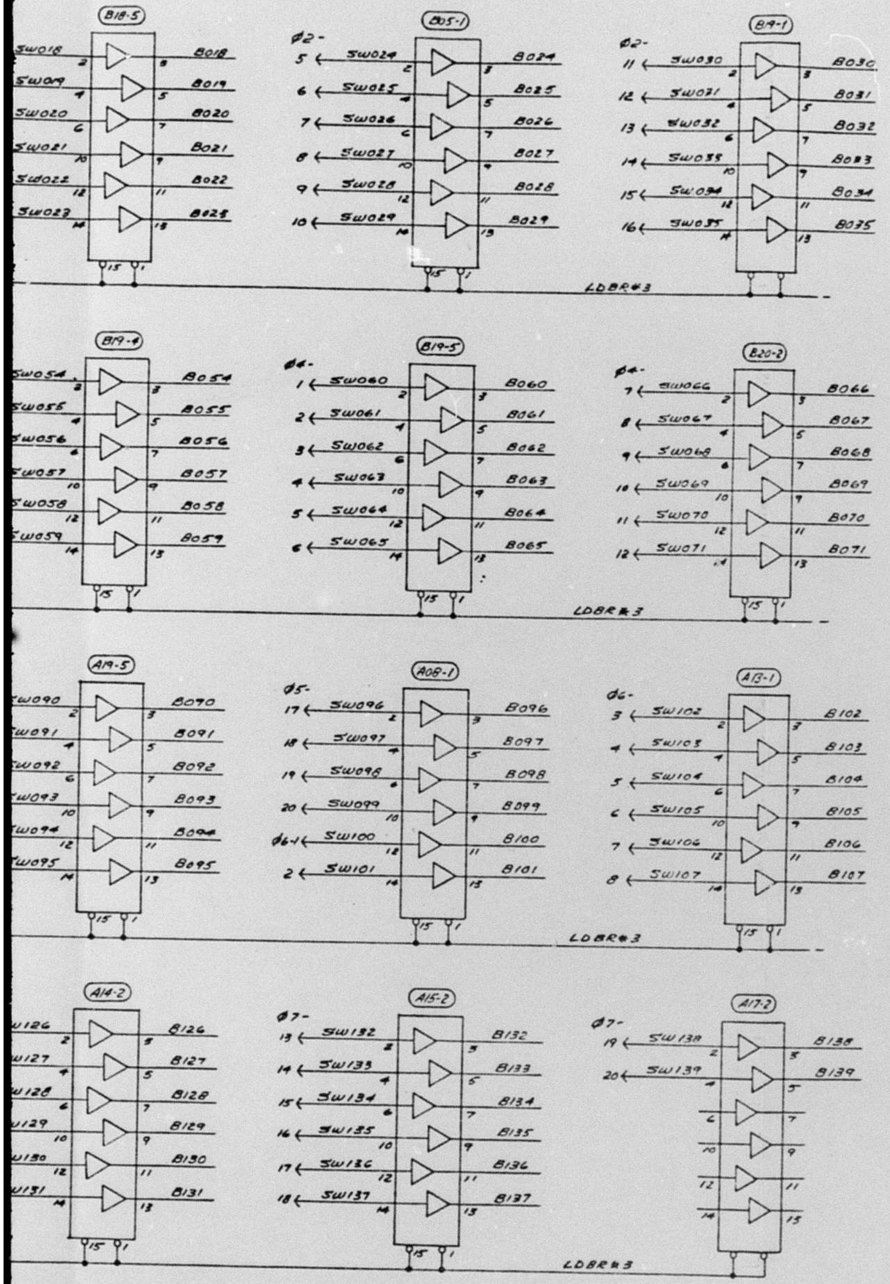
TOLERANCES (EXCEPT AS NOTED)	CHI CULLER · HARRISON INC.		
DECIMAL ±	MODULE ANALYZER	SCALE —	DRAWN BY [illegible] / APPROVED BY
FRACTIONAL ±	TITLE CONTROL REGISTER		
ANGULAR ±	DATE 7/25/80	DRAWING NUMBER MA06	

ALL DEVICES 74LS367



1

2



TOLERANCES (EXCEPT AS NOTED)	CHI CULLER-HARRISON INC.		
DECIMAL ±	MODULE ANALYZER	SCALE —	DRAWN BY [illegible] / APPROVED BY
FRACTIONAL ±	TITLE SWITCH BUFFERS		
ANGULAR ±	DATE 7/24/78	DRAWING NUMBER MA07	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CHI-QTR-301	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Structured Design for an LPC Array Processor Quarterly Technical Report Research on Speech Compression Prototype Development		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical Report May 1978 - August 1978 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. Judith Bruckner Mr. Roger Russ Dr. Glen J. Culler Dr. Michael McCammon		8. CONTRACT OR GRANT NUMBER(s) MDA903-78-C-0313
9. PERFORMING ORGANIZATION NAME AND ADDRESS Culler/Harrison, Inc. 150-A Aero Camino Goleta, CAlifornia 93017		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code: P8P10
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Supply Service -- Washington Room 1D245, The Pentagon Washington, D.C. 20310		12. REPORT DATE September 1978 13. NUMBER OF PAGES 46
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DCASMA Oxnard 300 Esplanade Dr., Suite 990 Oxnard, California 93030		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED (A)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3625.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

LPC Analysis,	Logical Modules,
LPC Synthesis,	Structured Design, and
Array Processor,	Computer Architecture. #1
Logical Design	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The architecture of an array processor that is tailored to the requirements of LPC analysis and synthesis is presented. Interim results in the development of a structured design system are also given.

Keywords included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)