DESIGN AND IMPLEMENTATION OF A SIX AXIS ROBOT CONTROLLER

by

Gautam B. Doshi

Memorandum No. UCB/ERL M89/11

7 February 1989

DESIGN AND IMPLEMENTATION OF A SIX AXIS ROBOT CONTROLLER

by

Gautam B. Doshi

Memorandum No. UCB/ERL M89/11

7 February 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

DESIGN AND IMPLEMENTATION OF A SIX AXIS ROBOT CONTROLLER

by

Gautam B. Doshi

Memorandum No. UCB/ERL M89/11

7 February 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Acknowledgements

The successful completion of this project, is the result of the help, cooperation, faith and goodwill of some of the best people I have met.

I would like to thank Professor Robert Brodersen for inspiring me through the course of this project and for his faith in my efforts.

Bill Baringer and Mani Srivastava have contributed to this project in more ways than I can list here. The insightful discussions we have had together have been instrumental in shaping this project.

I would like to extend my sincere appreciation to Phil Schrupp and Ken Lutz for their cheerful support and cooperation, and for sharing their invaluable experience in board design.

Special thanks are due to Jane Sun for her support in THOR simulations and to Anantha Chandrakasan for his help in compiling the plots in this text.

The presentation of this report has been possible because of the tireless efforts of Ming Lin.

Finally, I would like to thank my family, who for too long have found their lives to be disrupted by The Project.

Gautam Doshi
Jan. 20, 1989

This research was supported by DARPA/SPAWAR Contract N00039-87-C-0182.

Contents

Chapter 1

Introduction

1.1 A Perspective

The need for higher levels of productivity and consistency in the quality of the end product have encouraged the use of computer based machines in the industrial environment. Most such machines are special purpose units designed to perform a limited set of predetermined functions. However, a growing class of these automatons consist of Robots.

The word *robot* bears its origin in the Czech word *robota*, which means "work". There is little consensus on the exact definition of a robot, however the term is usually used to describe a general purpose, multi-functional manipulator that can be programmed to perform a wide variety of tasks. A connotation of "intelligence" is built into the word "robot", and is ascribed to the computer algorithms that govern the control and sensing mechanisms of the robot [4].

A large majority of the population of industrial robots employ relatively simple control schemes and have virtually no external sensors. This makes these robots well suited to perform simple repetitive tasks in a fixed environment, but hardly viable for complex tasks that require the robot to interact with a changing environment. This adaptability to the surroundings requires that the control mechanism of the robot use the sensory data acquired by the external sensors of the robot. The integration of sensing mechanisms in a robot, to allow it to interact intelligently with the environment with high speed and accuracy, is an active topic of research and development.

The external sensing mechanisms most often used are *vision* and *force*. Vision is excellent for locating parts with respect to the manipulator, while force sensing is much

more useful in mating parts together. Touch feedback (force) is sometimes slower and more restrictive, physically due to its groping nature [11]. For contact tasks such as grinding, polishing, writing, tightening etc, force feedback is extremely useful. In the case of tasks that require coarse and fine positioning without contact, vision is appropriate. Vision and force are therefore best used as complementary feedback mechanisms since each has specific advantages over the other. In either case, it is clear that the specifications, capabilities and "intelligence" of the robot are greatly enhanced by the use of sensory data in its control algorithm.

It is our endeavor to build such an "intelligent" robot system using a hybrid force-position control scheme at the low level and a high level controller that uses a combination of vision, force, proximity and tactile sensors. Towards this end, a commercial industrial robot system has been acquired. The robot system consists of a six-axis vertically articulated robot arm and a position controller with two peripherals - a teaching panel and a terminal. The robot arm is remarkably impressive in terms of its mechanical specifications. The controller, however, lacks several desired characteristics (delved into, later in the report). A new and improved system using the existing robot arm (along with its power amplifiers) is therefore conceived. The envisaged system is contrasted with the existing system in Fig 1.1.

The system is envisaged as consisting of a low level robot control board, a host computer board and one or more sensor board communicating over a common VMEbus. The robot control board will interface to the robot at a low hardware level, and to the user via the VMEbus and the host computer. The activities of the robot control board and the sensor boards will be orchestrated by the host computer. This report describes the design of the VMEbus based robot control board in this system.

1.2 Organization of the Report

This report is organized hierarchically, with the first few chapters presenting a broad functional overview of the system and its components and the later chapters explaining the details of their operation.

Chapter 2 starts with a brief description of the robot and its components and explains their operation in conjunction with the controller. The original controller and the proposed controller are contrasted functionally in this chapter. Chapter 3 provides a block diagram view of the proposed controller, describes the blocks, their operation and

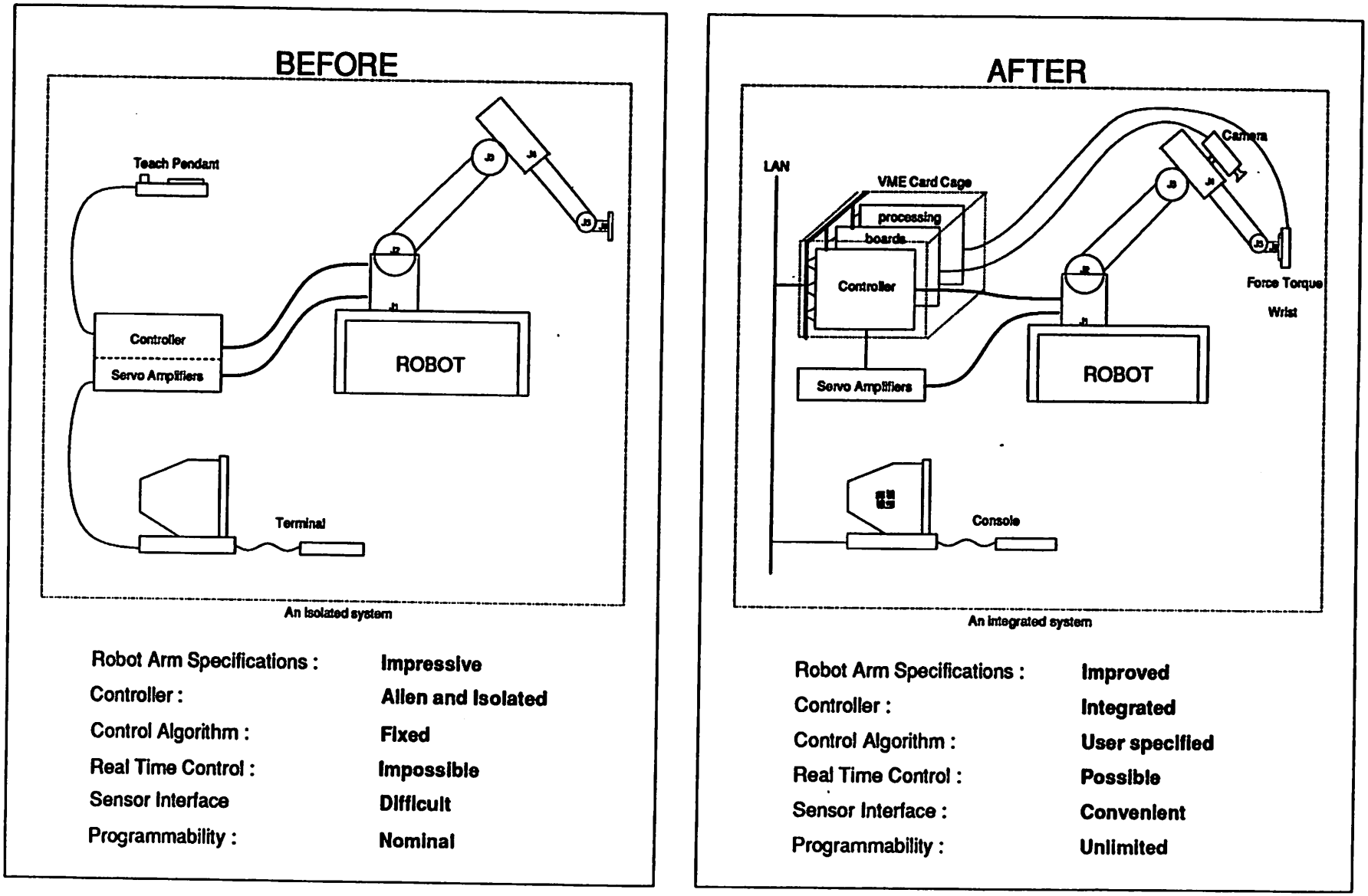


Figure 1.1: System Overview "Before" and "After"

their interconnection. These blocks can be divided into 3 general categories: blocks that perform the robot interface tasks, blocks that manage the on-board processing, and blocks that perform the VMEbus interface. A chapter is devoted to the discussion of the blocks in each of the three categories. Chapter 4 details the blocks that relate to the hardware interface to the robot. The on-board computing engine and its memory organization is described in Chapter 5. Chapter 6 discusses the details of operation of the VMEbus interface circuitry. The next chapter discusses the test strategy incorporated in the design and presents the idea of "self-test". Chapter 8 describes an assortment of on-board circuits best described as miscellaneous. Chapter 9 provides some suggestions that may be useful when the second generation of this controller board is designed. Chapter 10 concludes the report by presenting a few "golden nuggets" acquired in the course of this design exercise.

Chapter 2

System Overview

2.1 The Robot

2.1.1 A Functional Description

A typical industrial robot is made of mechanical *links* connected together by *joints*. The links can move, either by sliding within the joint (prismatic joints) or by rotating about the joint (rotary joints). (In common parlance, the *joint* rather than the *link* is referred to as moving.) The motion of a joint is effected by a motor connected to it, usually through a gear train. The link-joint assembly, referred to as the *robot arm*, is connected to a rigid support. The region of space traversed by moving all the joints through their operating range is called the *workspace* of the robot. By a coordinated motion of the joints, a trajectory may be traced by the robot arm within its workspace.

In order to perform useful work, an *end-effector* is connected to the end of the arm. The end-effector is a tool that is suited to perform the required task (drilling, picking, placing, etc.) on an object in the workspace. The robot arm is used to position and orient the end-effector over the object, and to move it from point to point within the workspace along a desired trajectory.

The desired trajectory is specified by the user, usually in Cartesian coordinates. In order to move the robot arm along this trajectory, the trajectory specification must be formulated in terms of the joint angles. The motion of all the joints must then be coordinated to obtain the desired motion of the arm. To move a joint itself, requires controlling the drive applied to the motor connected to it. Ensuring that the robot arm actually does trace

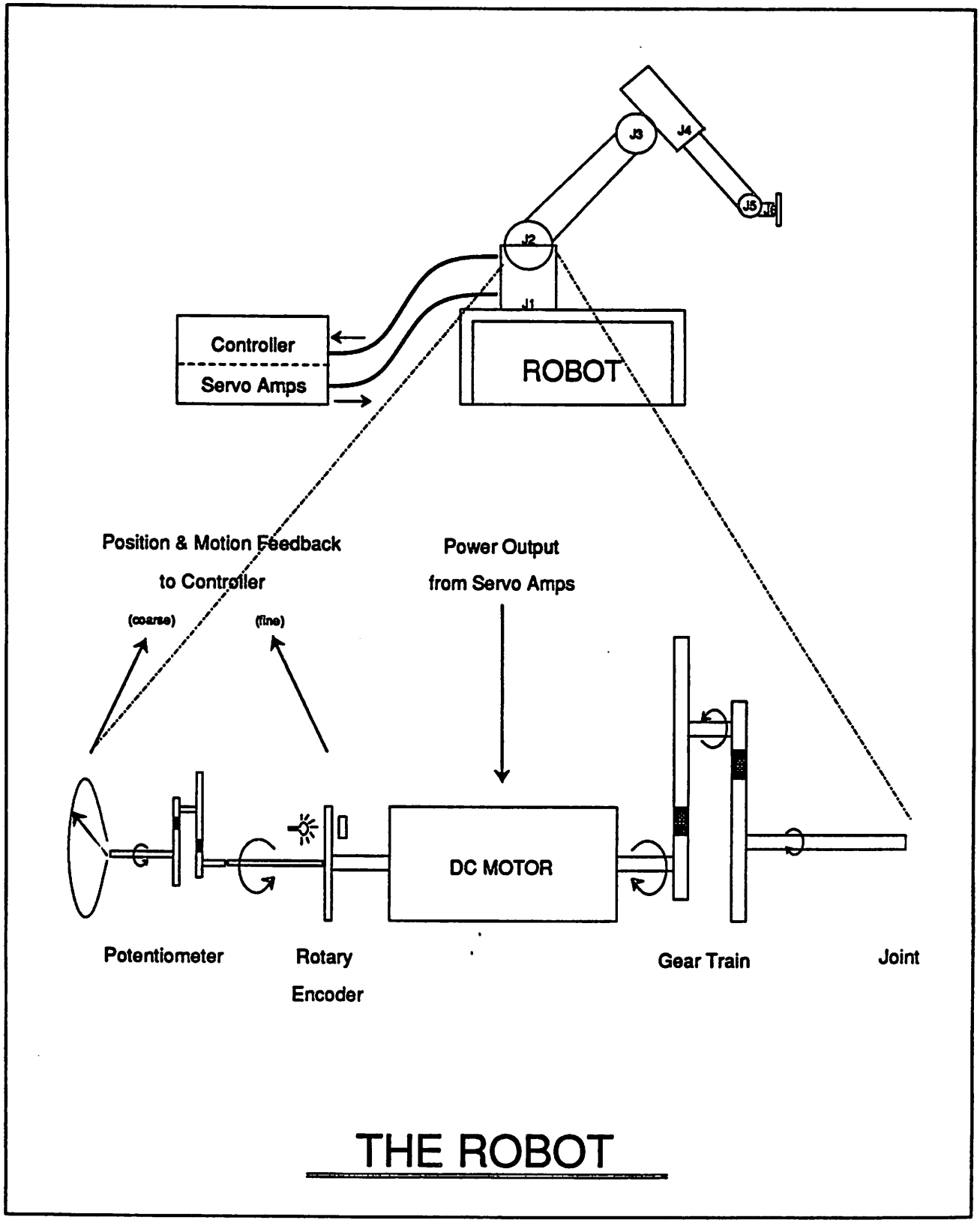


Figure 2.1: The Robot

the desired trajectory requires monitoring the motion of the robot arm and executing a control algorithm which attempts to correct any deviations of the actual trajectory from the desired one. All these tasks, at different levels of control, fall in the realm of the tasks of a *controller.*

The controller, the robot arm and the I/O devices that enable the user to interact with the controller, constitute a typical robot system.

The robot system, for which this controller is designed, is a commercially available industrial robot system made by Panasonic. It comprises the *PanaRobo V1* robot arm, the *PanaDac 360* robot controller, a *CRT display and keyboard* and a *teaching panel.* An overview of the specifications of the PanaRobo V1 robot arm and the PanaDac 360 controller are provided in Appendix A.

2.1.2 The Theory of Operation

This section briefly describes, the operation of the robot arm (PanaRobo V1) under the control of its original controller (PanaDac 360). The tasks of the controller are varied and span several levels of the control hierarchy. The low level control operations are described below since they are of specific interest here. The high level control tasks performed by the controller are described in detail in the User's Manual of the PanaDac 360.

At the lowest level of control, the controller performs the basic functions of monitoring the motion of the robot arm and driving the motors. The controller obtains the joint angle and the joint motion information from the potentiometer and the rotary encoder resident on each joint. This information is used in the control algorithm to determine the desired drive to be applied to the motor connected to each joint. The controller then issues actuating signals to the servo amplifiers, which drive the DC servo motors directly.

When the controller is powered up, the state of the robot arm (as defined by the joint angles) is unknown to the controller. A sequence of operations is thus performed to bring the robot arm to a known position, referred to as the *origin position.* A potentiometer, connected to each joint motor, is used to measure the absolute value of the joint angle. The potentiometer is connected to the motor shaft through a gear train. Therefore, the angular position of the variable terminal of the potentiometer, and in turn the output voltage of potentiometer, is representative of the actual joint angle.

The values of all the joint angles at the origin position is known to the controller.

Therefore, only the incremental motion information is needed to keep an updated account of the joint angles. The incremental motion of each joint is measured by the rotary encoder connected to the joint's motor shaft. The rotary encoder consists of a disk that is placed between a light source and a light sensor. The disk is directly connected to the motor shaft. The disk has two rings of uniformly spaced perforations (windows) along its periphery. Light passing through these windows is sensed by the light sensor, which produces two signals called the *encoder channels*. The encoder channels are two pulse trains that have a constant phase difference of 90 degrees, because the two rings are physically offset, relative to each other, by half a window. The pulse width of the signal on either channel indicates the speed of rotation of the motor shaft. If the shaft rotates rapidly, the windows pass by the light source quickly and the pulse width of the encoder channel signals is smaller. The relative phase (i.e. channel A leading channel B by 90 degrees, or vice versa) of the encoder channels indicates the direction of motion of the motor shaft. A measure of the angular displacement, between two positions of the motor shaft, is obtained by counting the number of pulses on either encoder channel, when the motor shaft is moved from one position to another. Thus the encoder channel signals provide the direction and magnitude of the angular motion of the joint.

In addition to the encoder channels, the rotary encoder also provides another signal called the *index* signal. A single pulse is issued on this signal, once every motor rotation. This helps to mark a reference position of the motor shaft.

The power to drive the motors is provided by the servo amplifier boards (the power section) within the controller. These boards contain a circuit implementation of an H-bridge configuration. The four switches of the H-bridge are actuated by Pulse Width Modulated (PWM) signals from the control section of the controller. A fixed voltage is applied to the motor by closing two of the four switches. The polarity of the applied voltage is reversed by closing the other two switches instead. The fraction of time for which the voltage is applied to the motor is regulated by changing the duty cycle of the actuating signals. Thus the direction and magnitude of the drive to the motors is controlled.

The servo amplifiers provide a feedback signal to the control section. This signal represents the actual current flowing in the motor and is used to implement a current control loop embedded within the position control loop. The output of the position control loop is the required torque or alternately the required motor current. The actual motor current is compared with the desired motor current and the difference signal is used to modulate the

pulse width of the PWM signals to the servo amplifier.

In order to handle exception conditions that may arise during the operation of the robot, a few other signals are provided to the controller. A signal indicating an overrun of a joint beyond its operating limit is provided by the robot arm to the controller. Another signal that indicates excessive motor currents is also provided by the servo amplifiers to the control section. In response to either of these signal, the controller halts the robot arm by discontinuing the power supplied to the motor and applying a set of brakes.

2.2 The Existing Controller

The PanaDac 360 is a low level position controller for the PanaRobo V1 robot arm. It is physically a large box (16"x17"x19") that connects to the robot via two large cables; one providing position feedback from the robot arm to the controller and the other providing drive from the controller to the motors in the robot arm. The user interface to the controller is via a keyboard and CRT display and via a teaching panel.

2.2.1 The Shortcomings

The PanaDac 360 provides adequate position control when the robot is used for primitive pick-and-place tasks in a structured environment, where the objects being operated on are placed at precise locations. Such scenarios are often seen in the industrial work environment. However, as the complexity, or the required dexterity, of the task increases, the limitations of the PanaDac 360 are evident.

Firstly, the control algorithm is hard-coded, and cannot be modified by the user. This is quite a handicap, since it limits the dynamic response of the robot and also precludes the use of additional sensory data for improved control.

Secondly, the controller is completely isolated from the computing environment. The teaching panel and the CRT-keyboard interfaces require that a user monitor the operations of the robot arm and direct the controller accordingly. Host computer access to the controller is provided via an RS232 link. However, only a very limited set of tasks may be performed via this channel.

Thirdly, the controller is unable to respond to changes in the working environment. It uses only the position feedback data from the robot arm, in its position control algorithm.

Due to the lack of external sensors, it is incapable of sensing changes in the environment and adapting to those changes.

Lastly, the controller can only be programmed to perform a repetitive task. By using the teaching panel, the user can "teach" the controller, a set of points in the robot's workspace, and the type of the interpolation to be used to join these points. This is recorded as a program. A set of 30 such programs may be "taught". The controller can then be directed to playback any one of these programs continuously or for a fixed number of times. Thus, the programmability of the controller is limited to "teach and play". Real time control is impossible.

2.3 The Proposed Controller

The proposed controller is a VMEbus based processing module designed for performing real-time force-position control of the PanaRobo V1 robot arm. It is physically a single VMEbus board (11" x 14.4") featuring a powerful digital signal processor for the execution of complex user-specified control algorithms, and several general purpose motion control ICs for interfacing to the robot arm. It connects to the PanaRob V1 robot arm and to the servo amplifiers of the original PanaDac 360 controller.

The controller board is a VMEbus slave module that resides in a VMEbus card cage along with the Heurikon HK68/V20 processor board and other data processing boards. The Heurikon board acts as a bus master running a real-time UNIX-like kernel — VxWORKS. An Ethernet controller board, also on the VMEbus, provides the user access to the Heurikon board and to the controller board, remotely from any computer on the network.

2.3.1 The Capabilities

The controller board is designed to alleviate the limitations imposed by the PanaDac 360 and to increase the capabilities of the robot system using the same robot arm and servo amplifiers.

The control algorithm is implemented in software and is therefore under the control of the user and easily modified. The option of using a simple control algorithm, implemented in hardware, is also provided.

The controller is well integrated in the computing environment. The choice of an industry standard bus has permitted the use of commercially available processing modules

for performing user interface functions. It has also enabled communication between the controller board and other computers on the local area network, allowing both remote access to the controller, and off-board computing power to the controller.

The controller board, per se, provides only an RS232 link for force-torque sensor interface. Moreover, since the controller is based on the VMEbus, other data processing boards may be designed as plug-in modules. One such module that will allow the integration of vision feedback to the controller is currently being designed.

The controller provides virtually limitless programmability. The on-board DSP has access to the low level robot interface signals, and the operation of the DSP can be controlled by the VMEbus master. The bus master can communicate with the network computers. Thus, the network computers can be used for off-line programming and controlling the operation of the controller board.

The intergration of the controller, the ability to interface external sensors and the high speed on-board computing capability are the key features that enable this controller board to be used as a force-position controller for the six axis robot arm.

Chapter 3

Architecture of the Controller

The basic function of a controller is to guide the robot, as directed by the user, along a desired trajectory in space. The task of controlling a robot transcends several levels of hierarchy - starting from global issues of strategy formulation and path planning, down to the specifics of issuing actuating signals to the drive mechanism [?]. Typically, the user issues control commands at a certain level (e.g. move left 10), and the controller takes care of all the lower level functions required to execute the command (i.e. coordinating the generation of control signals for the joint motors to effect a motion of the robot arm 10 units to the left in the Cartesian space).

The level at which the user can communicate to the controller and, in turn, the ability of the controller to handle all subsequent lower level functions, is often limited by its computing capabilities. The computing capabilities of the controller also limit the complexity of the control algorithm it can implement in real time. It was therefore important to decide the amount of processing power that the controller should be given. The controller presented in this report embodies three significant design decisions in this regard.

Firstly, a powerful floating point Digital Signal Processor (DSP) chip - AT&T's DSP32 is at the core of the controller. A simpler general purpose microprocessor would suffice for PID control. However, since this controller is targeted to perform real-time force control algorithms, which are extremely computation intensive, a DSP chip was found to be mandatory.

Secondly, the controller is designed to communicate with a single board computer (SBC), Heurikon's 68020 based HK68/V20, over the VMEbus. Other data acquisition and processing boards can be communicated to, via the SBC as well. An Ethernet board shall

be used to communicate with the SBC and in turn to the controller, remotely from other machines on the network. It is thus possible to integrate the controller into the network and use the resources of the network computers for the processing needs of the algorithm.

Lastly, a set of general purpose motion control chips (an HP's HCTL1000 per joint) have been used to off-load the robot-related interface tasks from the DSP, allowing the DSP more time for algorithm processing.

The controller is thus a powerful VMEbus based processing module that interfaces with the user (via VMEbus) on one side, and to the robot on the other. Its interaction with the user involves, obtaining commands and providing status reports. Its interaction with the robot involves, providing the necessary actuating signals and obtaining sensory feedback data.

The interface to the robot has been made by severing the connections between the robot arm (PanaRobo V1) and the original Panasonic controller (PanaDac 360) at two places. The original controller can be looked upon as consisting of two parts; a control section and a power section. The control section receives the sensory data from the robot arm, executes the control algorithm and issues the control commands (physically, the top half of the PanaDac 360). The power section (servo amplifiers) receives these commands and boosts their power level to drive the joint motors (physically, the lower half). While the control section of the PanaDac 360 is replaced by this controller the power section of the PanaDac 360 is still used. The compatibility in ratings between the servo amplifiers and the joint motors is thus assured. Additionally the interface between the control section and the power section was found to be relatively "clean" and, in the sparse documentation of the PanaDac 360, this was one of the better documented items. Thus, this controller interfaces to the robot by receiving the sensory data from the arm and by providing control commands to the original servo amplifiers that drive the joint motors.

The block diagram (Fig 3.1) serves to identify the significant functional blocks within the controller. The interconnections shown are only indicative of the flow of information (control or data) and do not necessarily denote physical connection between the blocks. The details of implementation (presented in later chapters) are hidden at this time — to enable the reader to get a better perspective of the major functional elements of the controller.

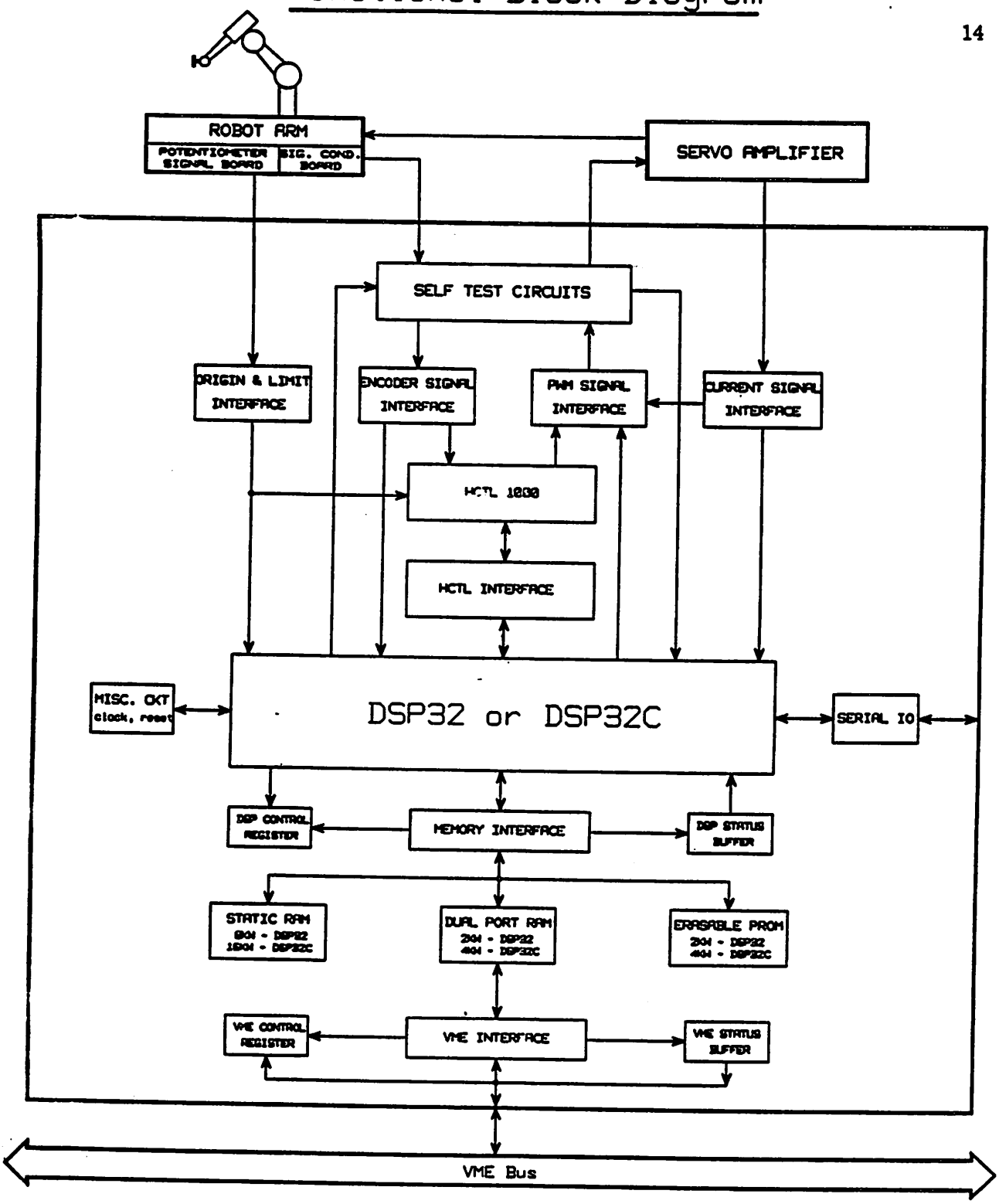


Figure 3.1: Block Diagram of the Controller

3.1 The Robot Side

The interface to the robot consists of four blocks that handle the four kinds of signals between the controller and the robot.

- Position Signals (Origin and Limit Signal Interface): The origin signal gives a coarse indication of the absolute position of the robot. The limit signals are asserted when a joint reaches the boundary of its operating range. These signals are used to bring the robot into a known position and to keep it operating within its workspace.
- Motion Signals (Encoder Interface): The information regarding the direction and magnitude of the incremental motion of the joints is provided by the encoder signals.
- Command Signals (PWM Interface): The controller sends commands to the servo amplifiers encoded as Pulse Width Modulated (PWM) signals. These signals control the magnitude and direction of the drive applied to the joint motors.
- Feedback Signals (Current Interface): The servo amplifiers provide feedback regarding the actual current flowing in the motors via the "I+,I-" signals. The servo amplifiers also assert an error signal (*Ierror*) when the current in the motor exceeds operating limits.

The Hewlett Packard HCTL 1000 is a general purpose motion control chip that can be programmed in various ways to control the point-to-point motion of a joint. A simple position control function, with programmable coefficients, is implemented in the HCTL. The DSP communicates with the HCTL to provide the coefficients of the control transfer function. The DSP has the option of bypassing the control function of the HCTL altogether, and using it only as an I/O device to the robot.

The *position signals* are provided to the DSP. The DSP can directly monitor the position signals of each joint of the robot and use them during the initialization of the arm. The *motion signals* are presented to the HCTL which uses them to keep track of the joint position and velocity. Based on the HCTL's programmed operating mode, and the control coefficients provided by the DSP, the HCTL produces the *command signals*. These signals are converted into a different format as required by the servo amplifiers. The servo amplifiers provide *feedback signals* that indicate the value of the current flowing in the motors. These analog signals are digitized and provided to the DSP.

The operation of these blocks is individually discussed in detail in later chapters. However, a common issue that underlies the design of most of these blocks is the use of reconfigurable logic circuits. Most of these blocks are implemented using *Erasable Programmable Logic Devices* (EPLD). One of the key advantages of using EPLDs, in a prototyping project such as this, is design flexibility. Lack of sufficient documentation on the original controller necessitated making certain assumptions in the design process. By using EPLDs, the penalty of making a wrong assumption was minimized. Additionally, the design turnaround time for EPLD was extremely small, and in most cases resulted in an implementation that was frugal in board area and power.

A key criterion in the choice of HCTL (over National's LM628) as the supporting robot control chip was that it allowed the user to bypass the control function built inside the HCTL. This allows the DSP to perform the control function while using the HCTL as a smart IO device.

3.2 The User Side

The user communicates (via a computer) with the controller over the VMEbus. The controller is a VMEbus Slave and Interrupter board. The VMEbus interface ensures the adherence to the VMEbus protocol for data transfer between the controller board and the VMEbus master and enables the controller board to interrupt the VMEbus master.

An additional communication channel is provided by a dual serial line. This was included to function initially as a debugging aid and later to allow interfacing a terminal and/or a computer to the controller. It can also be used to directly communicate with an end-effector or a force-torque wrist.

3.3 On-board Processing

The on-board computing engine is the DSP32. At the time that the choice of processor was made, advance information of the successor DSP chip, the DSP32C, was available. Since the DSP32C claimed to offer a higher MFLOP rate than its predecessor, along with interrupts and a larger memory space, the controller has been designed to work with either processor and to exploit the capabilities offered by both.

The memory for the processor consists of Erasable Programmable ROM, Static

RAM and (true) Dual Port RAM. The EPROM is provided to boot the controller from a resident kernel. Local memory consists of Static RAM (SRAM) and Dual Port RAM (DPRAM). The Dual Port RAM is accessible by both the VMEbus and the DSP, enabling a mail-box communication scheme between the VMEbus master and the controller.

3.4 Self-test Circuits

A novel feature of this controller is its ability to simulate the robot signals without being connected to an actual robot. During normal operation, encoder signals are received, and PWM signals are issued, by the controller. By providing suitable test circuitry - the DSP is capable of simulating these signals at the robot interface - even in the absence of an actual robot. The DSP, under program control, can generate the robot signals that test the operation of the HCTLs, the encoder interface and the HCTL interface. Since the robot is extremely agile and powerful, malfunction of the controller when the robot is connected to it can have serious repurcussions. This self-test scheme provides a safe way for the controller to perform sanity checks on its functional blocks during normal operation and to perform diagnostic checks periodically. In the initial stages of board debugging this can be an invaluable tool since it makes a large section of the board both controllable and observable by the DSP itself.

3.5 Miscellaneous Circuits

Other on-board miscellaneous circuits include clock generators, processor reset circuit, and dual serial input/output port.

Chapter 4

Robot Interface

4.1 The Origin and Limit Signal Interface

Before commencing normal operation, the robot arm has to be oriented to a known position. This position is known as the "origin" or "home". It is attained using the potentiometer and the joint encoder on each joint. The potentiometer signal boards, placed at the base of the robot arm, drive the joint potentiometers and preprocess their raw output to generate an origin signal (O+) for each joint. The O+ signal is binary and indicates whether the present joint angle is greater than or less than its value at origin. Based on the value of this signal the joint is slowly moved until a transition on this signal is observed. This indicates that the joint is in the vicinity of its origin position. The joint is then zeroed in to its origin position by moving it and observing one or more index pulses and then counting a certain number of encoder counts.

The joint angles at the origin must be known *exactly*, for the controller to correctly map locations between the Cartesian space and the joint space. A straight line motion command may not function correctly if the actual arm configuration differs from the controller's perception of the arm configuration. The specifics of the number of index pulses and encoder counts through which a joint should be moved, following a transition of the O+ signal, may be changed so long as the controller is aware of the exact joint angles at origin. The point at which the O+ signal itself makes the transition can also be altered by adjusting the potentiometer board pots.

Moving the robot arm to its "origin" position (referred to as "origining" through the rest of this text) is performed by the DSP. It is therefore necessary that the DSP be

able to read the O+ signal directly.

This ability to monitor the O+ signals is implemented as follows:

The OGNx [x=1–6] (sch#10) signals are the open collector signals (O+) from the potentiometer signal board. These signals are pulled high by the resistor pack (R15 sch#19) and are input to the "IxPls_Ogn13" and "IxPls_Ogn46" PLDs (sch#17). These signals are memory-mapped in the DSP's address space. "OGNSEL" (active low) buffers these signals on to the DSP data bus.

NOTE: In order to origin correctly, the O+ signal must always make the transition at a certain joint angle with an error of at most one motor rotation. We are unable to ascertain from the available PanaRobo documentation, whether this is so. However it would be difficult to origin if this were not the case. The documentation of the robot indicates the existence of another signal — "O-". However it was found that though this signal is being generated by the potentiometer boards it is unused by the current controller. Also, the schematics of the potentiometer board do not indicate how this (O-) signal is being generated.

NOTE: Joints 5 and 6 are coupled mechanically as follows:

$$\begin{aligned} thetaJ6 &= thetaM6/Gear6 + thetaM5/(Gear5 * 4) \\ thetaJ5 &= thetaM5/Gear5 \end{aligned}$$

Therefore joint 5 must be set to origin before joint 6.

Once the arm is brought to origin, the controller is aware of the exact joint angles of the arm. It therefore uses only the incremental motion information contained in the encoder signals to keep the joint angle information updated. If, for some reason, a joint reaches its limits of operation (e.g. controller malfunction, incorrect motion commands from higher levels of control) a limit signal (LL or LH) is asserted (low) by the potentiometer board. In such an event, the drive to the robot must be aborted to prevent any further damage.

The limit signal processing is implemented as follows:

The two signals LLx and LHx (x=1–6) are open collector signals. They are wire-ORed and brought to the controller as "LLxLHx" (x=1–6) (sch#10). This signal is pulled high by the resistor pack (R16 sch#19) and input to the "limitpld" PLD (sch#17). Since the "LLxLHx" signals are memory-mapped in the DSP's address space, these signals may be

read from the "limitpld" by asserting "LMTSEL". This allows the DSP to identify the joint that reached the limit. By reading the "LLxLHx" signal and the corresponding "OGNx" signal, the DSP can ascertain whether the low or the high limit has been reached.

In order to cut off the motor drive to the joint, the "LLxLHx" signal of a joint asserts the "LIMITN" input of the HCTL which drives that joint. When the "LIMITN" input of an HCTL is asserted, the duty cycle of its PWM output is reduced to 0, thus turning off the drive to the motor. The "LLxLHx" signals may be individually masked from asserting the "LIMITN" input of the HCTL by setting a bit (one bit per joint) in the limitpld.

The ability to mask the limit signal must be used with caution, because masking this signal defeats its protective function. The sole intent of providing this ability is to ease the task of recovering from a limit exception.

In the event of a limit violation of a joint, the limit signal corresponding to that joint would be asserted and the HCTL driving that joint would switch off the drive to the motor. In order to recover from such an exception the "LIMITN" input of the HCTL would have to be disasserted. If the "LLxLHx" signal could not be masked, recovery would require that the controller and the servo amplifiers be turned off and the robot arm returned within the workspace manually. This is the case with the PanaDac controller. However, for this controller, recovery is eased by allowing the DSP to mask the "LLxLHx" signals. The "LLxLHx" signal of the joint that has overrun, can be masked temporarily to cause the LIMITN input of the corresponding HCTL to be disasserted. The HCTL can then be instructed to move the joint back into the workspace. Once the joint is within bounds, the potentiometer boards disassert the "LLxLHx" signal. The "LLxLHx" signal can then be unmasked, to restore its protective function, and normal operation can be resumed.

The "limitpld" also generates a "LMTINTR" when any "LLxLHx" signal is asserted. "LMTINTR" is one of the sources of interrupts to the DSP and a bit in the DSP's status register. If the DSP is DSP32C, the interrupt facility may be used; however, if the DSP is DSP32, "LMTINTR" must be polled.

NOTE: The limits of operation of each joint may be individually changed by adjusting the pots on the potentiometer boards. Care must be taken to ensure that the new limit lies within the workspace and not beyond the physical limits of the joint's motion.

4.2 The Encoder Signal Interface

The rotary encoder on each joint provides information regarding the incremental motion of the joint, encoded in two quadrature pulse trains. The relative phase of the pulse trains denotes the direction of motion, and the width of a pulse in either train is inversely proportional to the joint's angular velocity. The encoders are connected directly to the motor shaft (rather than the joint shaft which is geared down) and generate a large number of pulses per motor rotation (2000 ppr). This enables them to resolve virtually undetectable changes in the joint angle. (about, [360 / (gear ratio x 2000 x 4)] degree).

These encoder signals (or quadrature pulse trains as they are sometimes called) are used to update a set of counters which maintain the count that represents the joint angle. These counters are set to a known value at origin and count up or down as governed by the encoder signals. The counters reside in the HCTL and can be read and written by the DSP.

The rotary encoders also generate a third signal called the index signal. The index signal asserts a pulse once every motor rotation. The occurrence of this pulse is used during origining since it marks a position of the motor shaft. The limit signal can also be used to correct for lost couont. By observing this signal and by reading the position counter value at the occurrence of the index pulse, the fidelity of the position count to the true count may be verified. If the encoder count is not a multiple of 2000 (the number of pulses per encoder shaft rotation) the position count may be corrected by modifying the HCTL counter to accommodate for lost count. Encoder counts may be lost due to electrical noise on the encoder channels or due to mechanical nature of the way in which the channel signals are generated.

The encoder channels, at the rotary encoders, are 250mV peak to peak sinusoidal signals at the output of the encoder. The signal conditioning board, placed at the base of the robot, converts these signals to TTL level pulse trains. The conditioned channel signals (JxENCy, x=1–6, y=A,B) are input directly to the HCTLs. (The self test circuits may be considered transparent for the moment.)

Unlike the encoder channels, the index signal is unconditioned. It is normally 5V and asserts an active low pulse. The six index signals (JxENCIX x=1–6) (one per joint) are input to the "IxPls_Ogn13 & IxPls_Ogn46" PLDs. The IxPls_Ognxx PLDs are memory-mapped in the DSP's address space.

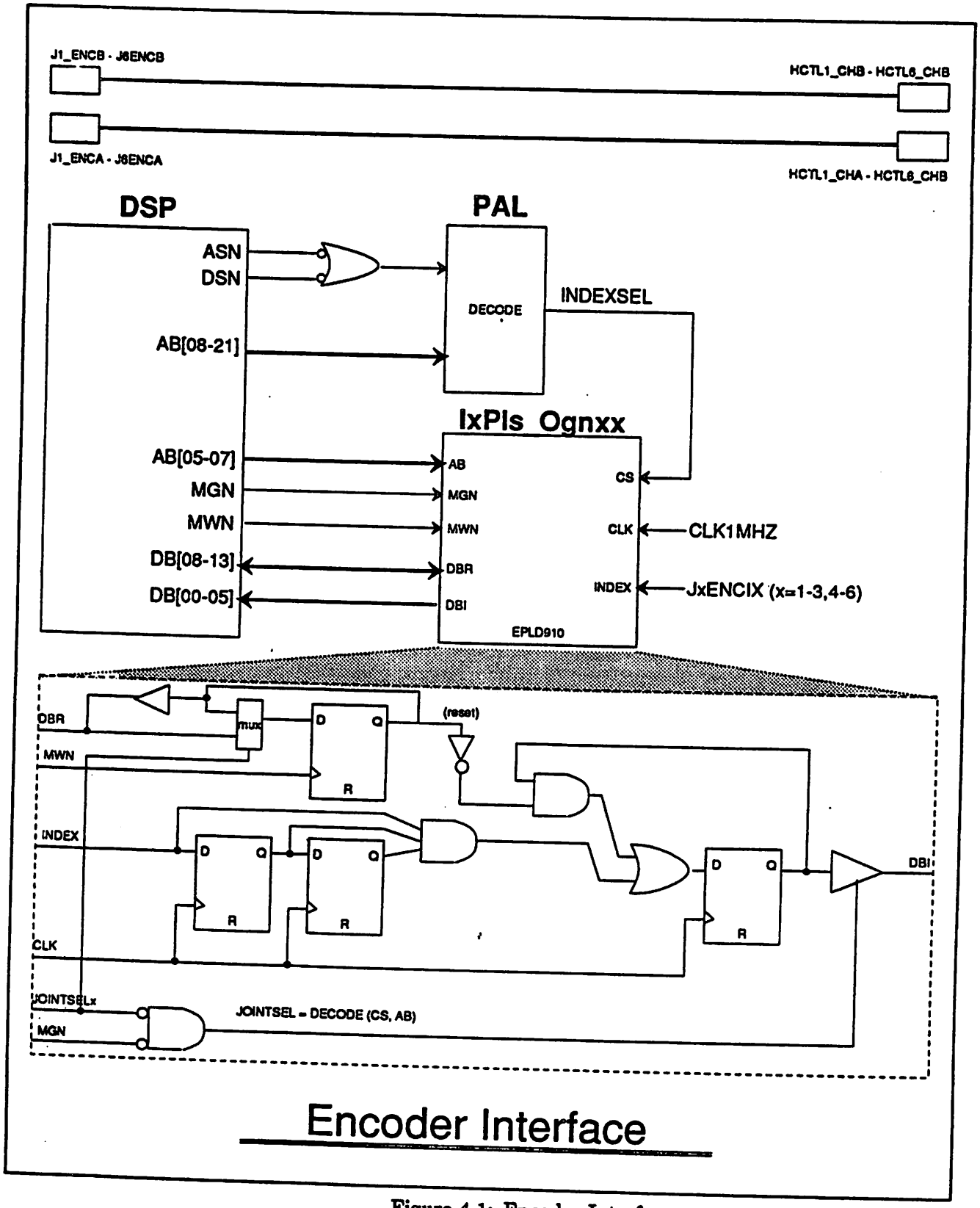


Figure 4.1: Encoder Interface

The occurrence of an index pulse is captured by a "ones-catching" circuit (actually a "zeros-catching" circuit, since the index pulse is active low) implemented within these PLDs (Fig 4.1). This implementation is a simple two state finite state machine (FSM). (See IxPls_Oxgnxx PlD.) The FSM makes a transition from State 0 to State 1 at the occurrence of the index pulse. It returns back to state 0 if the index signal is disasserted and a reset signal is asserted. The reset signal is asserted by the DSP by writing a <1> in the reset latch. (Using the asynchronous resets of flip-flops is considered one of the deadlier sins of digital design - therefore this more expensive scheme is used).

The way to locate an Index Pulse mimics a binary search algorithm. The DSP first clears the latch (corresponding to a certain joint) in the IxPls_Ognxx PLD. The joint is moved by half a motor rotation. The latch (FSM) is read to ascertain if the index pulse had been issued during that motion. If it was issued; the latch is cleared and the joint moved back by half the earlier motion (a quarter motor rotation). If it wasn't; the joint is moved further by half the earlier motion (i.e. a quarter motor rotation) and this process is continued until the index pulse position is found. This algorithm leads to the pulse in at most (log(base2) n) trials, where n = no. of ppr.

NOTE: Since the index pulse is generated by a mechanical device, it is qualified (within the PLD) by sampling it at 3 rising edges of a 1MHz clock (CLK1MHz). (The index pulse was measured to be 10us wide at the maximum operating speed of the existing controller, thus qualifying it for 2us was considered acceptable).

4.3 The PWM Signal Interface

The PWM signals are the means by which the controller commands the servo amplifiers to drive the motor. The direction and the magnitude of the drive to be applied to the motors is encoded in these signals. Of the several ways in which this encoding can be done, the two of concern are discussed here.

The HCTL produces a PWM output on two signals: "Pulse" and "Sign". "Pulse" is a pulse train whose duty cycle is modulated. The average (DC) value of "Pulse" represents the magnitude of the drive to be applied to the motors. "Sign" is a binary signal that denotes the direction.

The servo amplifiers require that the information contained in "Pulse" and "Sign" be provided in a different format. A functional diagram (Fig 4.2) of the servo amplifiers

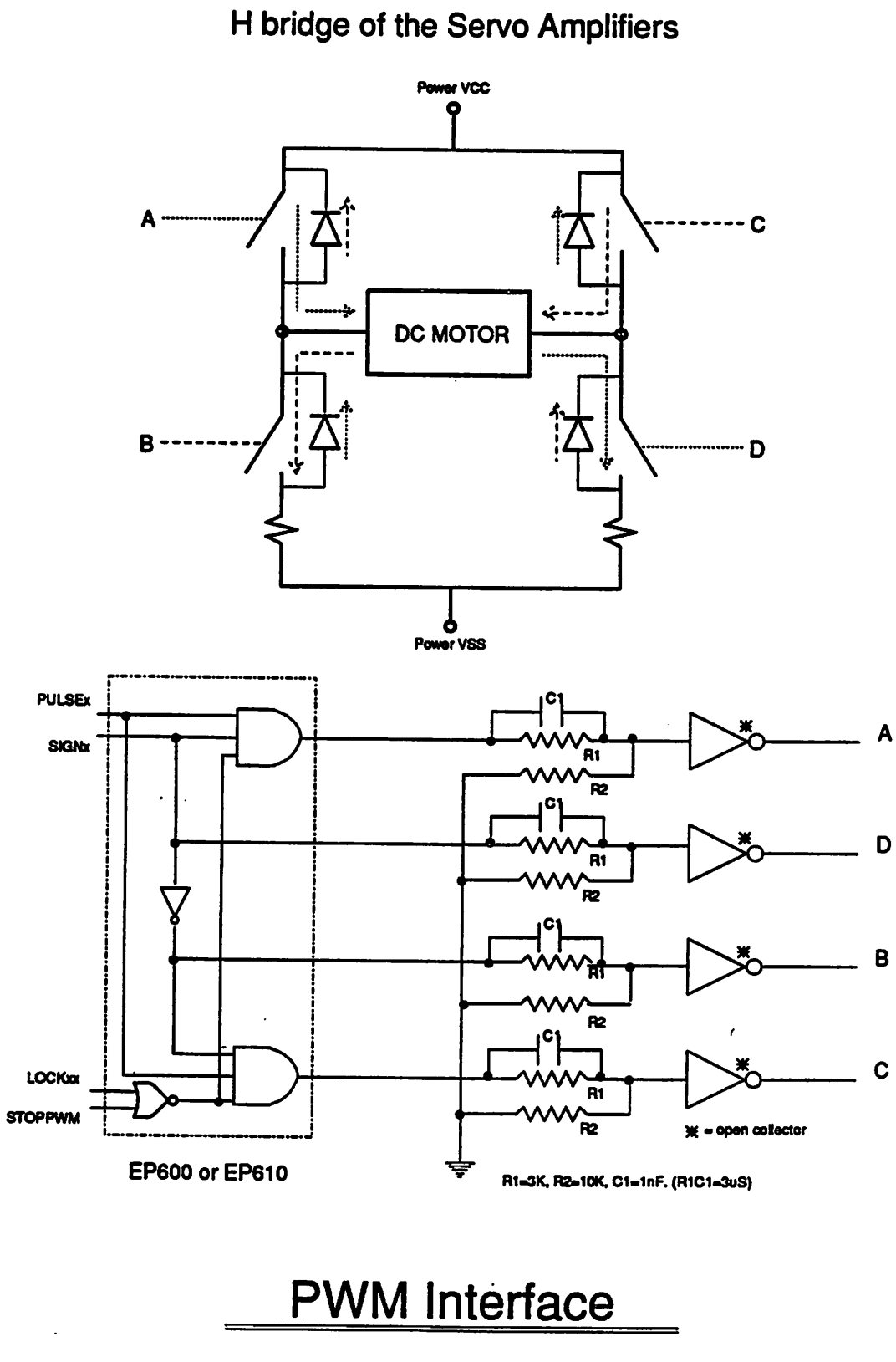


Figure 4.2: PWM Interface

will help illustrate this better. The servo amplifier is an H-Bridge. The motor is connected at the horizontal edge and four "switches" are in each of the vertical edges. These four switches are driven by four control signals: A, B, C, & D. By closing switches A and D, current in the motor flows in one direction; and by closing the other two instead, the direction of current in the motor is reversed. Since the motor is largely inductive, an abrupt change of current must be avoided. The diodes (spark killers) across the switches serve to maintain current continuity in the motor when the switches are opened. The average value of voltage applied to the DC servo motors is proportional to the duty cycle of the control signals (A&D and B&C). Thus a Pulse Width Modulated (PWM) signal can control the voltage drive to the motor.

For the HCTL to be able to drive the servo amplifiers a format conversion from "Pulse" and "Sign" to "A, B, C, D" is necessary. This format conversion is done by the PulseNsignxx PLDs (sch#12&13). The output of the PLD is used to drive common emitter transistors to produce the open collector signals (Jxy, x=1–6, y=A,B,C,D) to be input to the servo amplifiers. The pull-up resistors of these signals are in the servo amplifiers. This scheme ensures that the robot arm retains sanity even if the connection between the controller and the servo amplifiers is disturbed (e.g. loose cable connection).

Since the PWM output directly controls the drive to the motors and yet is a low power signal, it is a convenient place to exercise emergency power shut off. The LOCKALL or LOCKxx pin of the PulseNsignxx PLDs can be asserted (high) to shut off the PWM output of the controller. The LOCKALL pin is controlled by a bit in the DSP's control register and the LOCKxx pin is asserted when a feedback signal (Ierror) from the servo amplifiers, indicating excessive motor currents, is asserted.

4.4 The Current Signal Interface

The output of most control algorithms is the desired torque, or in the other words, the desired motor current. If the I-V relationship of the motor is precisely known, the motor current can be controlled by controlling the applied voltage to the motor. In such a case, the PWM output of the controller can effectively accomplish the desired control. However, better control of the motor current is obtained by implementing a closed loop current control scheme. In order to do so, the actual motor current must be measured. Provision has been made to enable the controller to measure the current in the joint motors. The

loop will be closed in software, by comparing the desired current with the actual current and appropriately modifying the PWM output.

The servo amplifiers provide current feedback information via the two signals, I+ and I-. These signals are the voltages across one-tenth ohm resistors placed in each of the lower vertical legs of the H bridge. When the current in the motor flows in one direction, it flows through one of the legs and develops a voltage across one of the resistors, while the voltage across the other resistor is zero. When it flows in the opposite direction, it flows through the other leg of the H bridge and develops a voltage across the other resistor. Thus, the voltage difference between I+ and I- is equal to the instantaneous current flowing in the motor.

The PanaDac controller implemented an analog filter to detect, amplify and filter the effective current signal — (I+,I-). SPICE simulation of this circuit was performed and the results were:

Differential Gain (at 100Hz) = 11.1

3 dB Frequency = 1.1kHz

Roll off = 20dB / decade (in the frequency range 1kHz - 10kHz)

These specifications are suitable for the following reasons:

Since the motors are rated at a maximum of 4 Amperes, the signals I+ and I- are expected (and verified) to be about 0.4 Vmax. This gives an effective current signal output of 4.4 Vmax - which is an adequate range for the analog to digital converter (described later in this chapter).

The motor response was calculated (using the motor parameters provided in the PanaSonic documentation and a simple DC motor tranfer function) to be about 700 Hz, therefore a 3 dB filter bandwidth of 1.1 kHz was reasonable. In the event that motor non-linearities are found to be important the filter parameter could be easily altered by changing the values of the discrete components.

Filtering the noisy signal from the motors via digital means would have meant undesirable computational load on the DSP besides requiring a high sampling rate to reduce aliasing effects. The analog implementation pre-emptied the need for this additional processing load. (A solution employing switched capacitor filters was investigated but the implementation difficulties precluded their use. Moreover, analog techniques to eliminate the clock feed-through would still have to be used.) This filter configuration has therefore been implemented using operational amplifiers and discrete components. The signals I+

and I- from the servo amps are named JxIP and JxIN x=1–6, on connectors PWM1 & PWM2. The reference for each of these signals is JxGRND, x=1–6.

The analog output signal of the filter (JxI, x=1–6) represents the instantaneous current in the motors. This signal is converted to a digital form using Microlinear's ML2208 12 bit multi-channel analog to digital converter. ML2208 is a versatile "Data Acquisition Peripheral" (DAP) — so called for its built in analog data acquisition and digital data processing capability. It is a 12 bit converter with a conversion time of 31.5 us at a clock input frequency of 7 MHz. The data sheet elaborates on all the unique features of the DAP, however those of significant interest are:

- 8 channel input : This permits the use of a single AtoD to sample all six motor current signals. This results in a significant savings in board area.
- Fast conversion speed : Although the DAP is rated at 7 MHz, it is operated at 8 MHz to give a conversion time of 25 us for 10 bit accuracy. (This trade-off was indicated permissible by Prof. Gray, under whose guidance this chip was designed.) Operating at 8MHz provides a sampling rate per joint of about 7 kHz. This is sufficient since the filter's 3 dB bandwidth is only 1.1 kHz.
- Advanced microprocessor interface : The DSP-DAP interface is simple and requires only a small amount of additional logic (to generate the SRDYN signal - explained below). The DAP's advanced on-chip data processing capabilities allow the DSP to program it to sequence through routines that sample and process data for all six joints.

The DAP is thus extremely well suited for the task. For its interface to the DSP, the DAP requires circuitry to generate the SRDYN signal for the DSP. SRDYN allows a slow peripheral to introduce wait states in the DSP's access cycles. (Available only in the DSP32C. DSP32 does not allow the introduction of wait states in its access cycles. It can interface to the DAP only if the DSP32's clock frequency is 12 MHz or lower.) The read and write cycle times for the DAP require the introduction of 8 wait states in the memory cycle of a DSP32C's operating at 50MHz. The circuit that implement in this is the "testenc" PLD. (If the DSP32C is operated at a different frequency this circuitry can be easily modified by reconfiguring the "testenc" PLD.) Besides the normal memory-like interface for data transfer, the control signals (RSTA2DN, A2DSYNC) needed for the DAP and the status signals (A2DINTR, A2DDBR) it generates, are communicated to the DSP

via the DSP's control register and status buffer respectively. Also, a voltage regulator is provided to derive the -5V supply to the DAP from the -12V provided on the VMEBus.

As mentioned in the previous section, the servo amplifiers provide a signal (Ierror) IERx, x=1–6, that is asserted when the motor current exceeds allowable limits. This signal is the output of a thyristor that is set to trip at the maximum motor current level. The IERx signals are provided as input to the "Ierror16" PLD where they are latched. By reading these latches the controller can ascertain the joint that caused the failure and provide the user with appropriate diagnostics. The latches are all reset by the DSP, by setting a bit in the Ierror16 PLD.

The assertion of the IERx signal constitutes an emergency, and the power supply to the motors must be cut off. Therefore, when any one of the IERx signals is asserted the Ierror16 PLD asserts a "LOCK" signal that is input to the servo amplifiers. The LOCK signal shuts off the power supply to the motors and applied the brakes to halt the arm. The LOCK signal is one of the sources of interrupt to the DSP and is a bit in the DSP's status register. If the DSP used is DSP32C the interrupt facility may be used, however, if the DSP used is DSP32, "LOCK" must be polled.

4.5 The HCTL Interface

The HCTL 1000 is a general purpose motion control chip that is used for controlling the motion of a joint. It accepts the rotary encoder signals from the robot to maintain the joint position and provides a PWM output to drive the servo amplifiers. The PWM output produced by the HCTL is dictated by a built-in control function whose coefficients are programmable by the DSP. The controlled joint's position and velocity are maintained in registers that can be read and modified by the DSP. The functionality of the HCTL can be altered by the DSP, by writing into the registers in the HCTL. Thus the HCTL appears as a read-write register file to the DSP.

Though functionally the HCTL may appear to the DSP as a register file, an elaborate interfacing scheme is required to allow the two to communicate with each other. (Fig 4.3) This is due to the two orders of magnitude difference in the operating speeds of the two devices. The DSP's access cycle time is tens of nanoseconds while the HCTL access cycle time is a few microseconds.

While it would still be possible to interface the HCTL to the DSP without much

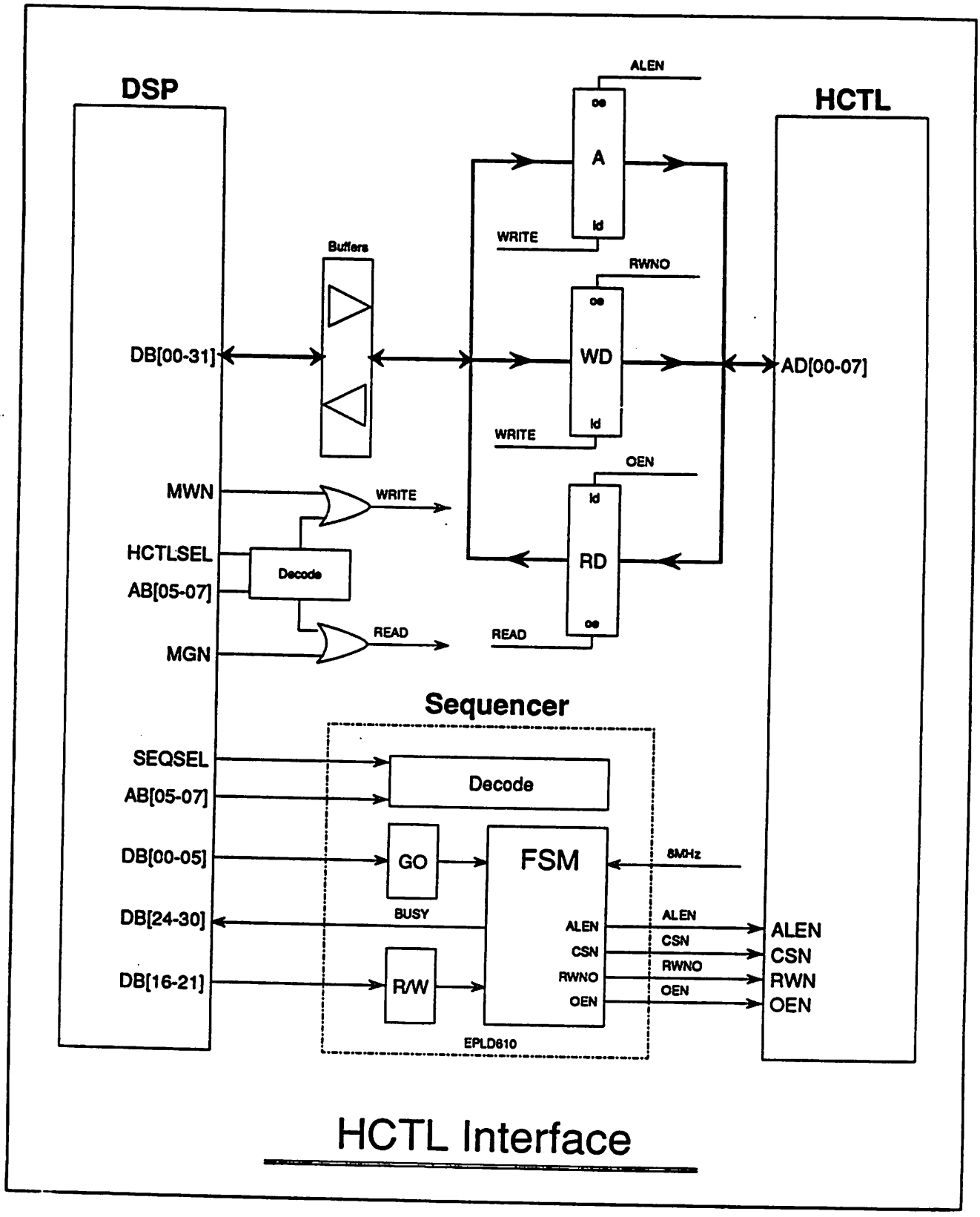


Figure 4.3: HCTL Interface

additional logic by merely introducing wait states in the DSP's access cycles, this would have meant an enormous waste of computing power. Instead the following communication scheme has been implemented.

Central to the interfacing scheme is the "sequencer". It is a finite state machine (implemented in a PLD) that orchestrates the data transfer between the two devices. (Fig 4.4)

To write into one of the HCTL registers, the DSP writes the address of that register (6 bits) in the "address latch" and the data to be written in the "write data latch". It then writes two bits in the sequencer. One bit (RWNI) indicates that the transaction is a write (RWNI=0) and the other bit (GO) triggers the sequencer to commence the data transfer (GO=1). The sequencer acknowledges the receipt of these signals by asserting a "busy" signal. The DSP polls for the "busy" and when it is asserted, the DSP disasserts the trigger signal (GO=0). This ensures that the sequencer does not restart another sequence (and thus be continually busy) when it reaches the end of the current sequence. The DSP can then continue with other processing tasks. The sequencer, in the meantime, sequences through a chain of states that assert the appropriate control signals (ALEN, CSN, OEN, RWNO) for the HCTL and effect a write in the desired HCTL register. At the end of the sequence the "busy" signal is disasserted. Here if the sequencer finds that GO=1, it waits for GO=0. When GO=0, it goes to the start state and waits for GO=1 to start another data transfer sequence.

In the case of a read cycle the DSP writes the address of the register to be read in the "address latch" and starts the sequencer (RWNI=1 and GO=1). The procedure for disasserting the trigger (GO) is as before. At the end of the read sequence (as indicated by a disasserted "busy") the contents of the specified register are transferred to the "read data latch".

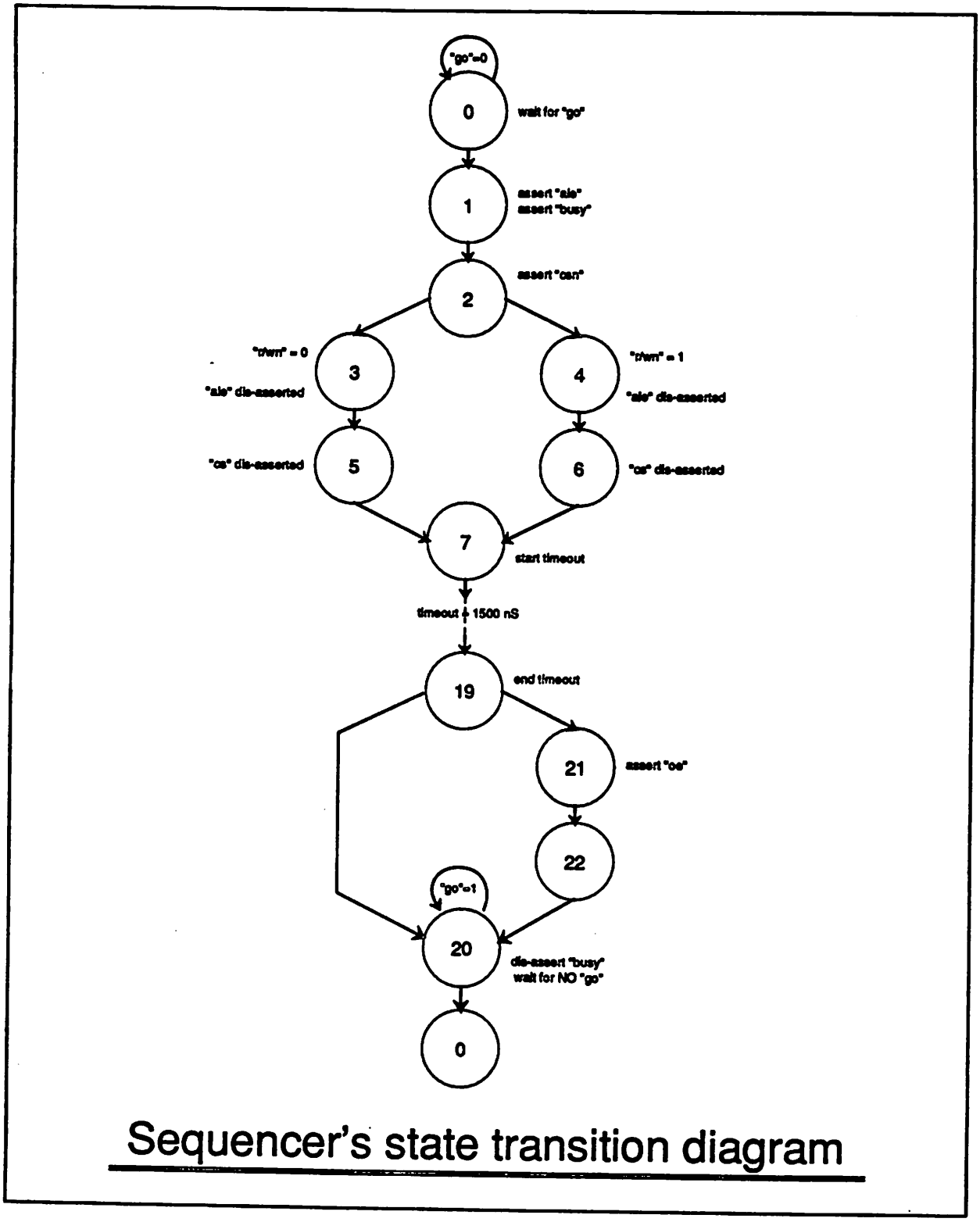


Figure 4.4: Sequencer's State Transition Diagram

Chapter 5

The Processor and the Memory Organization

5.1 The Processor

The processor on-board this controller is the WE DSP32 — a 32 bit Digital Signal Processor from AT&T. The DSP32 is a high performance programmable digital signal processor (DSP) that is suitable for use in high speed, real time, numeric processing applications. The DSP32 offers a unique set of architectural features that include: 32 bit floating point arithmetic, 16 bit integer arithmetic, 2KB on-chip ROM (mask programmable), 4KB on-chip RAM, serial and parallel I/O ports (with DMA), 4 accumulators and 21 general purpose registers. At its maximum operating clock frequency (25MHz), the DSP32 executes 6.25 MIPS and 12.5 MFLOPS concurrently.

The DSP32 is supported by a C Compiler, Support Software Library and an Application Software Library. A software simulator and a DSP32 development system are also available to ease the development of software and hardware for DSP32-based system.

This powerful processor, however, has a limited address space (56KB external memory) and offers no interrupt handling capability.

The DSP32C, a successor to the DSP32, overcomes these limitations and surpasses its predecessor in processing capabilities. It offers all the features of the DSP32, plus 24 bit integer arithmetic capability and a larger on-chip memory (6KB RAM or 4KB RAM and 4KB ROM). The address space of the DSP32C is 16MB and it responds to 2 external and 6

internal individually maskable interrupts. Operating at 50 MHz - *twice* the clock frequency of the DSP32, it processes data at the rate of 12.5 MIPS and 25 MFLOPS - *twice* that of the DSP32.

The high performance offered by the DSP32 (in both integer and floating point processing), the availability of support software and hardware, and the possibility of upgrading the DSP32 based system with the DSP32C, were some of the reasons for its choice. The choice of the DSP32 also helped share development effort with other contemporary projects which were using the same processor. Other commercially available digital signal processors considered for the task at hand, include Motorola's DSP560xx, NEC's upd77Cxx and Texas Instruments' TMS320xx.

At the time of the design of this controller, the DSP32 was available, while the DSP32C was still in the development stage. Only advance information on the DSP32C had been released by AT&T. The controller board was therefore designed to function with *either* processor. This was done to enable the board to be functionally tested using the DSP32, at the same time allowing the use of the features of the DSP32C when it became available. The board provides full functionality under the constraints of the DSP32 and exploits the additional features (larger address space, interrupts) offered by the DSP32C.

The block diagrams of the DSP32 and DSP32C are provided in Appendix A.

5.2 Organization of the Address Space

As mentioned above, the controller board is designed to function with either the DSP32 or the DSP32C. The two processors have very different address spaces as shown in Fig 5.1. The organization of the address space of the each processor is explained below.

The DSP32 has 14 address pins, allowing it to address 16K distinct locations (longwords). The memory select signals MSN[3:0] access one or more of the four bytes in each location (longword). This gives a total address space of 64KB. Different amounts of this space are consumed by the on-chip memory and reserved space, depending on the configuration set by the memory mode pins (MMD[1:0]). In memory mode 2, 8KB are occupied by on-chip memory and reserved space, leaving 56KB to external memory. (Mode 2 maps address location 0 to external ROM, enabling the DSP to boot from an external memory. This is therefore the memory mode of interest.) Also, since the DSP32 has no separate I/O space, both the external memory and the I/O have to be mapped in the

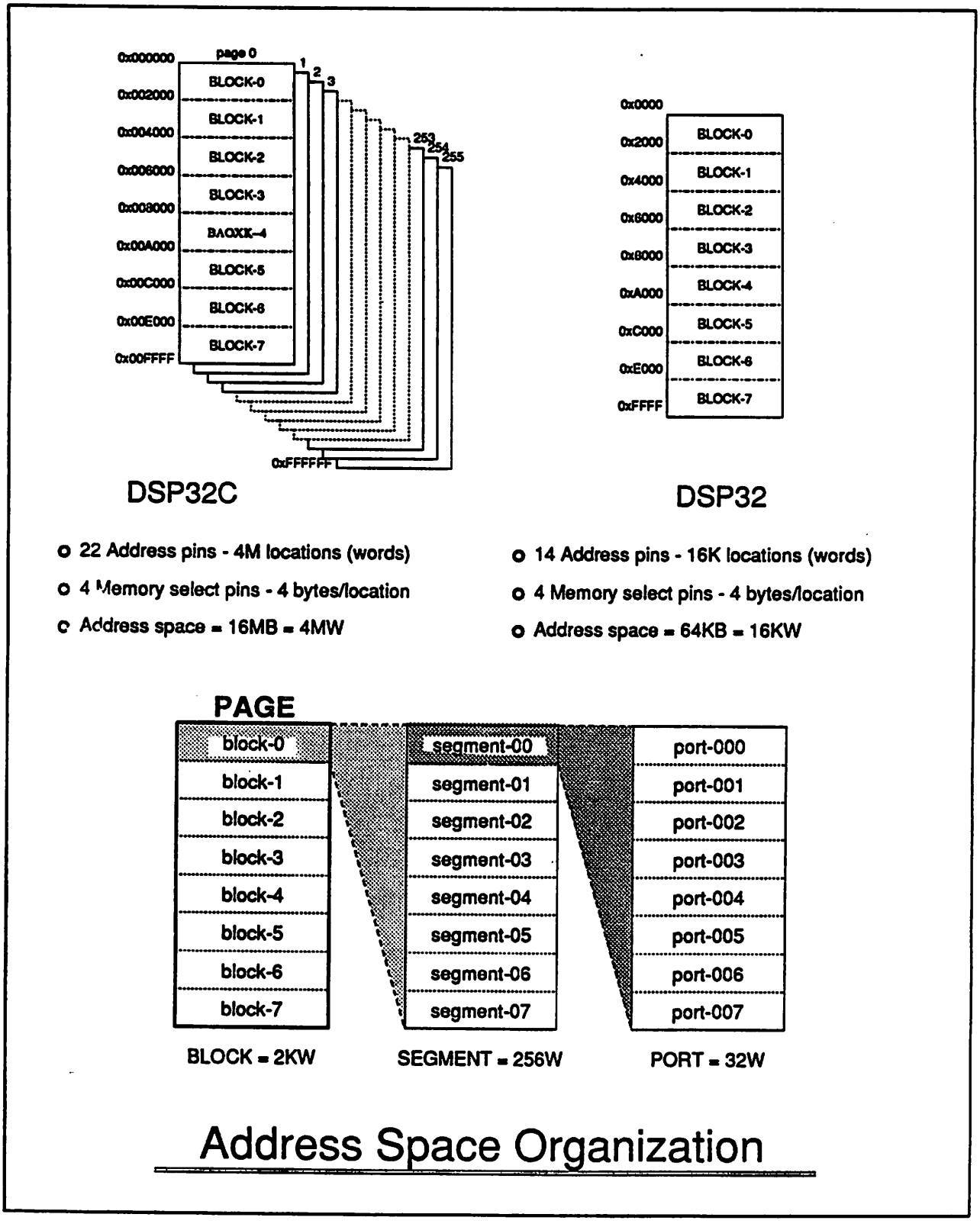


Figure 5.1: Address Space Organization

available 56KB space.

The DSP32C has a much larger address space which can be configured in 8 different ways by setting the 3 memory mode pins, MMD[2:0]. When MMD2 = 0, the DSP32C emulates the DSP32 address space. When MMD2 = 1, the 22 address pins of the DSP32C span a 4MegaWord (4 MWords) address space which the memory select signals, MSN[3:0], further resolve into 16MB. As with the DSP32, different amounts of this space are consumed by the on-chip memory and reserved spaces. The DSP32C is also available in a ROMless version in which the 4KB internal ROM is substituted by an additional 2KB internal RAM. In mode 6 of the ROMless version (where location 0 maps to external memory) the upper 8KB of the address space are occupied by the internal memory and reserved space. The rest of the address space is partitioned as "external memory A" and "external memory B." The DSP32C can be configured to introduce different number of wait states in its access cycles when accessing locations in either of these partitions. As is the case with the DSP32, the I/O devices have to be memory-mapped.

The memory mode configurations of the DSP32 and DSP32C are provided in Appendix A.

For the purposes of this report, the processor's address space is *conceptually* organized as follows. (see Fig 5.1). (The address space of the two processors is measured in terms of longwords (4 bytes) rather than bytes, since the address lines address a longword.) The address space of the DSP32C (16MB) has been divided into 256 pages, each spanning 16K LWords (64KB). One memory page of the DSP32C is thus equal to the total address space of the DSP32 (64KB). A page is divided into 8 blocks of 2K LWords (8KB) each. A block consists of 8 segments of 256 LWords (1KB) each and a segment consists of 8 ports of 32 LWords (128B) each. The page address is specified by address pins 21–14, the block address by pins 14–11, the segment address by pins 10–8 and the port address by pins 7–5. The complete port address can be specified as pageX:blockX:segmentX:portX. The following section uses this nomenclature to describe the memory map.

5.3 The Memory Map

The on-board memory consists of Erasable PROM (EPROM), Static RAM (SRAM) and Dual Port RAM (DPRAM). (see Fig 5.2.) Each of these is independently discussed in the following sections.

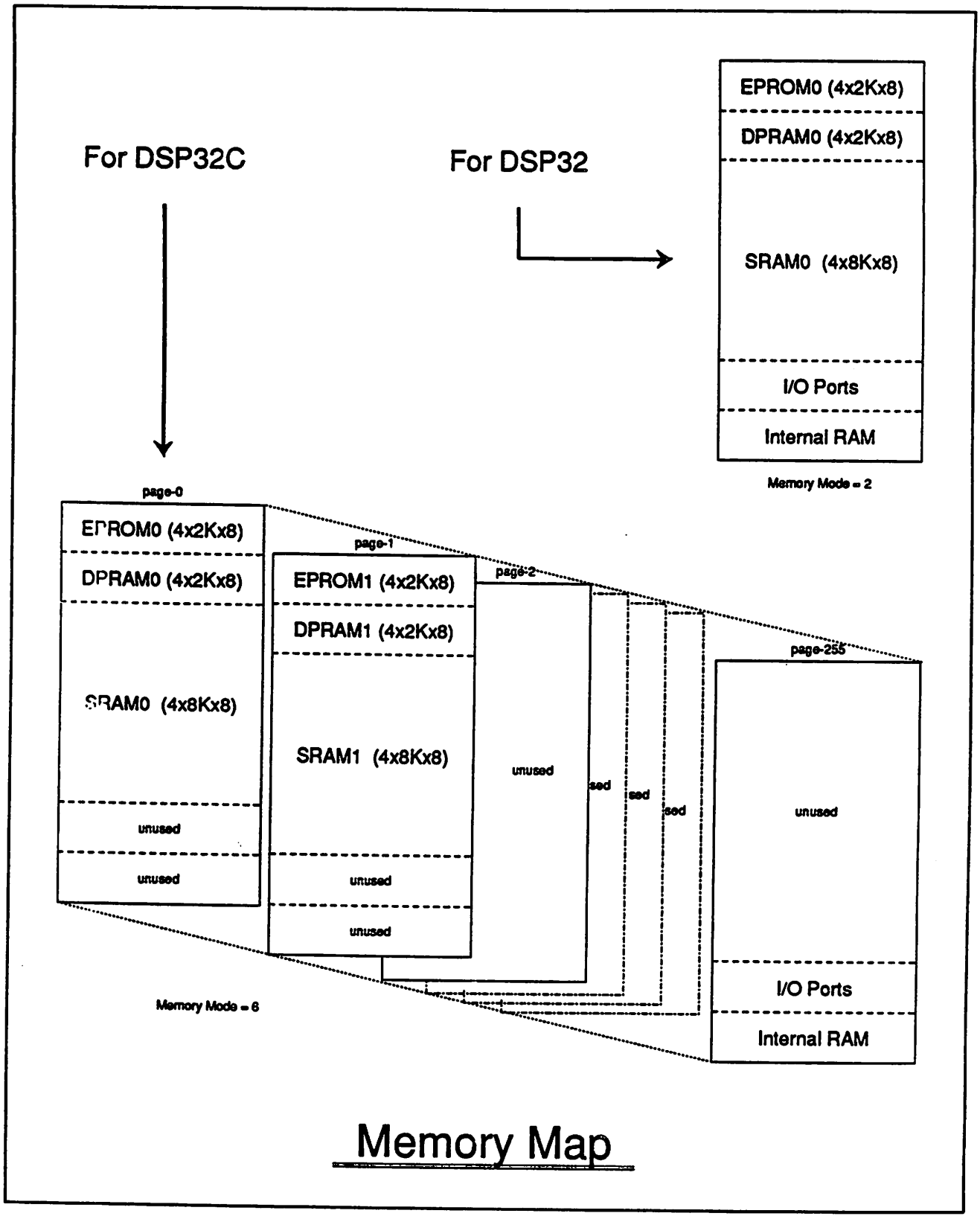


Figure 5.2: Memory Map

An issue common to these memories is the choice of their size. The size (quantity and word width) of these memories was governed by two criteria. Firstly, in order to make the board functional using the DSP32, the memories had to be small so that they (along with the I/O devices) could be accommodated in the small memory space of the DSP32. Secondly, since the DSP32 did not support wait states, the memories had to be fast and therefore small.

The Erasable Programmable ROM (EPROM)

The EPROM serves to boot the controller board and is therefore placed starting at address location 0 in the memory spaces of either processor. Two banks of EPROM of 2K LWords each have been provided. Only bank 0 of the EPROM is mapped in block 0 of the DSP32 address space. The additional bank is useful when using the DSP32C and is mapped to page1:block0. Each bank is implemented using 4 TMS27C292 (2K x 8) EPROMs. The EPROM is byte addressable.

The Static RAM (SRAM)

The SRAM is the local memory of the DSP. Two banks of 8K LWords each are provided on-board. When using the DSP32, bank 0 is mapped to blocks 2, 3, 4 and 5 and bank 1 is unavailable. When the DSP32C is used the bank 0 maps to page0:block2,3,4,5 and bank 1 to page1:block2,3,4,5. The SRAM is implemented using 4 CY7C185 (8K x 8) SRAMs. The SRAM is byte addressable.

The Dual Port RAM (DPRAM)

The DPRAM is provided to function both as local memory for the DSP and to enable the VMEbus host to communicate with the DSP without interrupting the operation of the DSP. The DPRAM is thus accessible to the DSP on the left port and to the VMEbus host on the right port. Two banks of 2KWords each of DPRAM have been provided. Bank 0 is mapped to block 1 of the DSP32 address space and bank 1 is unavailable for the DSP32. For the DSP32C bank 0 is mapped to page0:block1 and bank 1 to page1:block1. On the VMEbus side, the two banks are mapped consecutively, starting at address location 0x900000H in the VMEbus's standard address space. Each bank of the DPRAM is implemented using 4 VT7132As (2K x 8). The DPRAM is byte addressable by the DSP but is only word addressable by the VMEbus.

Since the DPRAM is accessed by two independent processors operating asynchronously — the DSP and the VMEbus host, there is a possibility of data contention.

This can happen when data accesses by both processors is to the same location in the DPRAM and is coincident or overlapping in time. Contention can either corrupt stored data during write or provide false data during read. It must therefore be avoided. In this controller, the onus of contention resolution is largely on the software. However, hardware "primitives" have been provide to ease this task.

In order to resolve contention, it must first be recognized, by the port that loses during contention, and then be corrected. Provision has been made to enable the processor on each port of the DPRAM to recognize contention. This is done as follows. The DPRAM chips provide a "busy" signal for each port. The "busy" is asserted if the port accesses a certain location that is being accessed by the other port. If during a DPRAM access (defined by active chip select) the "busy" signal of a port is asserted, a flip-flop corresponding to that port is set. By reading this flip-flop, it is possible for the processor on that port to recognize contention. By clearing this flip-flop, the stage is set for another access. The status buffer and the control register provided for each processor (detailed below) have been used to enable the reading and clearing of this flip-flop. The output of the flip-flop is provided as a status buffer bit, and the flip-flop can be cleared asynchronously by setting a bit in the control register. The logistics of this scheme are: the processor clears the flip-flop by writing to its control register (two writes are necessary), makes a DPRAM access, and reads the appropriate status register bit to identify if contention occurred during the DPRAM transaction. This scheme has a substantial overhead and must be used only to ensure the passing of critical flags between the VMEbus host and the DSP. Contention avoidance during routine data transactions must be handled by setting a communication protocol in software.

(NOTE: Since the DPRAM is byte-addressable by the DSP, all the DPRAM chips must be masters, i.e. they must all be capable of identifying contention. A single-master-multiple-slave configuration cannot be used.)

5.4 The I/O Map

The DSP32 and DSP32C do not have a separate memory and I/O address space. I/O devices therefore have been memory-mapped. The I/O devices, and their enable signals (in parenthesis) are: the IxPls_Ognxx PLDs (IXSEL), the limitpld PLD (LMTSEL), the ML2208 DAP (A2DSEL), the HCTL1000s (HCTLSEL), the Sequencerx PLDs (SE-

QSEL), the control registers (CSDSPCTRLN & CSTESTCTRLN) and status buffers (CSDSPSTATN & CSTESTSTATN), the VMEpld, the Ierror16 PLD (IERSEL) and the Dual universal asynchronous receiver transmitter - DUART (CSDUARTN).

The I/O devices are all mapped in block 6 of the DSP32 address space and in page255:block6 of the DSP32C address space. In most cases, signals of joint 1 through 6 are mapped to port 1 though 6 respectively. Sometimes port 0 is used to access the particular signal for all joints simultaneously. "byte 0" refers to least significant byte — LSB (DATA[7:0]) of the DSP data bus and "byte 3" to the most significant byte — MSB (DATA[31:24]) of the DSP data bus. Similarly, "byte 1" and "byte 2" refer to DATA[15:8] and DATA[24:16] respectively.

The Index Pulse and Origin PLDs (IxPls_Ognxx)

The IxPls_Ognxx are mapped in block6:segment0. Joints 1–6 map to ports 1–6 respectively. All joints are mapped in port 0 — i.e. all joint index signals may be simultaneously read from and all reset signals to FSMs simultaneously written to port 0. The index signals are read from byte 0 of the data bus and the reset signals are read and written to on byte 1 of the data bus. The origin signals are read only, and are provided on byte2 of the data bus. (please see Table 5.1)

The Limitpld PLD

The limit signals are mapped in segment1. Joints 1–6 map to ports 1–6. Port 0 and 7 are unused. The limit signals are provided on byte 0 and the limit mask signals are provided on byte 1 of the data bus. The origin signals may be read on byte 3 of the data bus. (please see Table 5.2)

The ML2208 Data Acquisition Peripheral (DAP)

The ML2208 DAP is mapped in segment2. The 8 registers in the DAP are mapped to ports 0–7. These registers are accessed on byte 3.

The HCTL 1000s

The registers needed to communicate to the HCTL are mapped in segment3. Joints 1–6 are mapped to ports 1–6. For joints 1–3, the "address" registers are mapped to byte 1, and the "data read" and "data write" registers are mapped to byte 0. For joints 4–6, the "address" registers are mapped to byte 4, and the "data read" and "data write" registers

DATA BUS BITS

BLOCK - 6: SEGMENT - 0.

```
PT #:|-----byte3-----|-----byte2-----|-----byte1-----|-----byte0-----|
      7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
PT 0: . . . . . . . . . . r r r r r r . . x x x x x x . . r r r r r r All-J
PT 1: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r  J1
PT 2: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r .  J2
PT 3: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . .  J3
PT 4: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . .  J4
PT 5: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . . .  J5
PT 6: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . . . .  J6
PT 7: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                    | Origin Signal | Reset IndxPls |  Read IndxPls |
```

legend:

"x"- (Read/Write), "r"- (Read only), "w"- (Write only), "."- (unused)

Table 5.1: Index Pulse and Origin Signals

DATA BUS BITS

BLOCK - 6: SEGMENT - 1.

```
PT #:|-----byte3-----|-----byte2-----|-----byte1-----|-----byte0-----|
     7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
PT 0: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
PT 1: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r  J1
PT 2: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r .  J2
PT 3: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . .  J3
PT 4: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . .  J4
PT 5: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . . .  J5
PT 6: . . . . . . . . . . . . . . . . r . . . . . . . x . . . . . . . r . . . . .  J6
PT 7: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                    | Read Origin |   Mask Limit  | Read Limit   |
```

legend:

"x"- (Read/Write), "r"- (Read only), "w"- (Write only), "."- (unused)

Table 5.2: Limit Signals

DATA BUS BITS

BLOCK - 6: SEGMENT - 3.

```
PT #:|-----byte3-----|-----byte2-----|-----byte1-----|-----byte0-----|
     7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
PT 0: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
PT 1: . . . . . . . . . . . . . . . . x x x x x x x x x x x x x x x x  J1
PT 2: . . . . . . . . . . . . . . . . x x x x x x x x x x x x x x x x  J2
PT 3: . . . . . . . . . . . . . . . . x x x x x x x x x x x x x x x x  J3
PT 4: x x x x x x x x x x x x x x x x . . . . . . . . . . . . . . . .  J4
PT 5: x x x x x x x x x x x x x x x x . . . . . . . . . . . . . . . .  J5
PT 6: x x x x x x x x x x x x x x x x . . . . . . . . . . . . . . . .  J6
PT 7: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     | J4-J6 Address |   J4-J6 Data  | J1-J3 Address |  J1-J3 Data   |

legend:
"x"- (Read/Write), "r"- (Read only), "w"- (Write only), "."- (unused)
```

Table 5.3: HCTL Registers

are mapped to byte 3. This was done to distribute the loading on the data bus buffers. (please see Table 5.3)

The Sequencers

The sequencers are all mapped in segment4. The sequencers for joints 1–6 are mapped to ports 1–6. All sequencers can be simultaneously accessed to port 0. The busy signals from the sequencers are mapped to byte 0. The read/write command signals and the trigger (GO) signal from the DSP are mapped to bytes 2 and 3 respectively. (please see Table 5.4)

To specify the address of the miscellaneous I/O devices that follow, the following

DATA BUS BITS

BLOCK - 6: SEGMENT - 4.

```
PT #:|-----byte3-----|-----byte2-----|-----byte1-----|-----byte0-----|
      7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
PT 0: . . w w w w w w . . w w w w w w . . . . . . . . r r r r r r All-J
PT 1: . . . . . . . w . . . . . . . w . . . . . . . . . . . . . . r  J1
PT 2: . . . . . . w . . . . . . . w . . . . . . . . . . . . . . r .  J2
PT 3: . . . . . w . . . . . . . w . . . . . . . . . . . . . . r . .  J3
PT 4: . . . . w . . . . . . . w . . . . . . . . . . . . . . r . . .  J4
PT 5: . . . w . . . . . . . w . . . . . . . . . . . . . . r . . . .  J5
PT 6: . . w . . . . . . . w . . . . . . . . . . . . . . r . . . . .  J6
PT 7: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     |     GO=1      |READ=1, WRITE=0|              |  BUSY = 1     |
```

legend:

"x"- (Read/Write), "r"- (Read only), "w"- (Write only), "."- (unused)

Table 5.4: Sequencer Signals

Bit #	Signal Name	Function
0	test	connected to status buffer bit 0
1	CLRBUSY0LN	clears the busy F-F of the left port of DPRAM bank 0
2	CLRBUSY1LN	clears the busy F-F of the left port of DPRAM bank 1
3	FLAGVME	connected to VMEbus status register bit
4	unused	
5	unused	
6	ENDEFFOUT0	signal for end effector control
7	ENDEFFOUT1	signal for end effector control
8	RSTDUART	DUART reset signal
9	A2DPDN	DAP power down signal
10	A2DSYNC	Synchronizing signal to DAP
11	RSTA2DN	DAP reset signal
12	RSTHCTLN	HCTL reset signal
13	unused	
14	RSTIX46	IxPls_Ogn46 reset signal
15	RSTIX13	IxPls_Ogn13 reset signal

Table 5.5: DSP Control Register

convention will be used. The page:block:segment portion of the address is as before. However address pins 7–5 do not define the port, instead pin 6–4 define a subport and address pin 7 is always 0. The subports within the space defined by address pin 7 = 1 is decoded but unused.

The DSP Control Register

The DSP control register is mapped to block6:segment5:0:subport0. (see Table 5.7). This control register is a 16 bit register that can be written to from bytes 0-1 of the DSP data bus. The outputs of the control register are always enabled. The control register bits and their function are described in the Table 5.5.

The DSP Status Buffer

The DSP status buffer is mapped to block6:segment5:0:subport1. (see Table 5.7). This is a 16 bit buffer mapped on bytes 0-1 of the DSP data bus. The status buffer bits and their function are described in the Table 5.6.

The Test Control Register and Test Status Buffer

A control register is needed to generate the test signals for the testenc PLD and to monitor the outputs of the testpwm PLD. The test control register is mapped to

Bit #	Signal Name	Function
0	test	connected to control register bit 0
1	BUSY0LN	output of the F-F of the left port of DPRAM bank 0
2	BUSY1LN	output of the F-F of the left port of DPRAM bank 1
3	FLAGDSP	connected to VMEbus control register bit
4	unused	
5	unused	
6	ENDEFFIN0	signal for end effector status
7	ENDEFFIN1	signal for end effector status
8	DUARTINTR	Interrupt from DUART
9	unused	
10	A2DDBR	Data buffer signal ready from DAP
11	A2DINTR	Interrupt from DAP
12	LMTINTR	Interrupt from limitpld
13	LOCKX	Lock signal to servo amplifier
14	IX46INTR	Interrupt from IxPls_Ogn46
15	IX13INTR	Interrupt from IxPls_Ogn13

Table 5.6: DSP Status Buffer

block6:segment5:0:subport2 and the test status buffer to block6:segment5:0:subport3. (see Table 5.7). The functionality of their bits is explained in detail in Chapter 7.

The VMEbus Interrupt

The DSP interrupts the VMEbus host by setting a latch in the VMEpld. This latch is mapped in block6:segment5:0:subport4 on bit 31 of the DSP data bus. (see Table 5.7).

The Ierror16 PLD

The Ierror16 PLD is mapped to block6:segment5:0:subport5 on bit 8 of the DSP data bus. All the current error latches in the Ierror16 PLD can be reset by writing a 1 to bit <8> on this location. (see Table 5.7).

The Dual UART

The DUART is mapped to block6:segment5:0:subport7. The 16 registers within the DUART are selected by the DSP's address pins 3-0. These 8 bit registers can be accessed on byte 3 of the DSP data bus. (see Table 5.7).

DATA BUS BITS

BLOCK - 6: SEGMENT - 5. (bit <7> = 0)

SPT#	byte3								byte2								byte1								byte0								
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SPT0:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
SPT1:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
SPT2:	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SPT3:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SPT4:	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SPT5:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	r	r	r	r	r
SPT6:	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
SPT7:	x	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

SPT0:	Ctl
SPT1:	Sta
SPT2:	Tctl
SPT3:	Tsta
SPT4:	VME
SPT5:	Ierr
SPT6:	
SPT7:	UART

legend:

"x"- (Read/Write), "r"- (Read only), "w"- (Write only), "."- (unused)

Table 5.7: Miscellaneous Ports

Chapter 6

The VMEbus Interface

The VMEbus interface provides communication between the VMEbus host and the DSP over the VMEbus. The VMEbus host currently in use is Heurikon's 68020-based single board computer HK68/V20 running "VxWorks", a real-time Unix-like kernel. The VMEbus interface performs the following two important functions:

- It allows the VMEbus host
  1. to down load the compiled code for the DSP into the DPRAM, and
  2. to periodically monitor the status of the DSP program.
- It allows the controller board
  1. to interrupt the VMEbus host, and
  2. to respond to an interrupt acknowledge cycle.

To enable data communication between the VMEbus host and the controller board, the VMEbus interface supports a Data Transfer Bus (DTB) cycle. The VMEbus host can transfer data to and from the DPRAM, write to the VMEbus control register or read the VMEbus status buffer via a DTB cycle.

The generation of an interrupt on the VMEbus and the response to an interrupt acknowledge cycle is specified in the VMEbus protocol as a Priority Interrupt Bus (PIB) cycle. The VMEbus interface allows the controller board to perform a PIB cycle when so instructed by the DSP on-board.

The basic protocol of both the DTB and PIB cycles is provided in the sections that follow, however, the detailed specifications are available in [6].

6.1 The Data Transfer Bus (DTB) Cycle

Data transfer over the VMEbus between a VMEbus master and a VMEbus slave is accomplished by the following sequence of events. The master places the address and the address modifier code (AMCODE) that describe the imminent bus cycle, on the appropriate VMEbus lines. The address strobe (AS*) is asserted to indicate valid addressing information. The master then places the data on the VMEbus data bus, in the case of a write cycle, and asserts one of more of the data strobes (DS0*, DS1*) to indicate the start of a bus cycle. The slave (being addressed) then asserts a signal (DTACK*) in acknowledgement. If the cycle is a read cycle, the slave places the data on the VMEbus data bus before asserting DTACK*. If the cycle is a write cycle, assertion of DTACK* by the slave indicates to the master that the data has been written. On receiving the DTACK* the master disasserts the data strobe(s) to indicate the completion of the cycle. Bus cycle exceptions are indicated by asserting the bus error (BERR*) signal.

The VMEbus specifications present various bus cycle options, which are different flavors (e.g. block transfer cycles, address pipelined cycles) of the above mentioned basic protocol. The VMEbus interface circuit that is implemented supports a quad byte standard program or data access in the user or supervisor mode (specified by address modifier codes 39, 3A, 3D or 3E). Block transfers are not supported. While address pipelined transfers function correctly, their speed advantage is not exploited in this implementation. The board also does not assert BERR* to indicate exceptions. This implementation also ensures that all VMEbus specifications are fully met.

The general scheme of implementation is explained below. The details may be derived from the board schematic #7 and from the VMEpld schematic.

The start of a bus cycle is recognized by the board when either DS0* or DS* is asserted (low). The falling edge of either signal is recognized as an event - "trig". This event is used to latch both the address and the data from the VMEbus, regardless of the cycle type. The "trig" is delayed (50ns) by a digital delay line to allow time for the VMEpld to decode the addressing information. The VMEpld ascertains if the cycle in progress pertains to the controller board, by verifying if the AMCODE is either 39, 3A, 3D or 3E and if the address is 0x019XXXXXH. If the current cycle pertains to the controller board, the VMEpld asserts a board select signal "BSEL". "BSEL" is used to qualify the delayed "trig" signal - "trigxdelay". If "BSEL" is not asserted, "trigxdelay" is ignored by the board and no further

action is taken. If "BSEL" is asserted, the qualified "trigxdelay" signal called "trigxvalid" is generated. "Trigxvalid" enables the data buffers and sets a flip-flop. The output of the flip-flop, "startxsfer", commences data transfer by asserting the appropriate chip select. "Startxsfer" is delayed (50ns) by another digital delayline to generate "endxsfer". The delayline delay is set to allow sufficient time to complete the data access to the DPRAM, the VMEbus control register or the VMEbus status buffer. "Endxsfer" completes the data transfer by (a) latching the data buffers with data (if the cycle was a read), (b) clearing the flip-flop set by trigxvalid and (c) latching a flip-flop that asserts DTACK*. When the master disasserts the DSx* signals in acknowledgement of the DTACK*, the "trig" is disasserted and the DTACK* flip-flop is cleared. The VMEbus interface circuit is then ready to respond to another bus cycle.

6.2 The Priority Interrupt Bus (PIB) Cycle

A PIB cycle can be initiated by the VMEbus interface to enable the DSP to interrupt the VMEbus host and be serviced by it. The necessary sequence of events are described below. (Please refer to board schematic #7 and VMEpld schematic).

In order to interrupt the VMEbus host, a VMEbus slave need only assert one of the seven interrupt request "IRQx" lines. The IRQx line asserted defines the priority level of the interrupt. All the VMEbus slaves share the same set of IRQ lines and several slaves may be configured to interrupt on any given priority level. The VMEbus master thus needs to obtain the identity of the interrupting slave so that it can execute the appropriate service routine. It initiates an interrupt acknowledge cycle to identify the interrupting slave. An interrupt acknowledge cycle resembles a DTB cycle but is distinguished by a low level on the "IACK" line. The lower three bits of the address bus denote the interrupt level being acknowledged. The VMEbus master then issues a falling edge on IACKIN* to commence the interrupt acknowledge cycle. IACKIN* is daisy-chained through the boards on the VMEbus. This provides an additional level of prioritizing amongst slaves interrupting on the same interrupt level. Slave boards closer to the master receive the IACKIN* signal before the other slaves downstream and thus have a higher priority. On receipt of the falling edge on the IACKIN* signal a non interrupting slave merely passes it down the chain to the next board via its IACKOUT* line. An interrupting slave, however, decodes the three bit code on the address lines. If it has a pending interrupt at that level, it places

an interrupt vector address on the data bus and asserts DTACK*. The VMEbus master uses the interrupt vector address provided by the slave to locate the service routine. The interrupt acknowledge cycle is terminated when the AS* and DSx* signals are disasserted.

When the interrupt acknowledge cycle is completed the slave must disassert the IRQx line it asserted. The VMEbus specification allows two schemes for the release of the IRQx line by the slave. The first scheme, called Release on Acknowledge (ROAK), requires that the slave release the IRQx line when its interrupt request is being acknowledged. The second scheme, called Release on Register Access (RORA) requires the VMEbus master to make a register access to the interrupting slave to require the release of the IRQx line.

The implementation scheme to support the PIB cycle is described below. Refer to board schematic #7 and to VMEpld schematic for details.

The DSP asserts IRQx by writing a 1 to location block6:segment5:0:subport4 in bit position <31>. This sets a latch in the VMEpld whose output is used to assert the IRQx line. The priority level of the interrupt is jumper selectable and is currently set to allow interrupt on IRQ1. When the VMEbus host initiates an interrupt acknowledge cycle, it places a three bit code on the address bus and asserts the AS* and DSx*. The "trig" is generated, as in the DTB cycle, to latch the address and data bus. In a PIB cycle, the VMEpld does not assert the BSEL signal and the "trigxvalid" does not start a data transaction. Instead the VMEpld decodes the three bit code to identify the interrupt level being acknowledged. If the code is 001 and if there exists a pending interrupt at that level, VECTEN is asserted when the IACKIN* pin goes low. If there is no pending interrupt by the board or if the three bit code does not match the interrupt level at which the board interrupts (level 1), the falling edge on the IACKIN* pin is merely passed out to IACKOUT*. If the interrupt level is changed by changing the jumper position on the board, the VMEpld must be reprogrammed to respond to the new interrupt level. The VECTEN signal enables a 16 bit vector from the interrupt vector register onto the VMEbus data bus [15:0] and asserts DTACK*. The interrupt vector register that stores the interrupt vector is mapped in the VMEbus address space at location 0x019808XXH on VMEbus data bus [15:0]. Thus the VMEbus host can program the interrupt vector that the controller board provides during an interrupt acknowledge cycle.

The choice of the scheme (ROAK or RORA) for the release of the interrupt is also jumper selectable. When set to ROAK (jumper marked on board), the latch in the VMEpld that asserts the IRQx line is cleared when the address strobe is disasserted to mark the end

of the interrupt acknowledge cycle. When the jumper is set to RORA, a 1 in bit <5> in the VMEbus control register clears the latch in the VMEpld.

6.3 The Dual Port RAM (DPRAM)

The DPRAM bank 0 and bank 1 are mapped in the VMEbus address space at location 0x01900000H–0x01901FFFH and 0x01902000H–0x01903FFFH respectively.

The contention resolution scheme on the VMEbus side is exactly the same as that on the DSP side and is explained in Chapter 6.

A peculiarity of the VME-DPRAM data bus connection is "byte swapping". This is done to bridge the differences in the addressing schemes of the VMEbus and the DSP. The VMEbus addresses bytes within a word in reverse order with respect to the way the DSP addresses them.

For the VMEbus:

databus [31:24] corresponds to byte 0 - LSB
databus [23:16] corresponds to byte 0 - LSB+1
databus [15:08] corresponds to byte 0 - LSB+2
databus [07:00] corresponds to byte 0 - MSB

For the DSP:

databus [31:24] corresponds to byte 0 - MSB
databus [23:16] corresponds to byte 0 - LSB+2
databus [15:08] corresponds to byte 0 - LSB+1
databus [07:00] corresponds to byte 0 - LSB

This has been corrected in hardware by swapping the bytes at the VME-DPRAM data bus connection. However, it appears that the Heurikon board does the same, either in software or in hardware. This requires that the data transferred over the VMEbus be appropriately processed (by swapping the bytes within a word) either before or after the transfer, as is convenient.

6.4 The VMEbus Control Register

The VMEbus control register is a 16 bit register mapped in the VMEbus standard address space at location 0x019800XXH. The control register can be written to (but not read) by a quad byte DTB cycle. However since the VMEbus control register is connected to the VMEbus data bus [31:16] only those bytes are latched when a quad byte transfer is performed. Out of the 16 control register bits, 11 are available for use. The 5 bits that have been used are:

```
bit <1> - RESETDSP - allows the VMEbus host to reset the DSP.
bit <2> - CLRBUSYORN - clears the busy flip-flop of DPRAM bank 0 right side.
bit <3> - CLRBUSY1RN - clears the busy flip-flop of DPRAM bank 1 right side.
bit <4> - FLAGDSP - used to pass a flag to the DSP status buffer.
bit <5> - CLRDSPINTR - used for RORA.
```

6.5 The VMEbus Status Buffer

The VMEbus status buffer is a 16 bit buffer mapped in the VMEbus standard address space at location 0x019804XXH. The status buffer can be read (but not written to) by a quad byte DTB cycle. Since the status buffer is connected to the VMEbus data bus [31:16], the data read on the other two bytes during a quad transfer must be ignored. Out of the 16 status buffer bits, 12 are available for use. The 4 bits that have been used are:

```
bit <1> - RESETDSP - connected to bit <1> of the DSP control register.
bit <2> - BUSYOR - output of the busy flip-flop of DPRAM bank 0 right side.
bit <3> - BUSY1R - output of the busy flip-flop of DPRAM bank 1 right side.
bit <4> - FLAGVME - used to pass a flag from the DSP control register.
```

6.6 Simulation and Testing

A THOR model [2] of the implementation of the VMEbus interface has been created. Both the DTB cycle and the PIB cycle have been simulated and found to meet all the VMEbus timing specifications. The DTB cycle has also been verified to function correctly on the actual controller board.

Chapter 7

On-board Self-Test Circuits

During the initial design stages of the controller board, the issues of testing and debugging the board were carefully considered. The sheer number of interconnections in the design necessitated that an effective tool for board testing be devised and incorporated into the design.

In order to devise a test strategy, it was important to identify modules in the design that were relatively loosely connected to the rest of the design. Functional testing of such a module could then be done independently from the rest of the design. A design overview revealed three such sections: the VMEbus interface mechanism, the processor and the memory, and the robot specific circuits.

The VMEbus interface circuitry is connected to the rest of the design via the DPRAM. Regardless of the operation of the rest of the board, the VMEbus interface can be tested by successfully reading and writing data over the VMEbus. Moreover, once debugged, the VMEbus interface can provide a boot-strap mechanism by which the rest of the board may be debugged. With a functional VMEbus interface, the conditions on the board can be probed (controlled and observed) over the VMEbus by the VMEbus host. Since the VMEbus interface itself can be exercised from the VMEbus host, the task of testing the VMEbus interface is eased.

The processor-memory interface is fairly standardized and the two devices are usually compatible almost by specification. Proper functionality is largely a timing issue that can be resolved by a proper choice of parts. These were therefore considered relatively easy to debug and were in fact relied upon to assist in testing the other sections. In retrospect, this assumption has proved to be correct.

The section specific to the robot was considered the major debugging challenge for two reasons. Firstly, this section cannot be debugged in situ because connecting an unstable controller to a robot can have disastrous effects. The robot could easily destroy itself or its surroundings. Secondly, all the robot signals are not digital. It was on this section, therefore, that the effort of providing test circuits was concentrated.

Extending the need for test circuitry one step further led to the idea of providing self-test circuits. It is evident that a functional VMEbus interface and a functional processor-memory interface together provide a powerful debugging tool. The VMEbus interface can be used to load the DPRAM with DSP programs that exercise the robot specific circuits which, in essence, are I/O devices in the DSP's address space. Now, by providing additional circuitry that enables the DSP to monitor the performance of these I/O devices, a self-test scheme can be established. In this scheme, the DSP provides the stimulus to an I/O device as well as observes its response. By comparing the observed response to the expected response, provided a priori to the DSP, the DSP can verify the operation of the device. Moreover, the DSP can be programmed to periodically run diagnostic routines that exercise the I/O devices, observe their response, verify the correctness of their operation and even suggest corrective measures in the event of a device malfunction. The notion of self-test thus relies heavily on being able to use the capabilities ("intelligence") of the processor on board and needs peripheral circuitry.

The robot specific circuits are: the origin and limit signal interface, the encoder interface, the PWM interface, the current signal interface, the HCTL 1000 and the HCTL interface. Of these, the origin and limit signal interface and the current signal interface are input devices directly accessible by the DSP. The HCTL 1000, however, interacts with the robot via the encoder and PWM interface and needs to be monitored.

In the course of its interaction with the robot, the HCTL receives the encoder signals and issues PWM commands. A PLD has been placed in each of the two HCTL-robot paths to enable the DSP to test the HCTL. The "testenc" PLD is placed between the robot and the encoder interface and the "testpwm" PLD is placed between the HCTL and the PWM interface. The "testenc" PLD allows the DSP to provide stimuli to the HCTL and the "testpwm" PLD enables the DSP to monitor the response of the HCTL.

7.1 The Testenc PLD

The "testenc" PLD allows the DSP to disconnect the encoder channel inputs to the HCTL from the robot and provide its own "encoder channels" to the HCTL. The PLD schematic labeled "Encoder Signal Generator" illustrates the implementation and is referred to in the discussion below. JxENCA and JxENCB are encoder channel outputs from the robot. CHAx and CHBx are the corresponding encoder channel inputs to the HCTLs. ENCA and ENCB are provided by the DSP test control register The multiplexors select between the ENCA (ENCB) input from the DSP and one or more of the JxENCA (JxENCB) from the robot to be provided as encoder channel signals — CHAx (CHBx) to the HCTL. The six multiplexor control signals, sX, are decoded from the three signals A, B and C. A, B and C are connected to ENCJSELA, ENCJSELB and ENCJSELC (bits <6>, <7> and <8>) of the DSP test control register.

The DSP can exercise any or all of the six HCTLs as follows. The three bit code on the ENCJSELx signals determine the HCTL(s) to which ENCA and ENCB are provided as encoder channel signals. For ABC=001–110, the HCTL corresponding to joints 1–6 respectively, is exercised by ENCA and ENCB while the other HCTLs receive their channel signals from the robot. ABC=000 exercises all the HCTLs simultaneously. ABC=111 permits normal operation by connecting the encoder channel signals outputs from the robot to the encoder channel inputs to the HCTLs.

Exercising a particular HCTL can be done by appropriately setting the ENCJSELx bits and then effecting sequential writes to the control register to mimic encoder channel signals on ENCA and ENCB. The HCTL position register can then be read to verify if it was appropriately modified to reflect the additional counts.

7.2 The Testpwm PLD

The "testpwm" PLD allows the DSP to monitor the "Pulse" and "Sign" outputs of each of the HCTL. The PLD schematic labelled "PWM monitor" illustrates how this is done.

Monitoring the PWM output requires that the pulse width of the "Pulse" signal be measured and the value of the "Sign" signal be determined. The pulse width of the "Pulse" signal is measured as follows. Following an "enable" signal from the DSP, the first rising

edge on the "Pulse" signal asserts a "start" signal that starts a counter. The counter is clocked by the same clock as the one provided to the HCTLs. After "start" is asserted, the first falling edge on the "Pulse" signal asserts a "stop" signal that stops the counter. The counter value thus represents the width of the pulse in terms of the number of HCTL clock periods. The counter value may be read by the DSP from the DSP test status buffer. The count enable signal to the counter is also provided as a status buffer bit to enable the DSP to determine when the count is valid. The "sign" bit is also provided as a bit (PWMSIGN) in the DSP test status buffer.

Before the test is started the DSP must assert the "STOPPWM" signal. Asserting this signal ensures that the output of the PulseNsgnxx PLD is set to zero regardless of the "Pulse" and "Sign" outputs of the HCTL. The DSP can then monitor one of the six "Pulse" signals at a time. This is selected by the three PWMJSELx (x=A,B,C) bits in the DSP's test control register. These bits also multiplex the corresponding "Sign" signal of the joint being monitored to the "PWMSIGN" bit in the DSP test status buffer.

7.3 The DSP Test Control Register

The DSP test control register is a 16 bit register mapped at page255:block6:segment5:0:subport2 on the DSP data bus [31:16]. The control register bits are described below:

```
bit 0 - STOPPWM - reduces the PWM output to the robot to zero.
bit 1 - PWMJSELA - joint select signal for PWM test.
bit 2 - PWMJSELB - joint select signal for PWM test.
bit 3 - PWMJSELC - joint select signal for PWM test.
bit 4 - PWMTESTENB - enable signal for PWM test.
bit 5 - ENCJSELA - joint select signal for encoder test.
bit 6 - ENCJSELB - joint select signal for encoder test.
bit 7 - ENCJSELC - joint select signal for encoder test.
bit 8 - ENCA - encoder channel signal for encoder test.
bit 9 - ENCB - encoder channel signal for encoder test.
bit 10 - unused
bit 11 - DSPCABOUT - signal for "dsp_conn2" cable connection check.
bit 12 - PWM1CABOUT - signal for "pwm1_conn" cable connection check.
```

```
bit 13 - PWM2CABOUT - signal for "pwm2_conn" cable connection check.
bit 14 - SCBCABOUT - signal for "scb_conn" cable connection check.
bit 15 - ROBOTCABOUT - signal for "rob_conn" cable connection check.
```

The xxxCABOUT signal are used to check cable connection. (described below).

7.4 The DSP Test Status Buffer

The DSP test control register is a 16 bit register mapped at page255:block6:segment5:0:subport3 on the DSP data bus [31:16]. The status buffer bits are described below:

```
bit 0 - PWMDC[0] - lsb of the PWM test counter value.
bit 1 - PWMDC[1] - lsb+1 of the PWM test counter value.
bit 2 - PWMDC[2] - lsb+2 of the PWM test counter value.
bit 3 - PWMDC[3] - lsb+3 of the PWM test counter value.
bit 4 - PWMDC[4] - lsb+4 of the PWM test counter value.
bit 5 - PWMDC[5] - lsb+5 of the PWM test counter value.
bit 6 - PWMDC[6] - lsb+6 of the PWM test counter value.
bit 7 - PWMDC[7] - msb of the PWM test counter value.
bit 8 - PWMDC[8] - carry output of the PWM test counter.
bit 9 - PWMSIGN - sign signal of PWM test.
bit 10 - PWMTESTBUSYN - busy signal of the PWM test (active low).
bit 11 - DSPCABIN - signal for "dsp_conn2" cable connection check.
bit 12 - PWM1CABIN - signal for "pwm1_conn" cable connection check.
bit 13 - PWM2CABIN - signal for "pwm2_conn" cable connection check.
bit 14 - SCBCABIN - signal for "scb_conn" cable connection check.
bit 15 - ROBOTCABIN - signal for "rob_conn" cable connection check.
```

Checking cable connections: A pair of signals (xxxCABOUT and xxxCABIN) on each connector are used to check for cable connection. The cable connects to the controller board on one side and to the robot on the other. On the robot side of the connection, the xxxCABIN-xxxCABOUT pair is tied together. This forms a loop from the DSP test control register bit, through the connector on the controller end, through the connector on the robot end and back through the connector on the controller end to a bit in the status register. This allows the DSP to write a bit pattern to the xxxCABOUT bit in the test

control register and observe the same bit pattern on the xxxCABIN bit in the status buffer and thus verify the cable connection.

7.5 Simulating the Robot

An interesting advantage of the on-board self-test circuitry is the ability to simulate the robot signals to the HCTL. The DSP can provide, to the HCTL, encoder channel signals as they would be generated by the robot. It can also monitor the output of the HCTL in response to the encoder channel signals provided. A robot-like interface can thus be created for the HCTL. This simulated interface is certainly far more limited in scope than the actual robot interface, both in terms of the number of joints that can be simulated and the variety of conditions simulated.

Chapter 8

Miscellaneous Circuits and Board Specifics

This chapter describes the essential miscellaneous support circuitry of this robot controller board. These circuits include the on-board clock generators and buffers, the reset circuit, and the serial I/O circuit. Also highlighted in this chapter are some board-specific issues that may help the user use the board effectively as well as provide insights that will help debugging in the event of a malfunction. These issues are organized by way of comments referencing the board schematics.

8.1 On-board Clocks

Clock signals are required by the DSP and some of its peripheral devices. The speed of operation of the DSP and that of the peripheral circuits is very different. For this reason, several clock signals are required. The different clock signals needed are generated from two sources: a crystal oscillator and the VMEbus system clock.

A crystal oscillator is used to provide the clock signal to the DSP. The crystal oscillator provides a MOS level clock signal of a very stable frequency (with negligible variation over the temperature and power supply range of interest.)

The peripheral devices require 8MHz, 4MHz, 2MHz and 1MHz clocks. These clocks have been derived from the VMEbus backplane signal SYSCLK. SYSCLK is specified at a nominal frequency of 16MHz with a maximum variation of +/- 1.6%. A buffered version of SYSCLK is used as clock input to a four bit counter. The output of the counter thus provides

the desired clocks with the "lsb"=8MHz and the "msb"=1MHz. The counter output is buffered using an ACT244. The ACT family is TTL input compatible and provides a high drive CMOS output. Jumper connections provided at each clock output, to enable the use of an external clock generator, if required.

8.2 Reset Circuit

The reset circuitry provided performs the functions of power-on reset as well as providing a hardware reset for the DSP. The reset circuit is obtained by connecting an IC555 (timer chip) in a monostable configuration. A high pulse of a fixed duration (R13 x C1 x ln 3) is obtained at the output of the IC555 when a falling edge on the "trigger" input triggers the monostable into its quasi-stable state. [5] The switch connected to the "trigger" input provides the necessary trigger by discharging the capacitor connected to it. The output of the IC555 is used to assert the RESTN signal of the DSP. The RESTN signal is also asserted when either the system reset signal SYSRESET on the VMEbus is asserted, or when the VMEbus host writes a 1 to bit <1> of the VMEbus control register on the board. As required by the DSP32C specifications the ZN signals is disasserted after the RESTN signal is disasserted.

8.3 The Serial Input/Output Ports

The Dual Universal Asynchronous Receiver Transmitter (DUART) provides two independent, full-duplex, asynchronous, serial communication channels to the controller board. The DUART can be put to several different uses. It can perform as a simple debugging tool. By connecting a terminal to one serial port of the DUART, access to the different sections of the board may be obtained by directing the DSP appropriately. Moreover, since the two serial channel can be independently configured, the second channel can be used to connect the board to a computer. The board can then be programmed to respond only to a specific pattern of control characters from either the user (terminal) or the computer. This allows the user to interact only with the computer while the computer handles the details of directing the controller board.

Once the controller board is operational the DUART can be used to control an end effector or a force-torque sensing wrist. The bandwidth required for data transfer to

and from these devices is low and the serial I/O channel rate of a few kilobauds is sufficient.

The DUART used on the board is the 24 pin version of the Signetics SCN2681. It is mapped at location page255:block6:segment5:0:subport7 on the DSP data bus [31:24]. Its internal registers may be accessed by specifying the remaining 4 bits of the DSP's address. The SCN2681 provides a convenient microprocessor interface, however, it does not tri-state the data bus until 100ns after the microprocessor disasserts the read signal. A buffer controlled by the chip select of the DUART is used to tri-state the data bus sooner. Also, the read/write pulse required by the SCN2681 is 225ns. Thus in order to interface the DSP to the DUART, the DSP clock must be reduced to about 7MHz. However, interface to the DSP32C is facilitated since the DSP32C supports the introduction of wait states in its memory access. The circuitry required to introduce the wait states, by disasserting the SRDYN signal, is implemented in the unused section of the "testenc" PLD. The circuit disasserts the SRDYN signal for 5 clock periods of the DSP's clock. This allows a DSP32C operating at 45MHz, or less, to be interfaced to the DUART. The DUART also generates an open drain interrupt signal. This signal is used to interrupt the DSP and is also provided as a status buffer bit for the DSP to poll.

The DUART provides TTL level outputs and accepts TTL level inputs. The translation of voltages from the TTL levels at the output of the DUART to the RS232 levels, and from the RS232 levels to the TTL levels at the input of the DUART is achieved by the "driver" (board sch #18). This is Maxim's MAX233. It is a convenient single chip solution that operates from a 5V supply and does not require any external boot-strapping capacitors to produce the RS232 levels.

8.4 Board Issues

This section serves to provide additional comments to the board schematics included in Appendix B. These comments are ordered as the schematic sheets. Also included here are some board level issues and hints that (it is hoped) will help the user use and maintain the board efficiently.

Reference to locations on the actual board is made with respect to the following orientation: The component side facing the reader and the VMEbus edge connectors to the right of the user. A plan of the board is provided in the Appendix A to help locate the different sections of the board.

- Schematic Sheet #1: The address buffers each bear a capacitive load of about 8 pins. With an estimated 6pF per pin this gives the rated 50pF load on the buffers. The read/write signals (MGN/MWN) are connected to the memory devices and the buffered read/write signals (MGNIO/MWNIO) are connected to the peripheral devices.

- Schematic Sheet #2: All the DSP signals are available for future use and for probing on these two connectors.

- Schematic Sheet #3: When changing the on-board processor from DSP32 to DSP32C the PALs must also be changed too.

- Schematic Sheet #4: Banks of EPROM may be effectively removed by grounding CS3. This is useful when changing the processor from DSP32C to DSP32.

- Schematic Sheet #5: The DPRAMs must all be of the "master" type if byte access from the DSP port is required, else only one can be a "master" and the other three "slaves". Beware — bytes are being swapped within a word.

- Schematic Sheet #6: Same as sheet #5.

- Schematic Sheet #7: The RORA/ROAK jumpers and the IRQx jumpers are in the top right corner of the board. The VMEpld must be reprogrammed whenever the IRQx jumper position is changed.

- Schematic Sheet #8: IRQx lines from the jumper to the VMEbus edge connectors could not be routed by the board router. These connections have been made by manually connected wires.

- Schematic Sheet #9: On power-up, the interrupt vector must be written by the host before the controller board issues an interrupt request. The upper byte of the status buffer may be used to monitor specific robot signals.

- Schematic Sheet #10: "pwmx_conn" connects to the servo amplifier, "scb_conn" to the signal conditioning board, and "robot_conn" to the origin and limit processing boards.

- Schematic Sheet #11: Gated clocks used. Use a fast 3–8 decoder.

- Schematic Sheet #12: Comment under LOCKX signal should read: "from IERROR PLD".
- Schematic Sheet #13: Same as sheet #12.
- Schematic Sheet #14: The td62501p common emitter transistor array chips are placed above the analog section.
- Schematic Sheet #15: The analog section is shielded from the rest of the board by a grid of holes around it. These holes can also be used for prototyping purposes.
- Schematic Sheet #16: The voltage regulator (bottom center of the board) has a heat sink built onto the board.
- Schematic Sheet #17: These PLDs reside on the top left corner of the board.
- Schematic Sheet #18: Same as sheet #17.
- Schematic Sheet #19: An error in the connections to the ACT has been corrected by a post-PCB alteration. (The inputs and outputs appear swapped on the schematic.)

Chapter 9

Future Enhancements

This chapter outlines a few suggestions that could be incorporated in the future versions of the robot controller. These suggestions are an embodiment of the learning curve traced in the process of designing this controller. They are directed towards making the second generation of this controller considerably more efficient, powerful, versatile, robust and user-friendly.

9.1 Suggested Hardware Improvements

- **Increasing the On-Board Processing Power**

As is customary with computing resources, a dearth of the on-board processing power will inevitably exist in the future. To some extent, this crisis has been mitigated by designing the board to be upgradable to a more powerful processor, the DSP32C, and by interfacing the board to a standard bus, the VMEbus. Among other viable alternatives, is the use of multiple DSPs on one board. The DSPs may be cascaded serially using their on-chip serial ports (as in [1]) or a host processor may be provided to orchestrate the DSPs using each DSP's parallel I/O port. In either case, theoretically, a significant improvement in the processing capability of the board may be realized.

- **Reducing the Number of TTL Circuits**

A considerable amount of board space (about 1/3) and power is consumed by the TTL circuits on-board. An effort to limit their use has been made by employing, to the

fullest extent, functional PLD equivalents. However, the small number of I/O pins on PLDs often becomes a constraint to their use. As the I/O pins required by a functional block increase, the PLD implementation quickly degenerates to a TTL implementation as far as board area is concerned. Also, in some cases the TTL circuitry is necessary by specification (e.g. the VMEbus interface chips) and in other cases by the speed requirement (e.g. of the HCTL registers, the control register and status buffers). Some of the suggestions that follow indicate how the number of TTL circuits can be reduced.

- **Replacing the HCTLs**

The HCTL is a versatile motion controller. However, since its design in 1985, rapid progress in IC technology has rendered its microprocessor interface obsolete. A significant overhead in terms of board area and programming inconvenience is borne to interface the HCTL to the DSP. A custom I/O device could be used in conjunction with additional processors to replace the HCTL. Using the same amount of board area, a significant improvement in the capabilities of the controller could be realized.

- **Providing an Input/Output Controller**

As currently designed, sensor interface to the controller is expected to be via the VMEbus. However, as the number of signal processing boards in the VMEbus card cage increases, bus bandwidth limitations may become apparent. It would therefore be convenient to be able to interface sensors directly to the controller via parallel ports. This can be made possible by incorporating smart peripheral controllers on the board.

- **Replacing the Analog Filters**

As mentioned in Chap 4, the analog filters have been implemented to maintain compatibility with the servo amplifiers. Once the interface issues are better resolved, the analog section could be replaced with alternate implementations such as switched capacitor filters.

- **Providing Prototyping Area**

The "3-high, 2-deep" VMEbus board described here is filled with about 150 ICs. As a result, there is little room for prototyping. Prototyping area could be useful if custom chips (e.g. inverse kinematics chip using CORDICs [2]) are to be used as on-board hardware accelerators.

9.2 Suggestions for Software and Firmware

This project deals primarily with the hardware aspect of the controller design. Though ample support functions for software have been provided, a significant amount of software and firmware needs to be designed to exploit fully the capabilities of the board. Some suggestions, towards that end, are presented below.

- **Boot Sequence**

The initialization routine must include:

a) Resetting peripherals - the HCTLs, all PLDs, the DAP.
b) Initializing the registers in the HCTLs, in the DAP and in the DUART.
c) Servoing upon brake release.
d) Origining on cold starts.

The periodic routines must include:

a) Encoder count loss correction.
b) Cable connection check.
c) HCTL performance check.
d) Self test checks.

Chapter 10

Conclusions

The design of a powerful hybrid force-position controller for a six axis robot arm has been described. The board implementation of this controller is complete and it is currently being integrated into the robot system. Preliminary tests of the controller's capabilities have yielded very promising results.

In the course of building a complete robot system, of which this controller forms a part, a number of architectural and implementation issues have surfaced and a lot has been learned in the process of resolving these issues.

The design of this controller commenced with the definition of the input-output requirements and a preliminary outline of the architecture. Further architectural refinements were made to accommodate the multitude of envisaged tasks of the controller. This *"top-down" design approach* has been extremely helpful in zeroing in on a task specific efficient architecture. The architectural decisions such as the choice of the system bus, the use of dual-port RAMs for interboard communication, the use of a powerful on-board DSP have all been crucial to the success of the resultant implementation.

Partitioning the design at the top level has helped in two other respects. Firstly, the design optimization has been done at a higher architectural level. Coupled with a fair understanding of the underlying implementation technology, this has lead to the efficient use of resources. Secondly, it has helped focus the design effort to more manageable sub-blocks. This has eased the task of implementing the architecture immensely.

Use of the existing system blocks has been an important design decision. The key decisions in this regard have been the use of the original servo amplifiers and the use of a commercially available VMEbus host for user interface. From the point of view of

interfacing and system development, these choices have been very critical.

During the phase of the architectural definition, a key principle has been that of *accommodating for future expansion.* The robot system and the controller board have both been designed with the future needs in mind. In microprocessor based system applications, an area of rapid obsolescence, this decision has helped prolong the useful life span of the system.

Following the architectural definition, rapid prototyping was made possible by the *extensive use of CAD tools.* The Hewlett Packard's Design Capture System (DCS) and the Printed Circuit Design System (PCDS) have been used for the schematic capture of the design and the layout of the printed circuit board, respectively. The THOR environment has been used to model and simulate the behavior of the asynchronous VMEbus interface circuit. SPICE has been used to verify the characteristics of the analog filters. The A+PLUS PLCAD Supreme software package by ALTERA, has been used for the schematic capture, design and functional verification of the PLDs. All these CAD tools have been crucial to the rapid prototyping of the design.

In the area of implementation, the concept of *design flexibility* has been used effectively. Since it is very difficult to account for all the influencing factors a priori, the ability to modify an existing hardware design is extremely important. The lavish use of reconfigurable logic circuits is an instance of the application of this concept.

The concept of *incorporating test circuits* within the design has proved to be very useful in easing the task of debugging the hardware. By carefully considering the modules of the design, it is possible to devise a test strategy by investing a relatively small fraction of design effort and hardware. The extension of the use of test circuits for self-test is both useful and novel.

While tireless efforts continue in the search of a perfect machine the achievements of "nature" must be applauded with due respect.

In the words of Elbert Hubbard:

> "One machine can perform the tasks of a hundred ordinary men, but no machine can perform the tasks of one extraordinary man."

Bibliography

[1] J. S. Albus, A. J. Barbera, and M. L. Fitzgerald. Sensor robotics in the national bureau of standards.

[2] H. H. Alrutz, S. R. Quackenbush, and J. H. Snyder. *An Enhanced DSP32 Signal Processing Module for the VMEbus.* Technical Report, A. T. & T., October 1987.

[3] John J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, 1986.

[4] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence.* McGraw-Hill, 1987.

[5] David A. Hodges and Horace G. Jackson. *Analysis and Design of Digital Intergrated Circuits.* McGraw-Hill, 1983.

[6] Motorola Inc. *The VMEbus Specification.* revision c.1 edition.

[7] VLSI Technology Incorporated. *Application Specific Memory Products Data Book 1988.* San Jose, CA., 1988.

[8] Texas Instruments. *ALS/AS Logic Circuits Data Book.* 1983. Advanced Low-Power Schottky/Advanced Schottky.

[9] Texas Instruments. *Supplement to TTL Data Book.* volume 3 edition, 1984. Advanced Low-Power Schottky/Advanced Schottky.

[10] Texas Instruments. *The TTL Data Book.* volume 1, 2 and 3 edition, 1984.

[11] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control.* The MIT Press, 1981.

[12] et. al. Robert Alverson. *THOR User's Manual: Tutorial and Commands.* Technical Report, Stanford University, January 1988.

[13] Cypress Semiconductor. *CMOS Data Book.* San Jose, CA, January 1988.

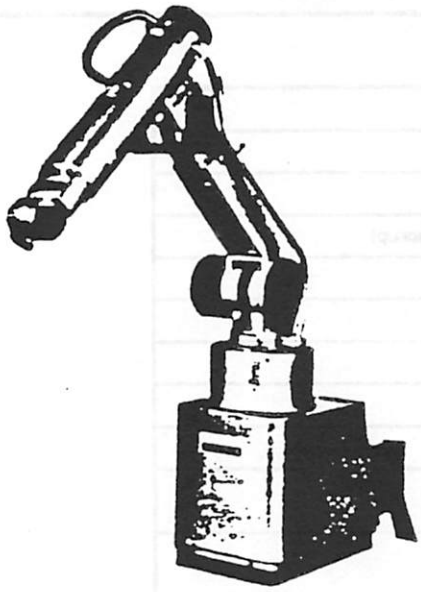
Appendix A

Block Diagrams

Pana Robo V_I Series

Vertical Articulated System

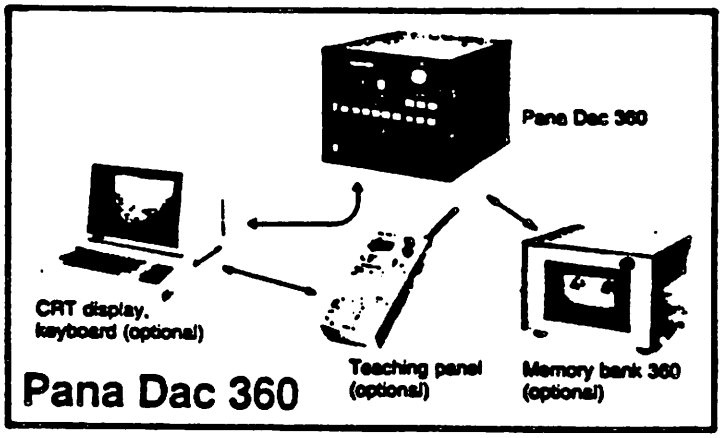
(6 axes)



Specifications

Robot unit	Model No.		NM-6740 (Large)	
	Operating range		Operating range	Max. speed
		Joint 1 α1	270°	90°/sec
		Joint 2 φ1	180°	54°/sec
		Joint 3 φ2	170°	76°/sec
		Joint 4 α2	300°	140°/sec
		Joint 5 φ3	240°	120°/sec
		Joint 6 β	360°	180°/sec
Control unit	Composite speed		max 2000mm/sec (78.8"/sec)	
	Repeatability		±0.1mm (±0.004")	
	Load capacity		5kg (11lb)	
	Weight		110kg (242lb)	
Operating range	Applicable control unit		Pana Dac 360	
	Control system		CP/PTP control	
	Programming system		Teaching	
	Number of control axes		6 axes simultaneously	

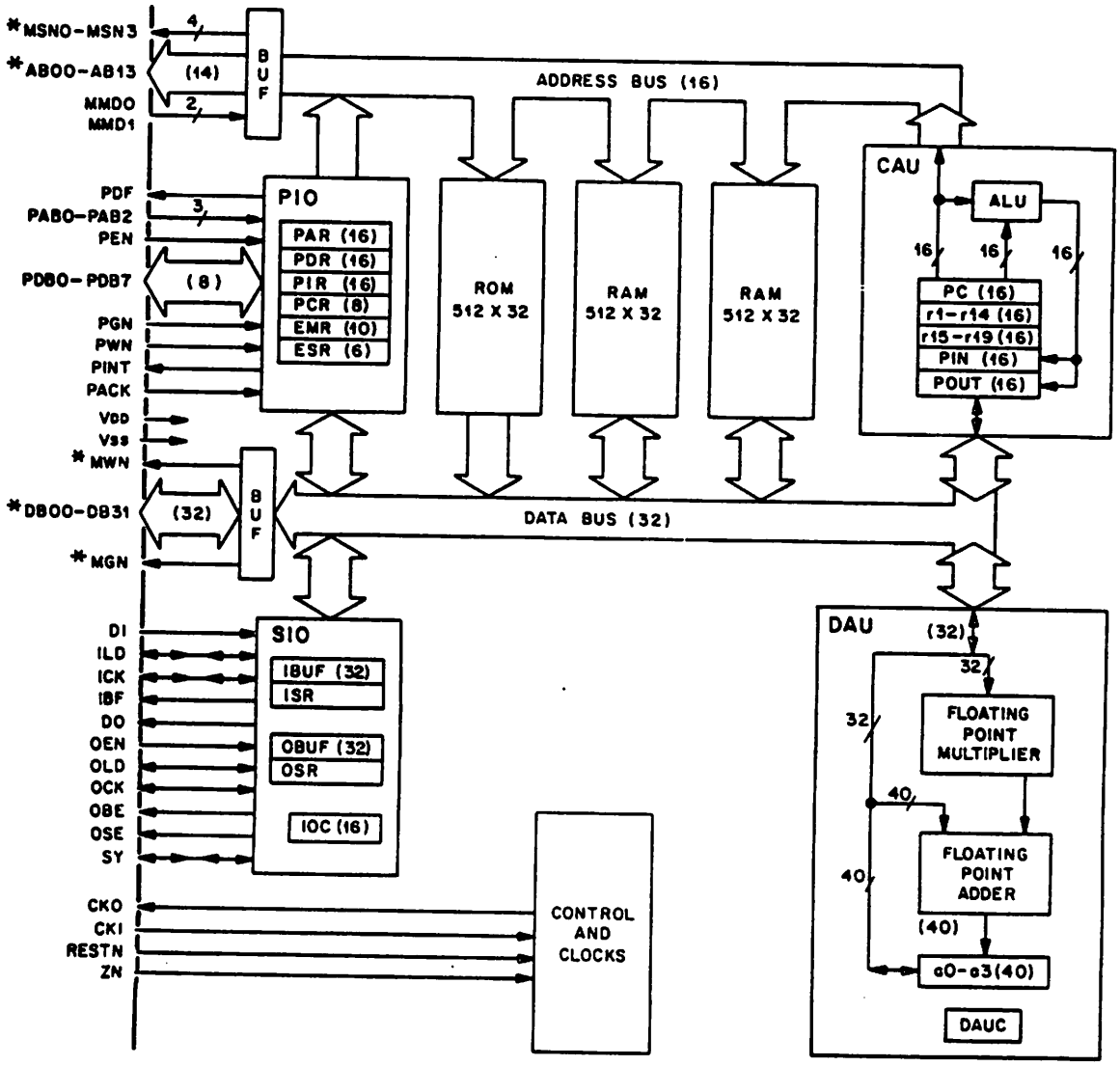
Note: Under CP control, max. speed and repeatability may diminish due to movement pattern.



Model		Pana Dac 360
Control system		PTP CP
Position detection system		Rotary encoder
Memory system		IC memory (battery backup)
CPU type		16-bit CPU (MN1613)
Program steps		1000
Program divisions		10
Number of control axes		4, 6
Simultaneously controlled axes		4, 6
Speed control	Speed setting	Possible (as desired)
	Acceleration/ deceleration setting	
Programming		1) Playback using teaching panel (Teaching requires teaching panel, CRT display and keyboard)
Least input increment		— *
Max. effective setting amount		— *

External input/output signal	I : 9 + 7 (Available for user) O : 8 + 8 (Available for user)
External communication	RS-232C (Optional)
Timer function	0.1 ~ 32.0sec
Program alteration function	Available
Alarm function	Overrun, overload, emergency, input error
Dimensions	405 (H) x 430 (W) x 475 (D)mm [16.0 (H) + 16.9 (W) + 18.7(D)"]
Weight	65kg (143lb)
Power supply	100V AC ±10%, 50/60Hz
Environmental conditions	Temperature: 0 to 40 degrees C, R.H.: 35 to 85% (Non condensing)
Standard specifications	1) Liner/circular arc interpolation function 2) Jump function 3) Conditional jump function 4) Origin offset function 5) Coordinate conversion function 6) Subroutine function 7) Self diagnosis function
Applications	Horizontal/vertical articulated system
Options	1) CRT display, keyboard 2) Memory bank 360 3) Teaching panel 4) External communication (RS-232C) 5) Palletizing function

* It depends upon combination with robots.

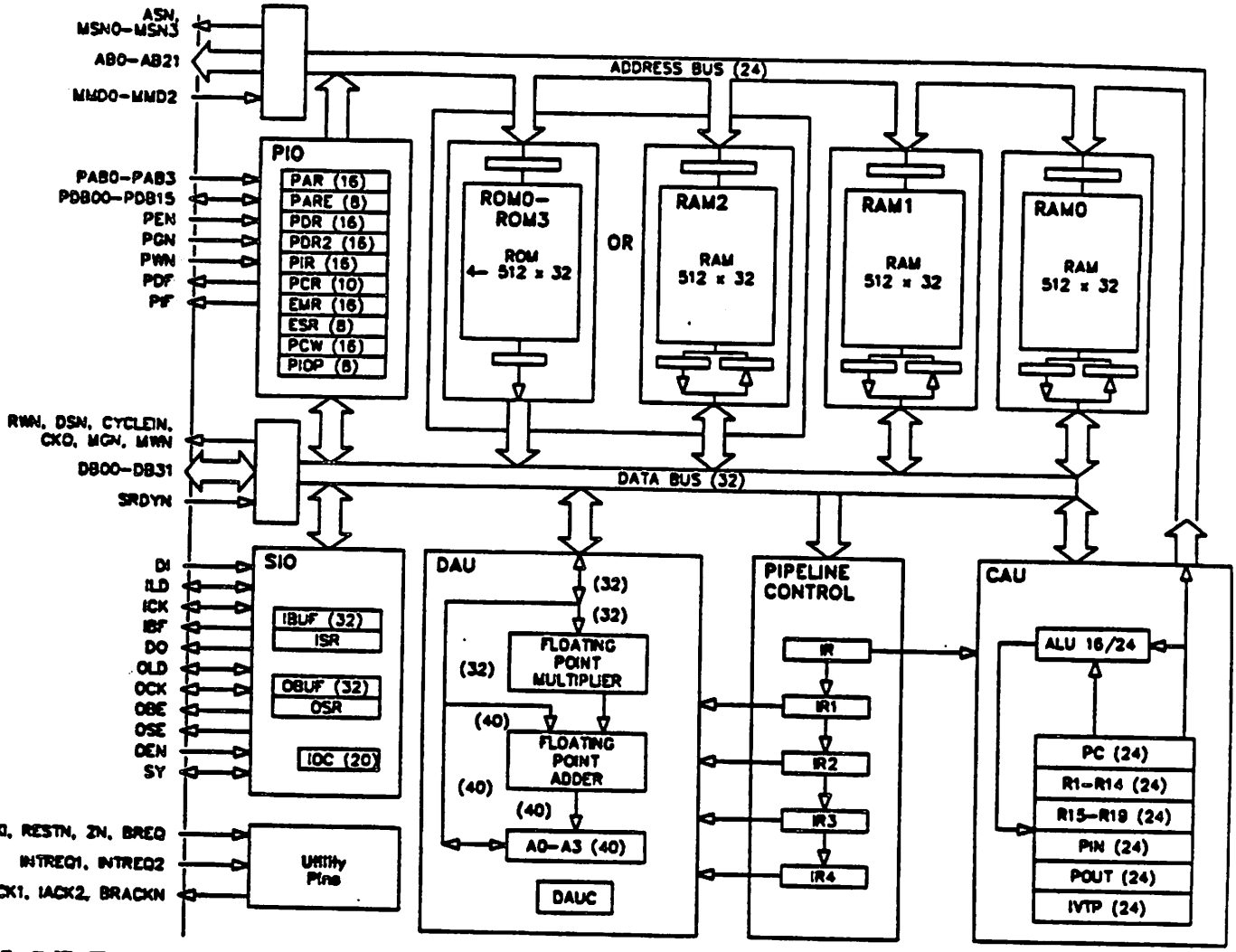


Note: See Tables 21—27 for signal name definitions.
* Not available on the 40-pin DIP.

Legend:

a0—a3	Accumulators 0—3	FPM	Floating-Point Multiplier	PCR	PIO Control Register
ALU	Arithmetic Logic Unit	IBUF	Input Buffer	PDR	PIO Data Register
CAU	Control Arithmetic Unit	IOC	Input/Output Control Register	PIO	Parallel I/O Unit
DAU	Data Arithmetic Unit			PIR	PIO Interrupt Register
DAUC	Data Arithmetic Unit Control Register	ISR	Input Shift Register	r1—r19	Registers 1—19
		OBUF	Output Buffer	PIN	Parallel Input Register
EMR	Error Mask Register	OSR	Output Shift Register	POUT	Parallel Output Register
ESR	Error Source Register	PAR	PIO Address Register	RAM	Read/Write Memory
FPA	Floating-Point Adder	PC	Program Counter	ROM	Read-Only Memory
				SIO	Serial I/O Unit

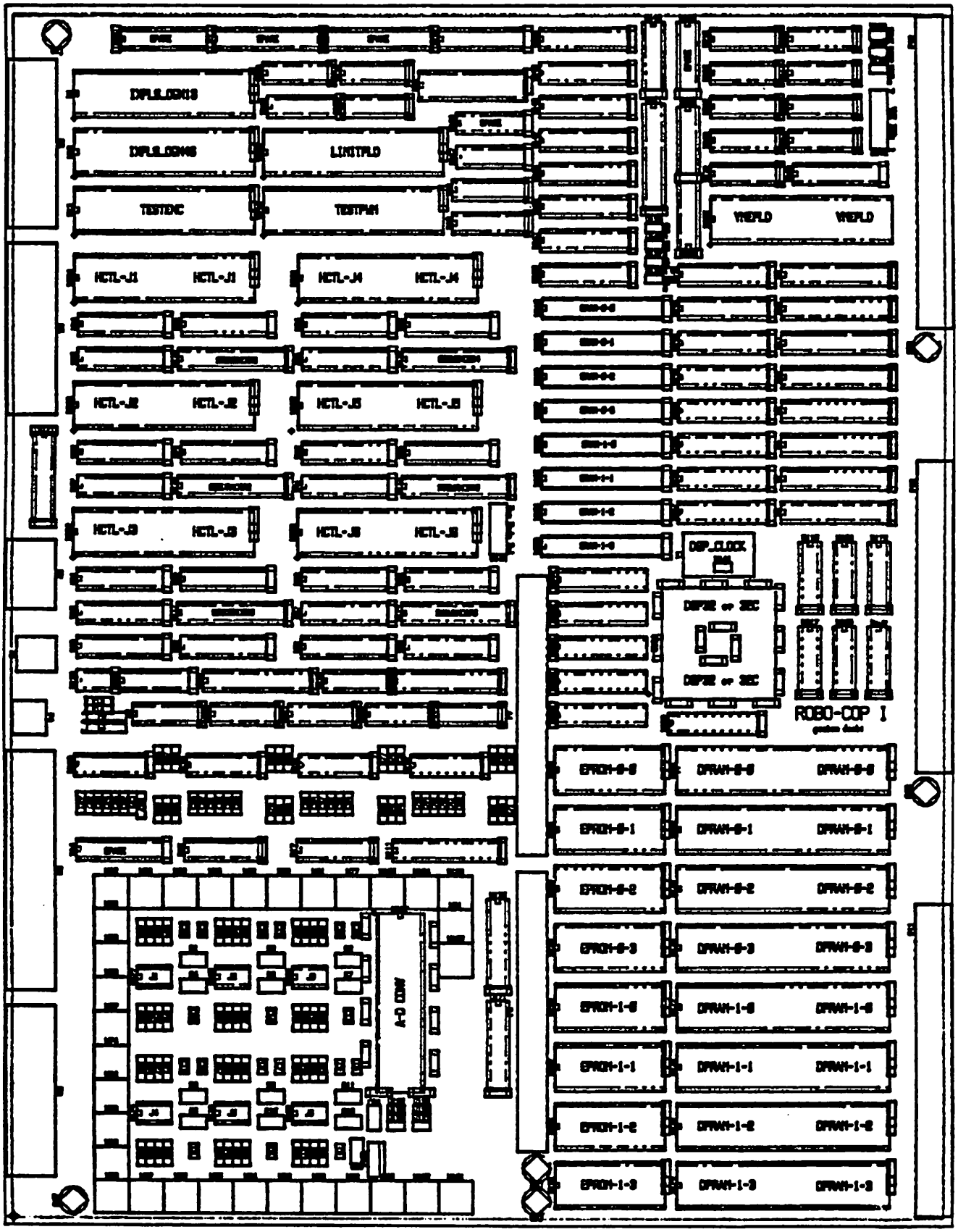
Block Diagram



Block Diagram of the DSP32C

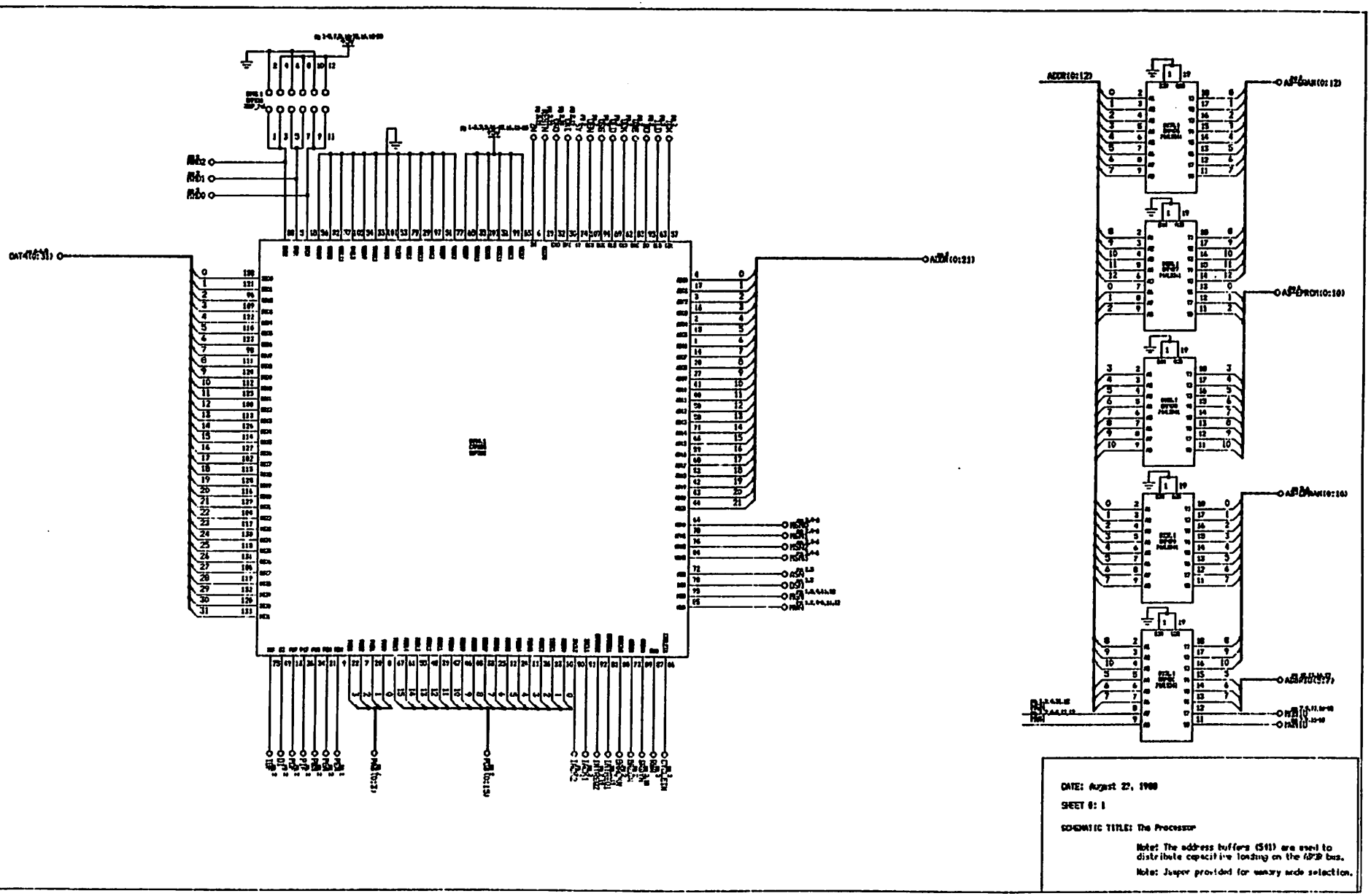
LEGEND:

A0—A3	Accumulators 0—3	ISR	Input Shift Register	PDR2	PIO Data Register 2
ALU	Arithmetic Logic Unit	IVTP	Interrupt Vector Table Pointer	PIN	Serial DMA Input Pointer
CAU	Control Arithmetic Unit	OBUF	Output Buffer	PIO	Parallel I/O Unit
DAU	Data Arithmetic Unit	OSR	Output Shift Register	PIOP	Parallel I/O Ports
DAUC	DAU Control Register	PAR	PIO Address Register	PIR	PIO Interrupt Register
EMR	Error Mask Register	PARE	PIO Address Register Extended	POUT	Serial DMA Output Pointer
ESR	Error Source Register	PC	Program Counter	R1—R19	Registers 1—19
IBUF	Input Buffer	PCR	PIO Control Register	RAM	Read/Write Memory
IOC	Input/Output Control Register	PCW	Processor Control Word	ROM	Read-Only Memory
IR	Instruction Register	PDR	PIO Data Register	SIO	Serial I/O Unit


DSP_CLOCK
ROBO-COP I

Appendix B

Board Schematics



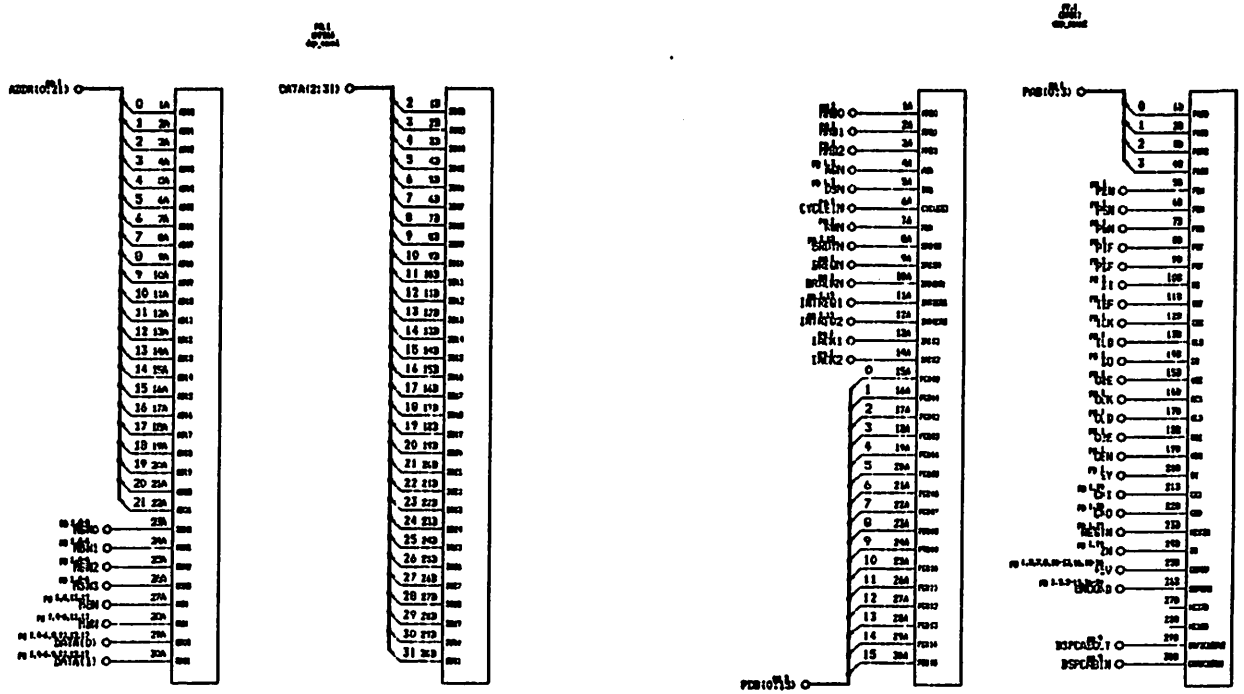
DATE: August 27, 1988

SHEET 8: 1

SCHEMATIC TITLE: The Processor

Note: The address buffers (S41) are used to distribute capacitive loading on the ADSB bus.

Note: Jumper provided for memory mode selection.



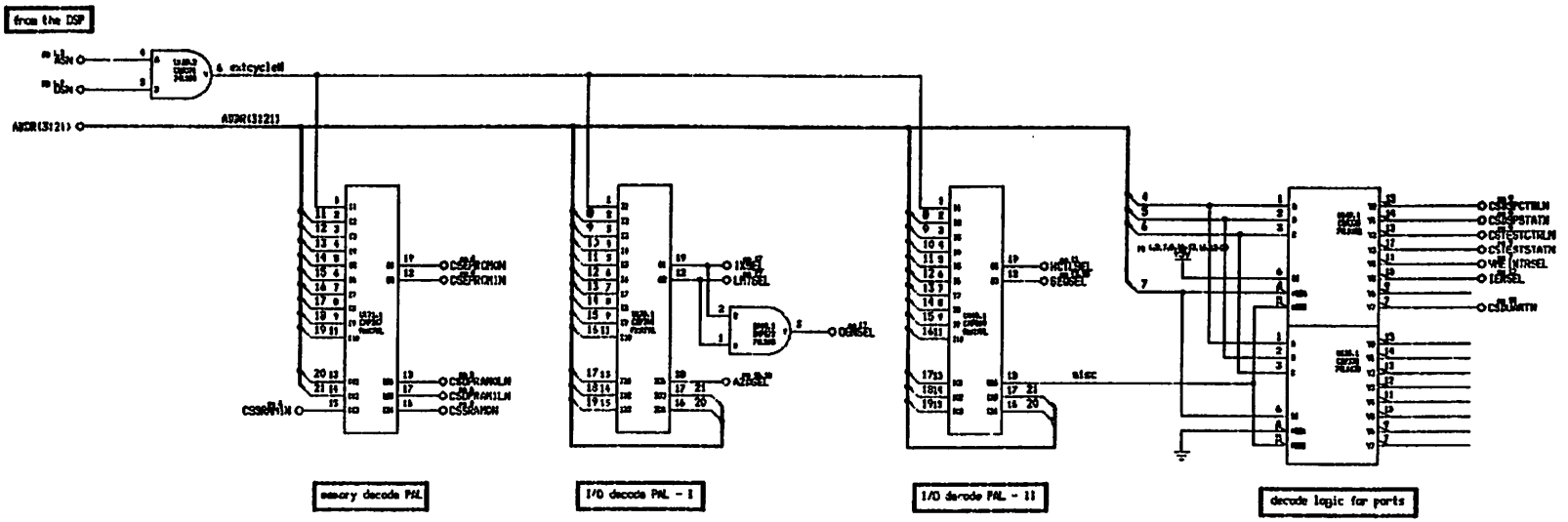
DATE: August 22, 1988

SHEET: 2

SCHEMATIC TITLE: Connectors to provide access to DSP32 pins (made available for future use).

Note: All signal pins from the DSP are available unbuffered. Buffering must be considered for heavily loaded pins.

Note: DSPCABIN & DSPCABOUT used to check cable connection. (short these on receiving end).



DATE: August 2, 1988

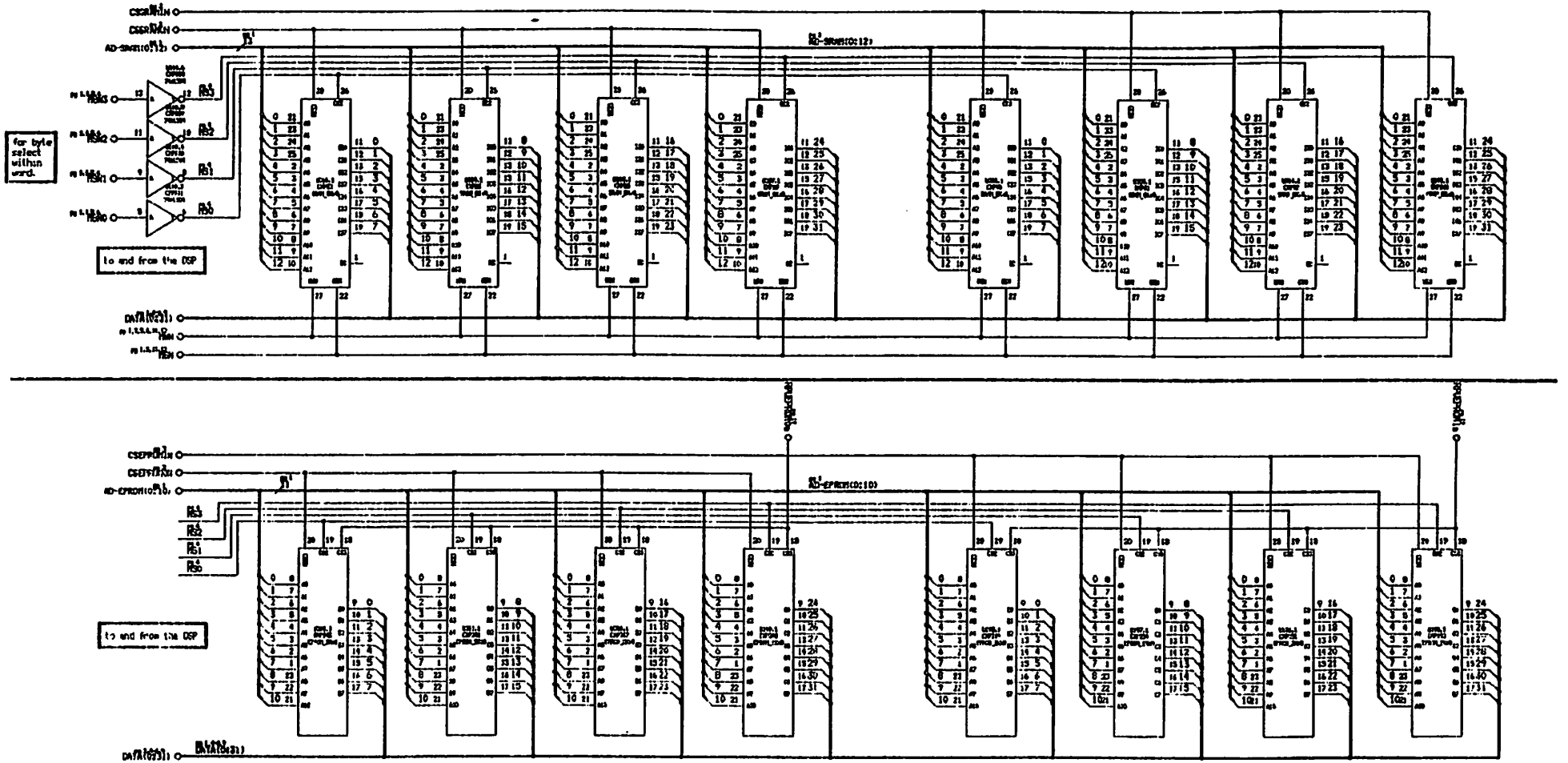
SHEET 8: 9

SCHEMATIC TITLE: DSP Address decoders.

Note: ASN or BSN low denotes an external memory transaction. This is used to save power diss during inactive periods by deselecting memory.

Note: UDNSEL is ored with IISEL and LMTSEL since DUN signal is read only and placed on the upper bytes of the data bus.

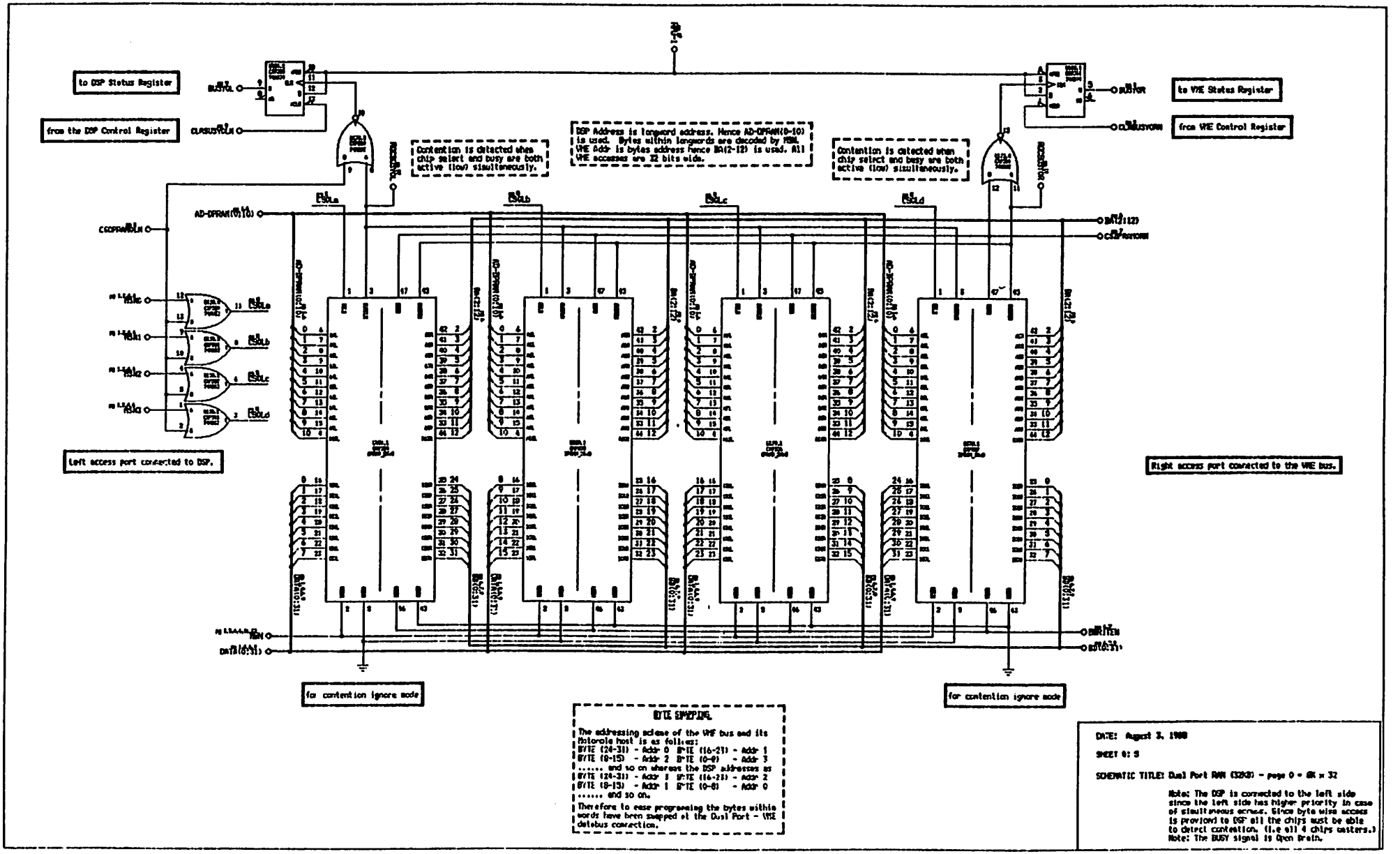
Note: All PAL outputs are active low.

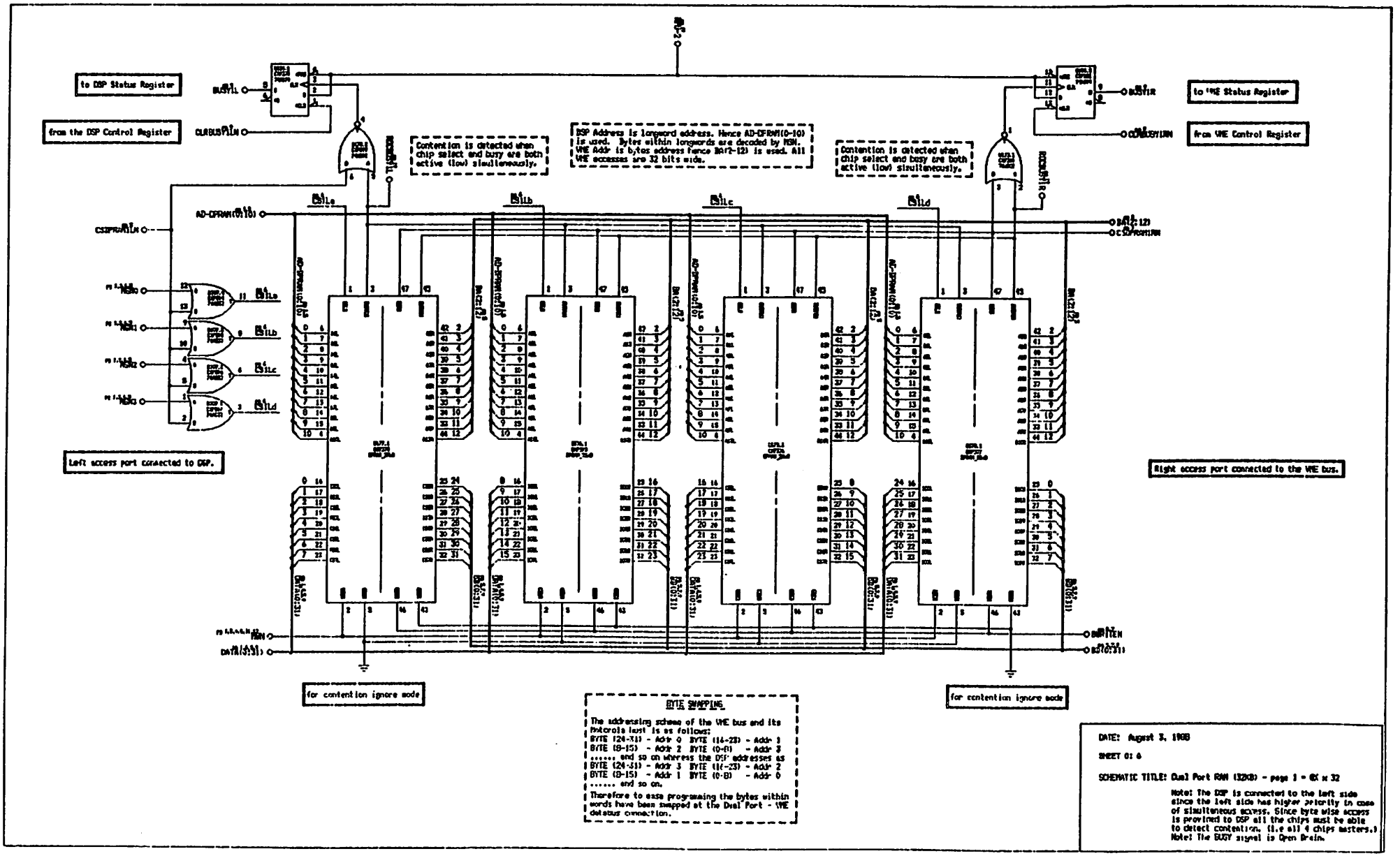


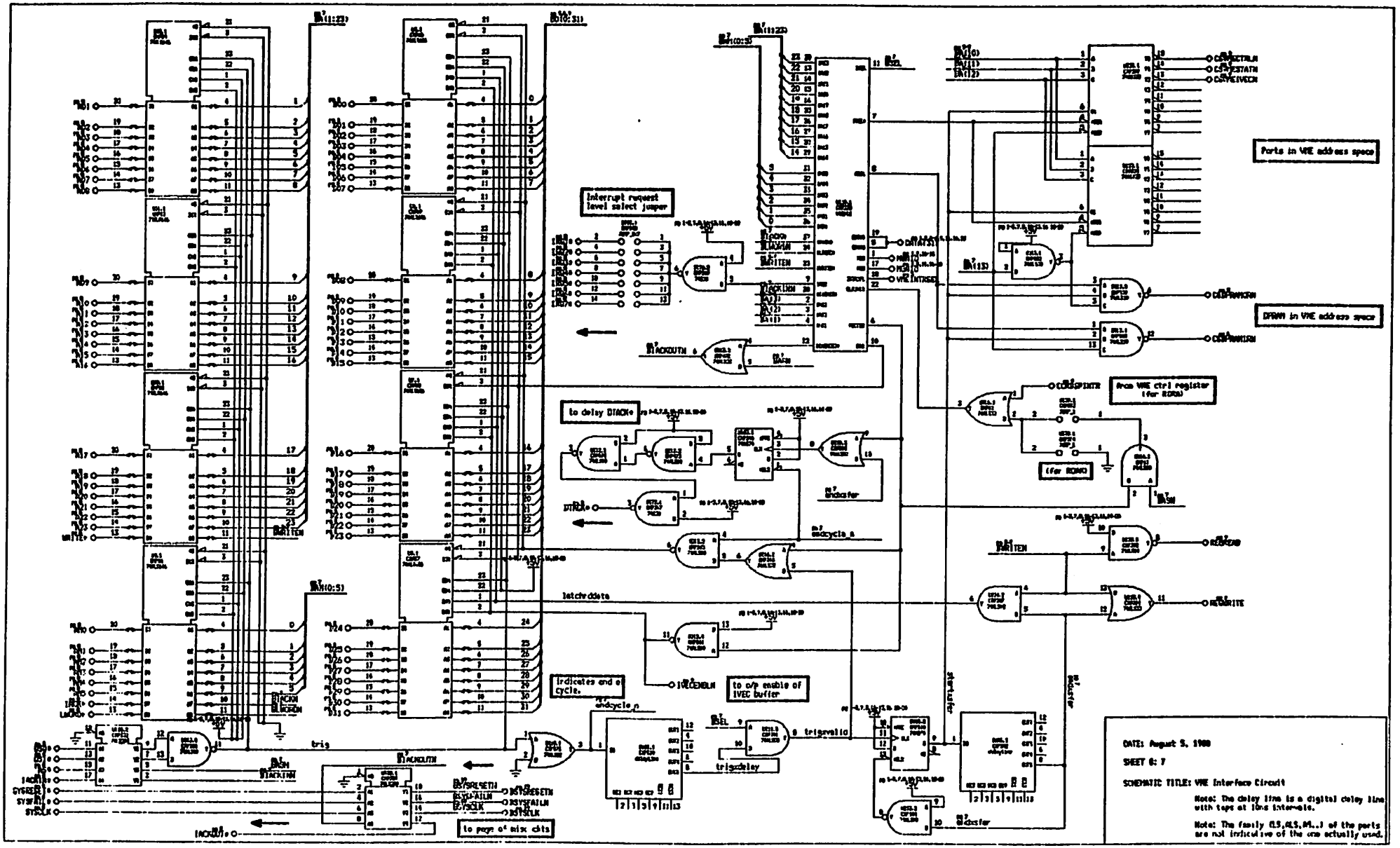
DATE: August 4, 1988

SHEET 8: 4

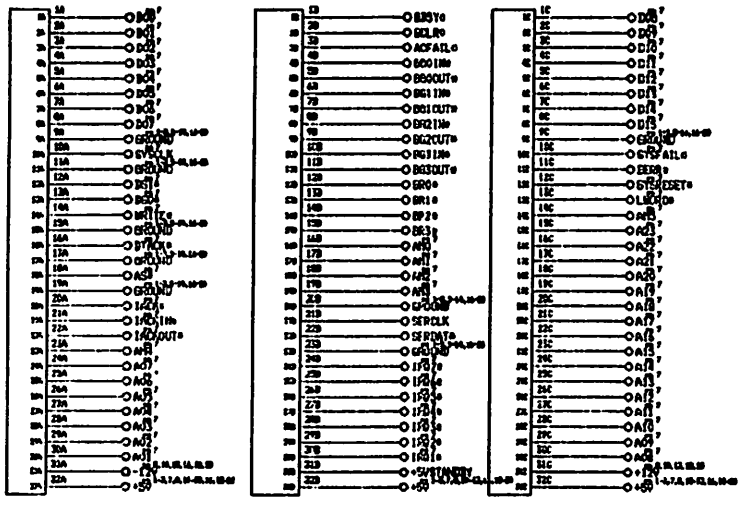
SCHEMATIC TITLE: DSP Memory.
Static RAM = 64KB organised as:
SRAM0 = 8K x 32.
SRAM1 = 8K x 32.
Erasable PROM = 1.0MB organised as:
EPROM0 = 2K x 32.
EPROM1 = 2K x 32.


to DSP Status Register
from the DSP Control Register
to VME Status Register
from VME Control Register
Contention is detected when chip select and busy are both active (low) simultaneously.
DSP Address is longword address. Hence AD-DPRAM(0-10) is used. Bytes within longwords are decoded by MSK. VME Addr is bytes address hence BA(2-12) is used. All VME accesses are 32 bits wide.
Contention is detected when chip select and busy are both active (low) simultaneously.
Left access port connected to DSP.
Right access port connected to the VME bus.
for contention ignore mode
for contention ignore mode
BYTE SWAPPING
The addressing scheme of the VME bus and its Motorola host is as follows:
BYTE (24-31) - Addr 0 BYTE (16-23) - Addr 1
BYTE (8-15) - Addr 2 BYTE (0-8) - Addr 3
..... and so on whereas the DSP addresses as
BYTE (24-31) - Addr 1 BYTE (16-23) - Addr 2
BYTE (8-15) - Addr 1 BYTE (0-8) - Addr 0
..... and so on.
Therefore to ease programming the bytes within words have been swapped at the Dual Port - VME databus connection.
DATE: August 3, 1988
SHEET 8: 8
SCHEMATIC TITLE: Dual Port RAM (32K8) - page 0 = 8K x 32
Note: The DSP is connected to the left side since the left side has higher priority in case of simultaneous access. Since byte wise access is provided to DSP all the chips must be able to detect contention. (i.e all 4 chips casters.)
Note: The BUSY signal is Open Drain.

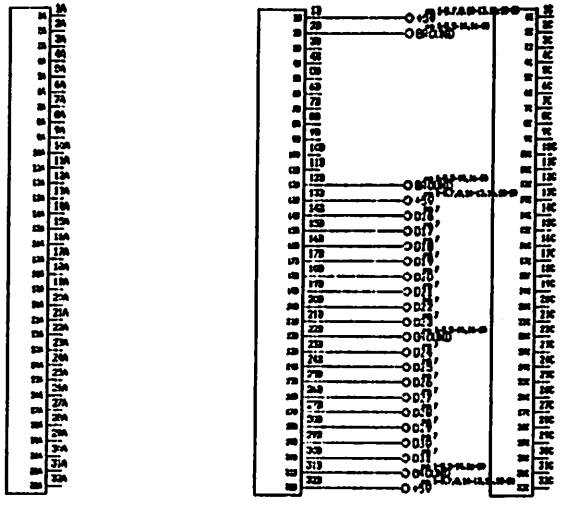

to DSP Status Register
from the DSP Control Register
to VME Status Register
from VME Control Register
Contention is detected when chip select and busy are both active (low) simultaneously.
DSP Address is longword address. Hence AD-DPRAM(0-10) is used. Bytes within longwords are decoded by MSN. VME Addr is bytes address hence BA<2-12> is used. All VME accesses are 32 bits wide.
Contention is detected when chip select and busy are both active (low) simultaneously.
Left access port connected to DSP.
Right access port connected to the VME bus.
for contention ignore mode
for contention ignore mode
BYTE SWAPPING
The addressing scheme of the VME bus and its Motorola test is as follows:
BYTE (24-31) - Addr 0 BYTE (16-23) - Addr 1
BYTE (8-15) - Addr 2 BYTE (0-8) - Addr 3
..... and so on whereas the DSP addresses as
BYTE (24-31) - Addr 3 BYTE (16-23) - Addr 2
BYTE (8-15) - Addr 1 BYTE (0-8) - Addr 0
..... and so on.
Therefore to ease programming the bytes within words have been swapped at the Dual Port - VME datatus connection.
DATE: August 3, 1988
SHEET 01 0
SCHEMATIC TITLE: Dual Port RAM (32K8) - page 1 - 8K x 32
Note: The DSP is connected to the left side since the left side has higher priority in case of simultaneous access. Since byte wise access is provided to DSP all the chips must be able to detect contention. (i.e all 4 chips masters.)
Note: The BUSY signal is Open Drain.


Interrupt request level select jumper
Ports in VME address space
DRAM in VME address space
from VME ctrl register (for ROMs)
to delay DTACK
(for ROMs)
indicates end of cycle.
to o/p enable of IVEC buffer
to page of misc cits
DATE: August 5, 1988
SHEET 6: 7
SCHEMATIC TITLE: VME Interface Circuit
Note: The delay line is a digital delay line with taps at 10ns intervals.
Note: The family (LS,ALS,AS..) of the parts are not indicative of the one actually used.

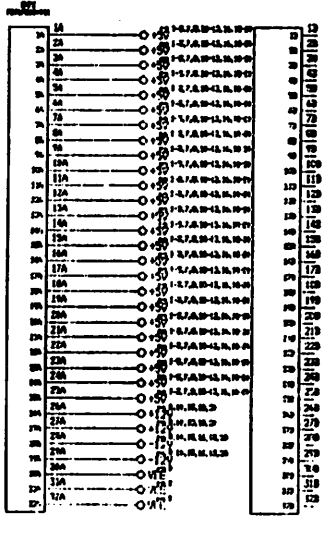
J1 - ENTIRELY USED

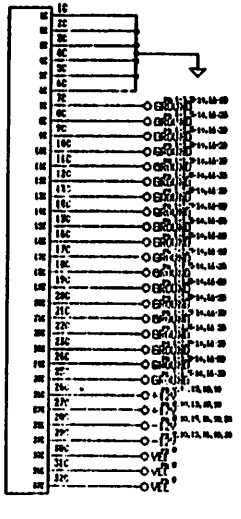


J2 - A & C ARE FOR USER DEFINED SIGNALS



J3 - B IS THE SUMBUS



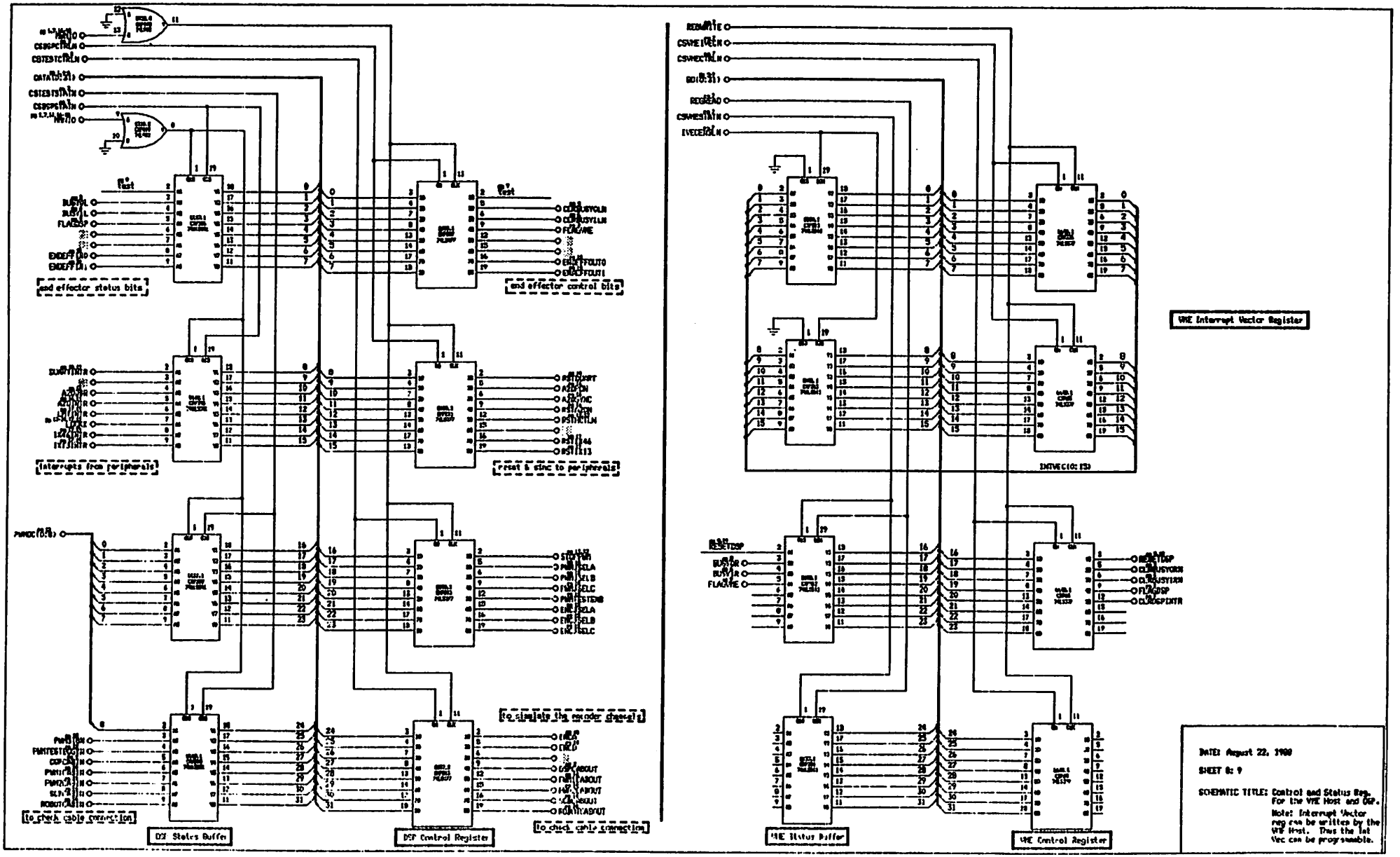


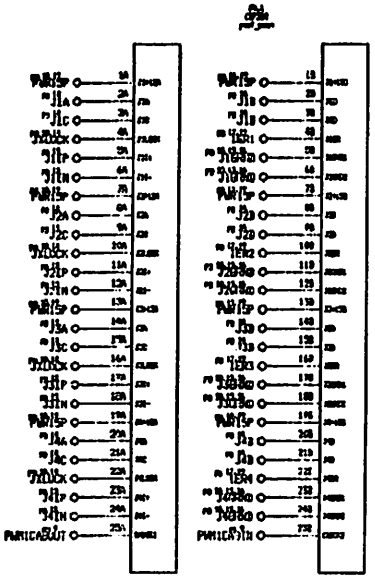
to ensure that the analog ground is kept seperate from the digital ground the two are merged at the backplane. With six pins chosen as analog ground the 8 of 450 pins = 6 of GROUND pins. This power supply current and ground current per pin is equalised.

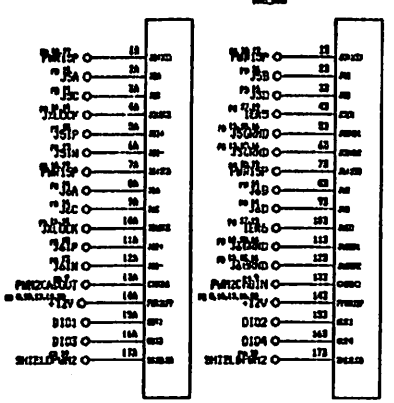
DATE: August 3, 1988

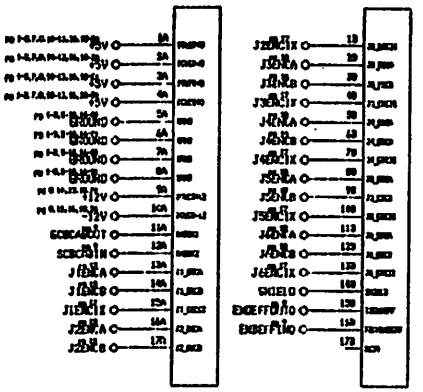
SHEET: 8

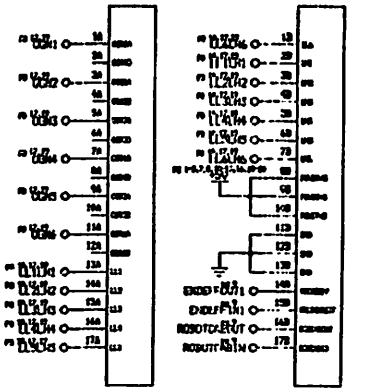
SCHEMATIC NAME: VME Bus Edge Connectors


end effector status bits
end effector control bits
interrupts from peripherals
reset & sinc to peripherals
to simplify the reader drawing
to check cable connection
to check cable connection
DSP Status Buffer
DSP Control Register
VME Status Buffer
VME Control Register
VME Interrupt Vector Register
DATE: August 22, 1990
SHEET 8: 9
SCHEMATIC TITLE: Control and Status Reg. For the VME Host and DSP.
Note: Interrupt Vector reg can be written by the VME Host. Thus the Int Vec can be programmable.









PAR1 and PAR2 Connectors

PAR1P - is +13V to be provided to the servo box to operate the opto couplers. This is supply is referenced to the ground that Jxa,B,C &D are ref. to.
Jxa,B,C & D come from the Pulse W Gig. PLD via transistor array (ITD625IP).
JxIP and JxIN are current feedback signals to the A/D via the analog filters.
IERx - is the current error signal from the servo amp board.
ERRD - JxIP and JxIN and IERx are referenced to this ground signal.
LOCKx - is the common joint lock signal.
PARxCABOUT - comes from the DSP control register and PARxCABIN goes to the DSP status register - and the two together can be used to verify the cable connection.
DIOx - is provided for Digital Input Output by Panasonic. (currently unused).

Signal Conditioning Board Connector

JxENCA - is the A channel from the rotary encoder
JxENCB - is the B channel from the rotary encoder
JxENCIX - is the index pulse from the rotary encoder.
+5V - is the power supply to the S. C. Board.
GROUND - is the reference for the +5V supply.
SCBCABOUT - comes from the DSP control reg. and SCBCABIN goes to the DSP status reg. These pins must be shorted on the other end of the cable to enable DSP to check connection.
SHIELD - is connected to the sheath.
ENDEFFOUTO is a pin (DSP control reg) to control the end-effector and ENDEFFIND is a pin to monitor end-eff's states.

Robot Signal Connector (origin and limit)

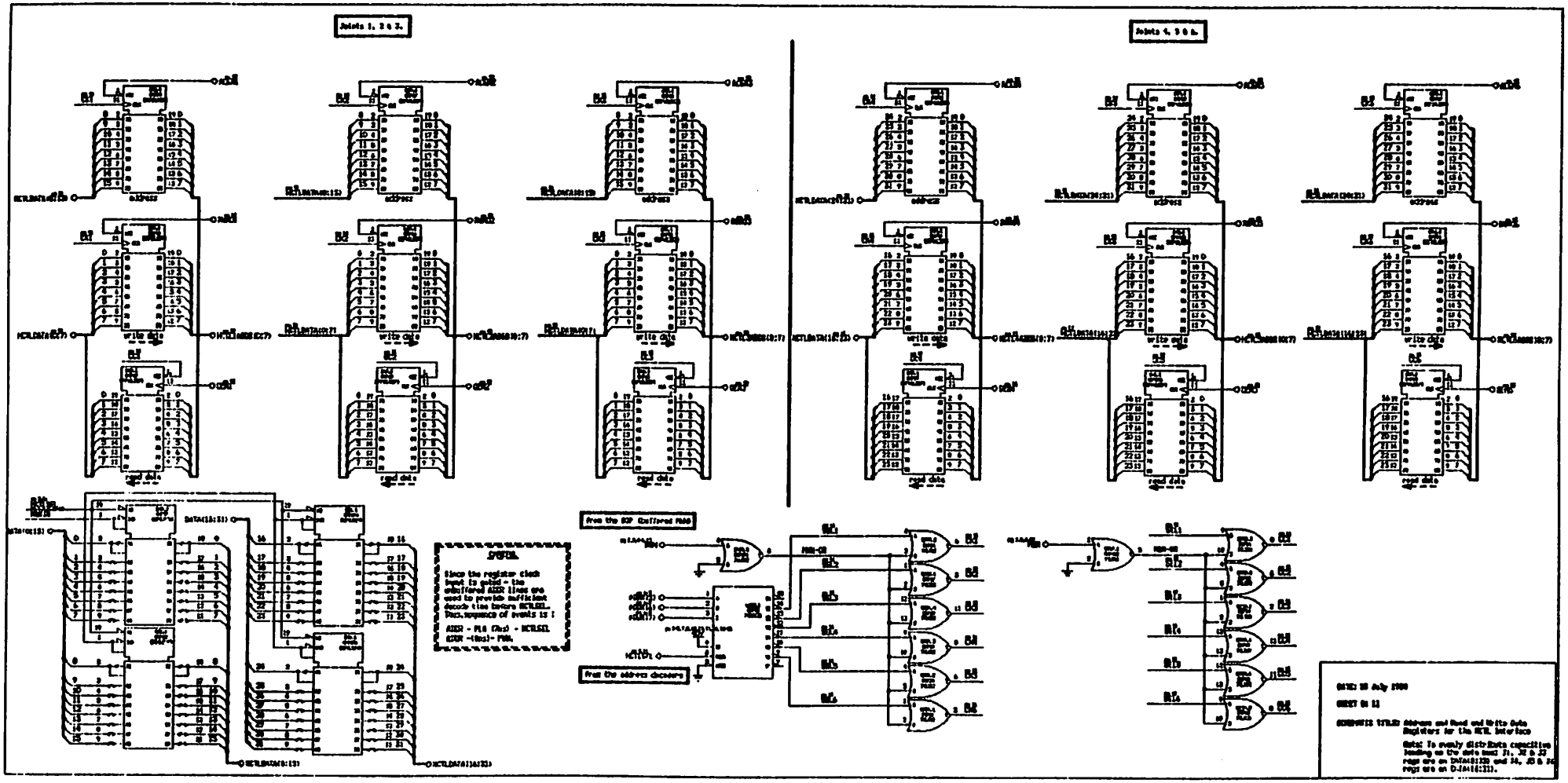
ORNA - is the Origin signal.
ORNB - is unused - but is provided to connect the different origin signals mentioned in the PANAMOTO manual (slow1 slow2 etc)
LLx - is the low limit signal
LLx - is the high limit signal
LLxLHx - since the limit signals are open collector - they have been wire ored to this signal. The direction of the limit is ascertained by monitoring the origin signal.
ROBOTCABOUT - comes from the DSP control reg and ROBOTCABIN goes to the DSP status reg. These must be shorted on the other end of the cable to enable the DSP to check cable connection.
ENDEFFOUT1 and ENDEFFIN1 are control and status pins for the end-effector.

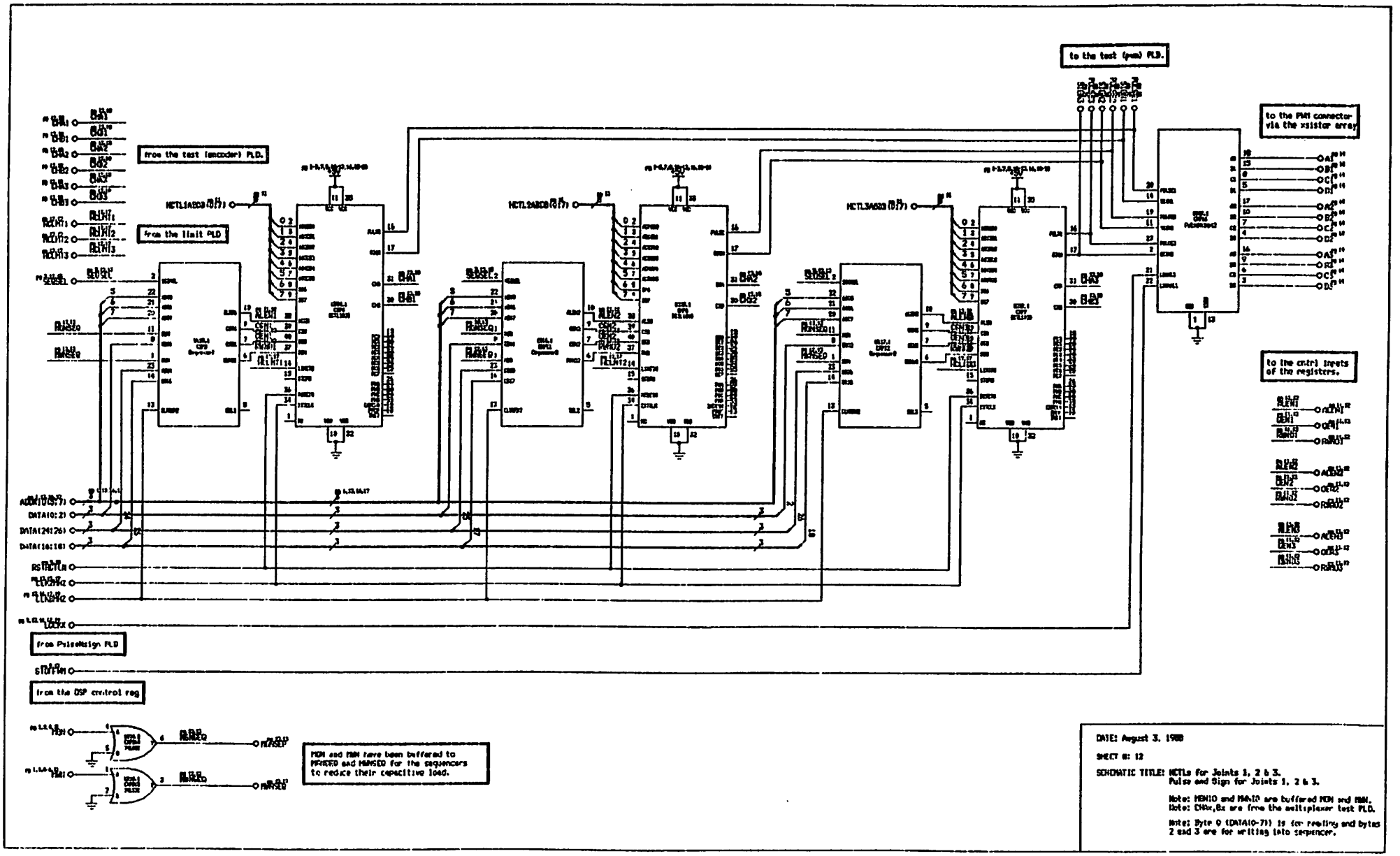
DATE: August 17, 1988

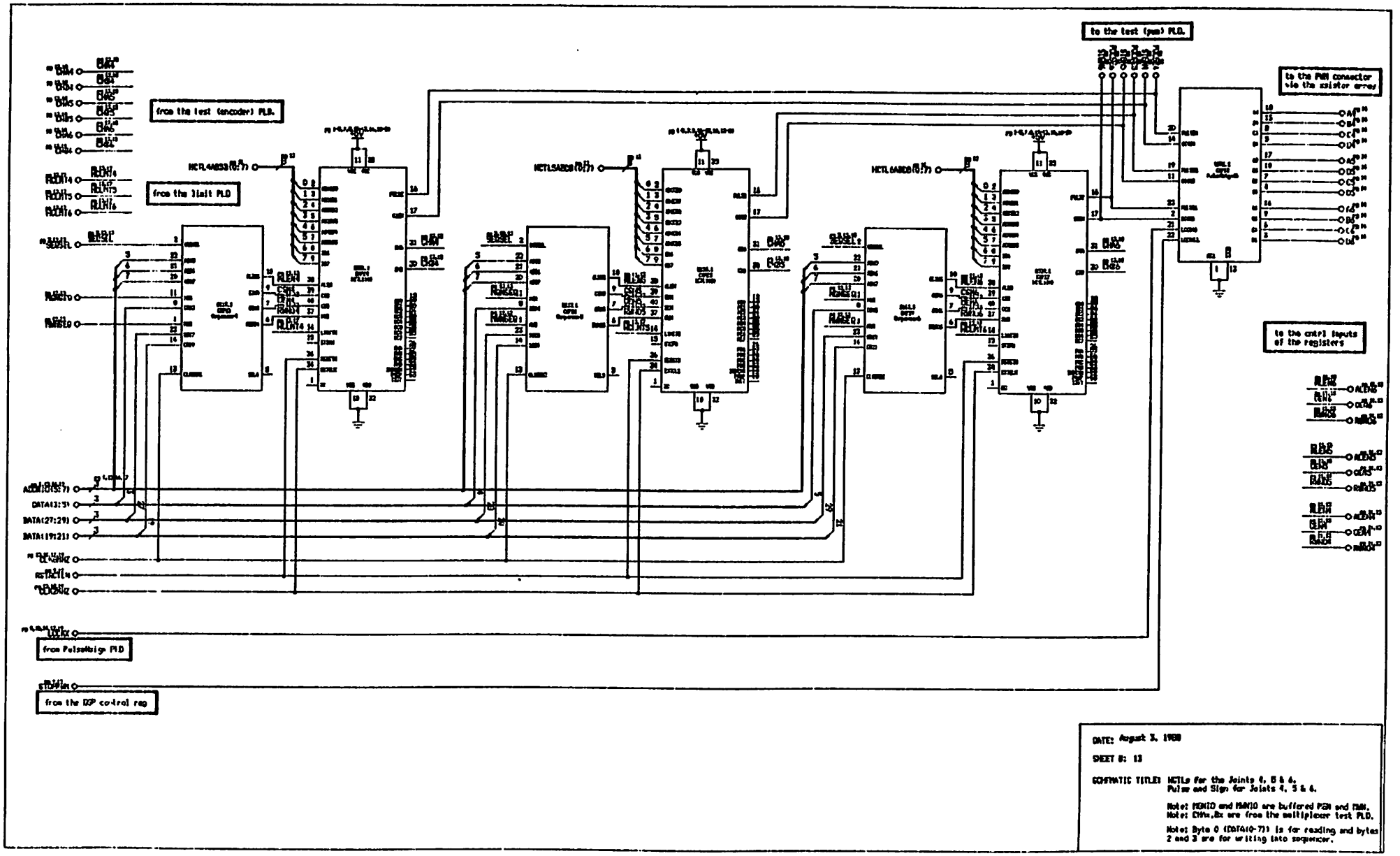
SHEET #: 10

SCHEMATIC TITLE: Edge connectors (for signals to & from the robot)

PAR1 (A,B,C,D,IER,Ix,I- for J1-J4)
PAR2 (A,B,C,D,IER,Ix,I- for J5-J6)
SLO (ENCIX, ENLF, ENCB for J1-J6;
ROBOT (LOR, LL, LH for J1-J6)




to the test (pwm) PLD.
to the PWM connector via the xsistor array
from the test (encoder) PLD.
from the limit PLD
to the cntrl inputs of the registers.
from PulseSign PLD
from the DSP control reg
MDN and MAN have been buffered to MDNSEQ and MANSEQ for the sequencers to reduce their capacitive load.
DATE: August 3, 1988
SHEET #: 12
SCHEMATIC TITLE: MCTLs for Joints 1, 2 & 3.
Pulse and Sign for Joints 1, 2 & 3.
Note: MENIO and MAIO are buffered MDN and MAN.
Note: CH4x,8x are from the multiplexer test PLD.
Note: Byte 0 (DATA10-7) is for reading and bytes 2 and 3 are for writing into sequencer.



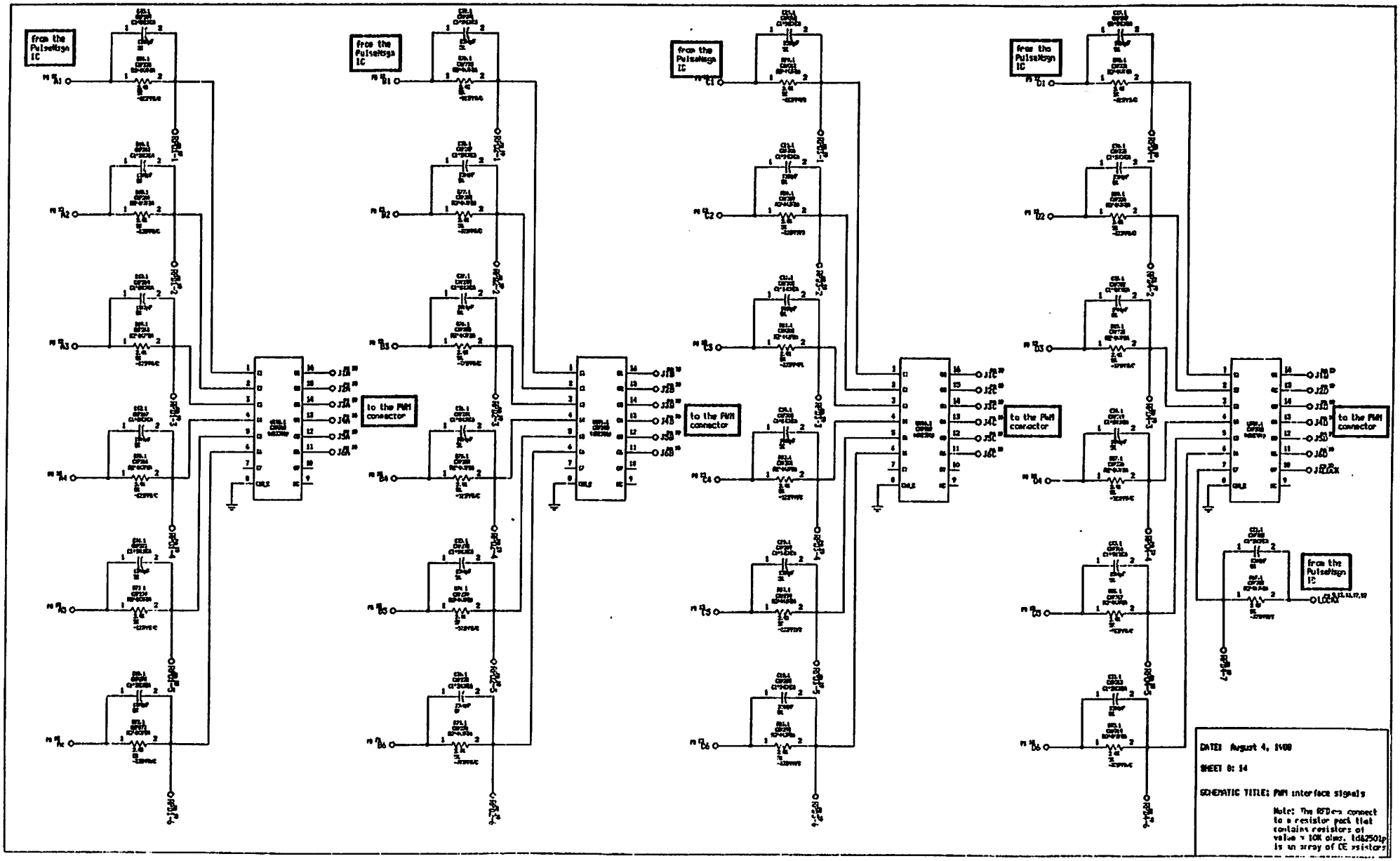
DATE: August 3, 1989

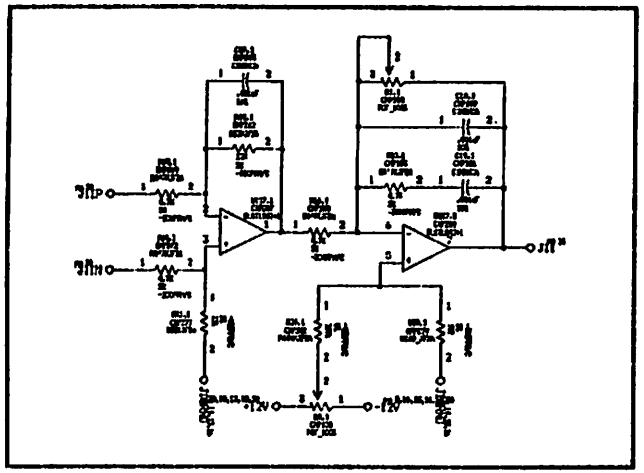
SHEET #: 13

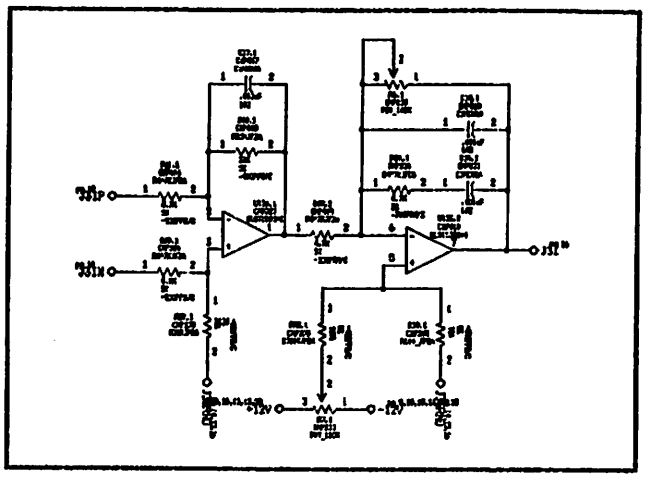
SCHEMATIC TITLE: HCTLs for the Joints 4, 5 & 6.
Pulse and Sign for Joints 4, 5 & 6.

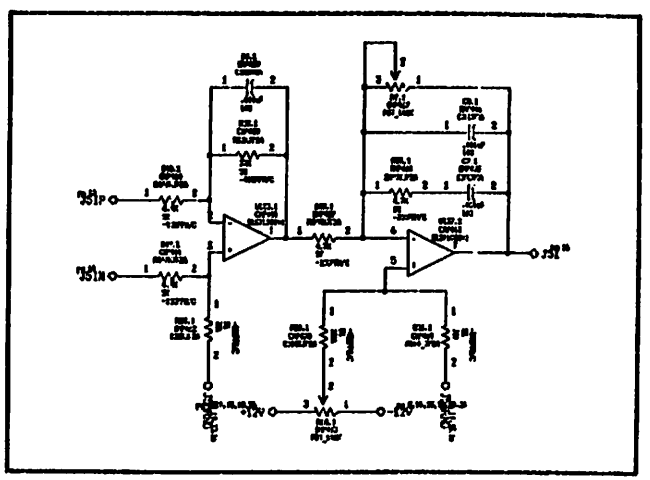
Note: PENIO and PWMIO are buffered PEN and PWM.
Note: CNTx..0x are from the multiplexer test PLD.

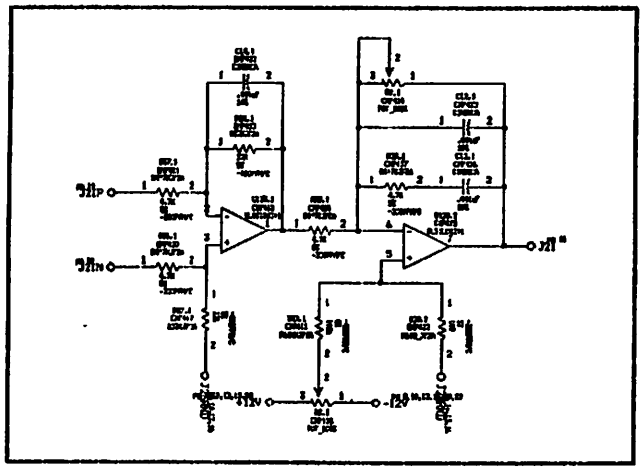
Note: Byte 0 (DATA(0-7)) is for reading and bytes 2 and 3 are for writing into sequencer.

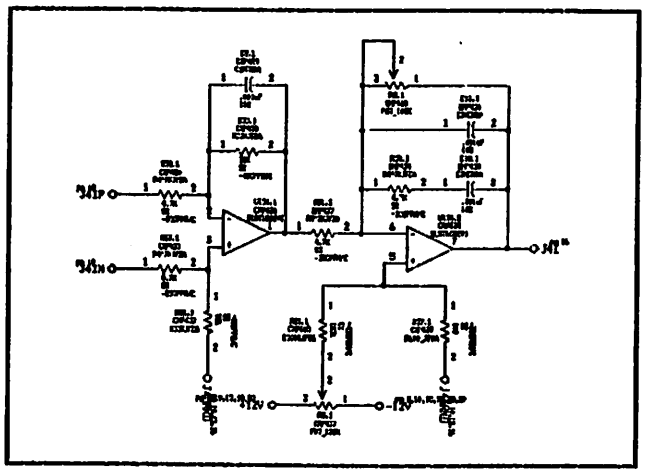

from the Pulsetrain IC
from the Pulsetrain IC
from the Pulsetrain IC
from the Pulsetrain IC
to the PWM connector
to the PWM connector
to the PWM connector
to the PWM connector
from the Pulsetrain IC
DATE: August 4, 1988
SHEET 8: 14
SCHEMATIC TITLE: PWM interface signals
Note: The RPD-s connect to a resistor pack that contains resistors of value = 10K ohms. 1d42501p is an array of CE resistors

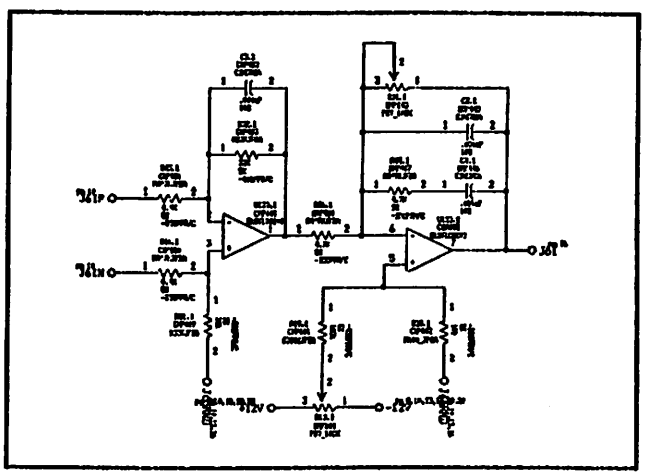










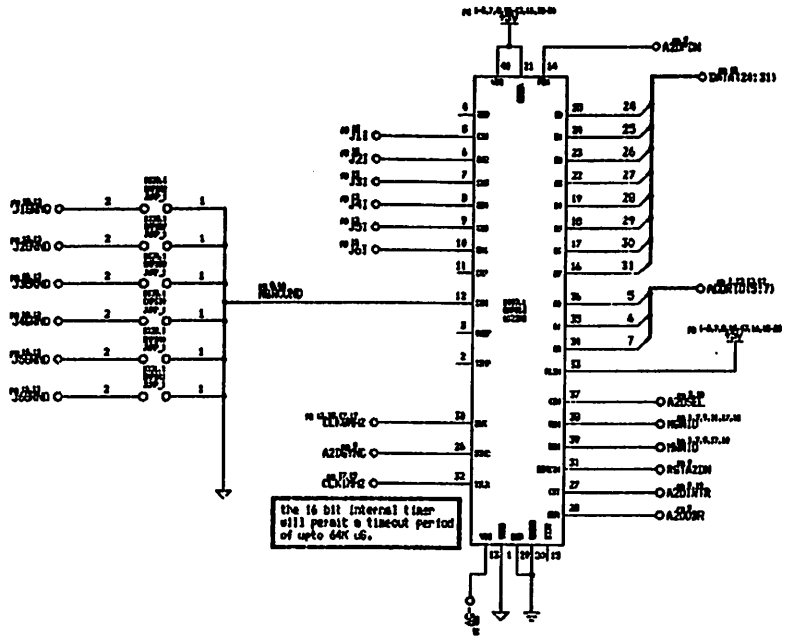


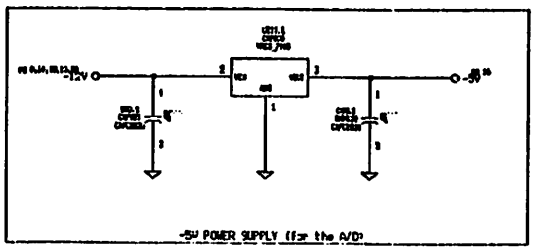
DATE: August 8, 1988

SHEET #: 15

SCHEMATIC TITLE: Filters for the current feedback signals (I+ and I-).

Note: JxGND are the grounds for the joints as they arrive from the servo boards. These grounds are merged with the analog ground at the A/D conv and then with the digital ground at the VME edge connectors.



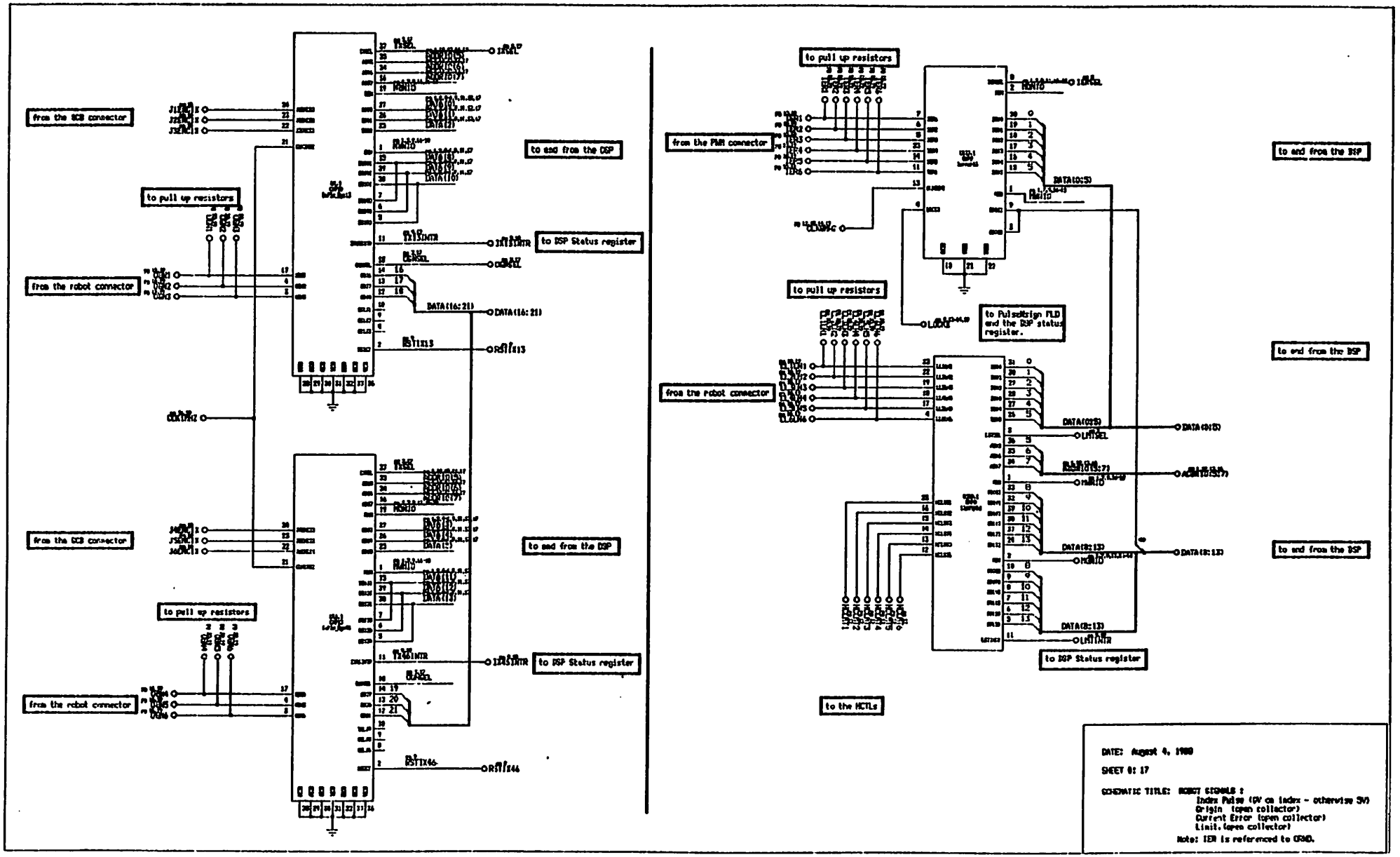


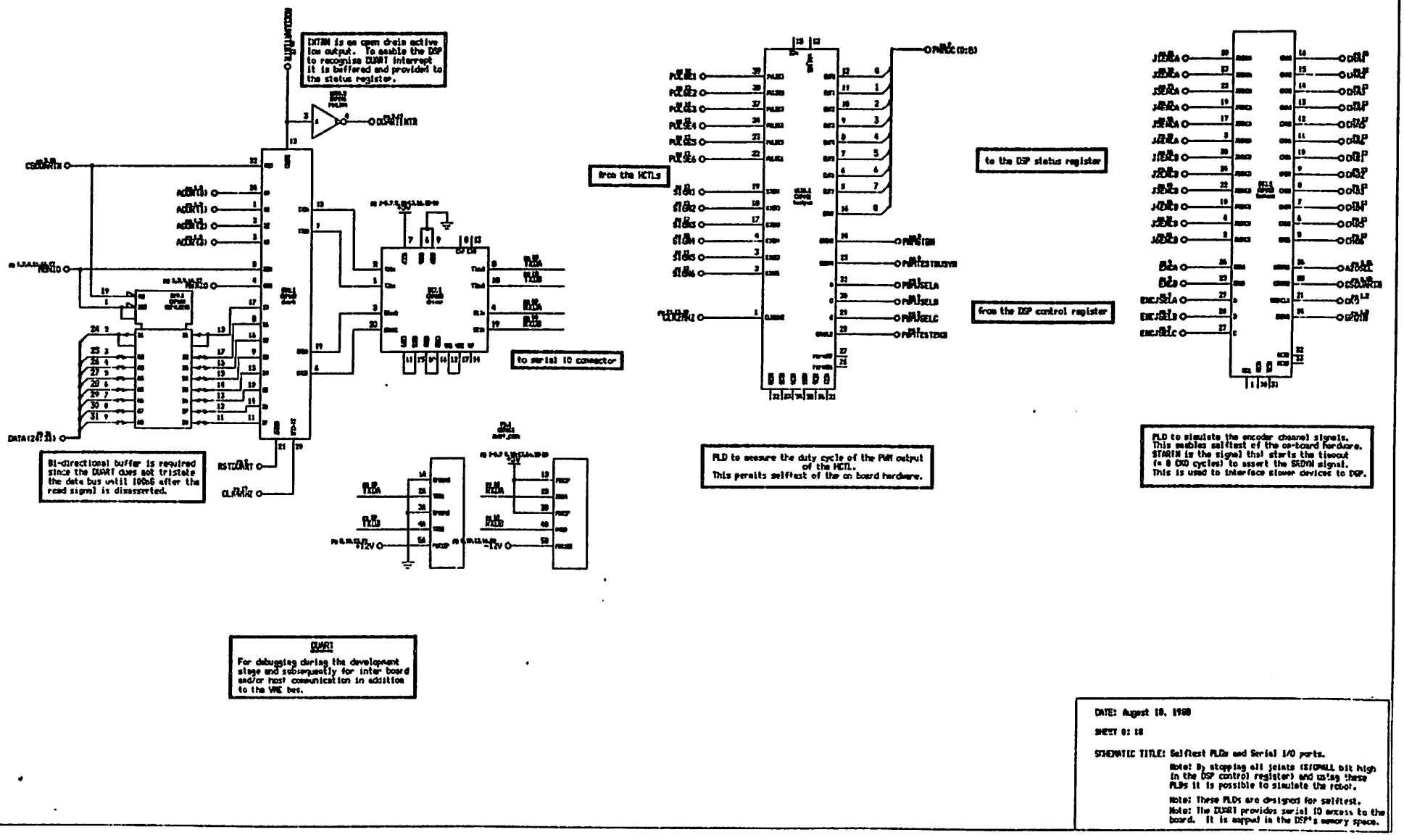
DATE: August 29, 1988

SHEET 8: 16

SCHEMATIC TITLE: Analog to Digital Converter

Note: The analog filters remove high freq noise and provide a signal proportional to the current in the joint. This A2D conv provides this current value to the DSP.


from the SCB connector
to pull up resistors
from the robot connector
to and from the DSP
to DSP Status register
from the SCB connector
to pull up resistors
from the robot connector
to and from the DSP
to DSP Status register
from the PWR connector
to pull up resistors
to and from the DSP
to pull up resistors
to Pulsebrign FLD and the DSP status register.
from the robot connector
to and from the DSP
to and from the DSP
to DSP Status register
to the MCTLs
DATE: August 4, 1988
SHEET 8: 17
SCHEMATIC TITLE: ROBOT SIGNALS :
Index Pulse (0V on index - otherwise 5V)
Origin (open collector)
Current Error (open collector)
Limit (open collector)
Note: IER is referenced to GRND.



DATE: August 19, 1988

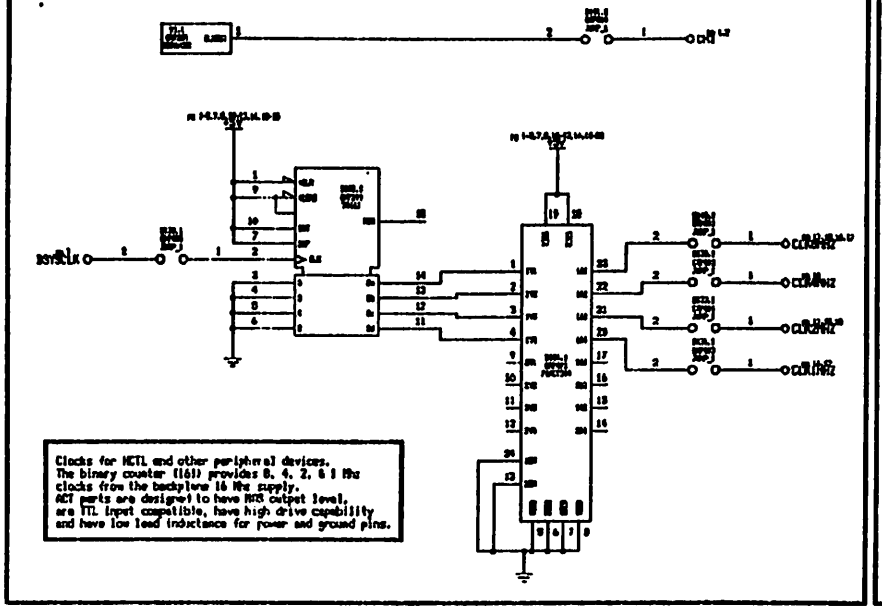
SHEET 0: 18

SCHEMATIC TITLE: Selftest PLDs and Serial I/O ports.

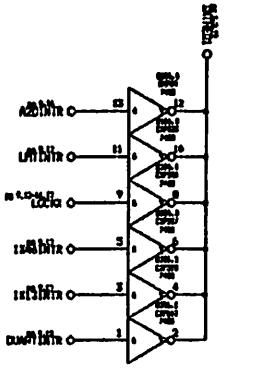
Note: By stopping all joints (STOPALL bit high in the DSP control register) and using these PLDs it is possible to simulate the robot.

Note: These PLDs are designed for selftest.

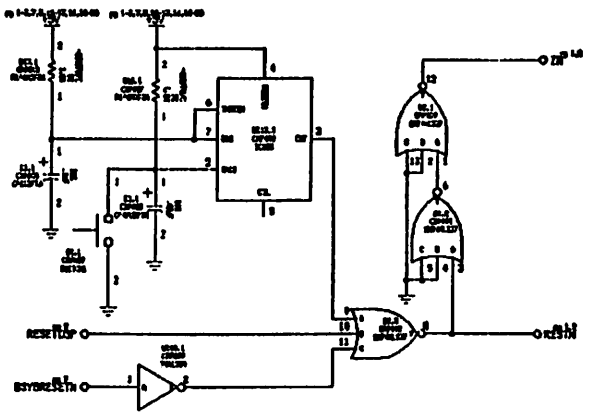
Note: The DUART provides serial IO access to the board. It is mapped in the DSP's memory space.



Clocks for HCTL and other peripheral devices.
The binary counter (161) provides 8, 4, 2, & 1 Mhz clocks from the backplane 16 Mhz supply.
ACT parts are designed to have MOS output level, are TTL input compatible, have high drive capability and have low lead inductance for power and ground pins.

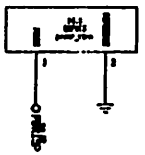


Different interrupting sources are connected to open collector inverters so that they may be wire ored to the DSP interrupt.

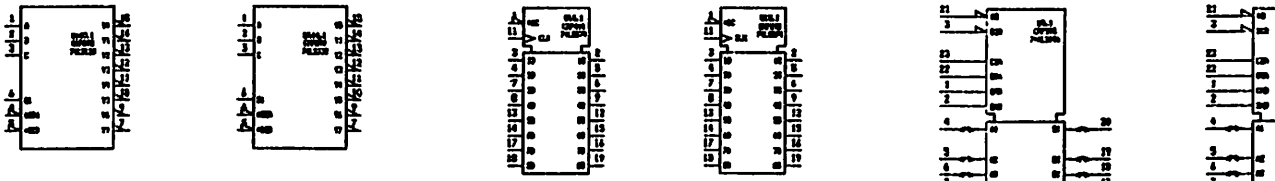


RESET ciruit. The board may be reset using the switch, or when the SYSRESETN line on the VME backplane is activated or when the VME hosts writes 1 to the RESETDSP bit in the VME control register.

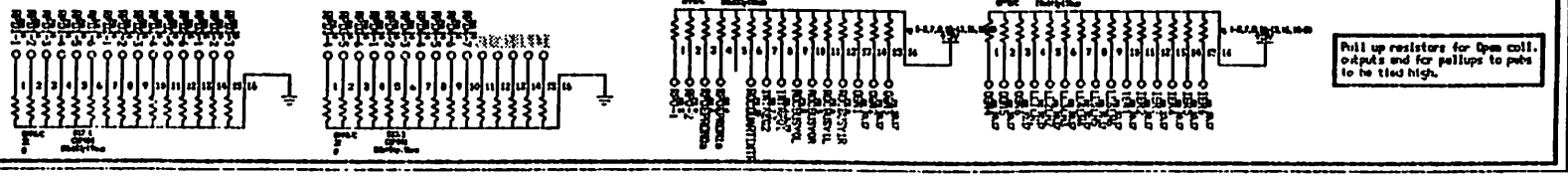
Note: As required by the DSP32C data sheets IN is disasserted after RESTN.



Off board Power connector to obtain 15V power for the servo board opto couplers. PWR15P is connected to PWM connectors.



16 pin, 20 pin and 24 pin spares (2 each)

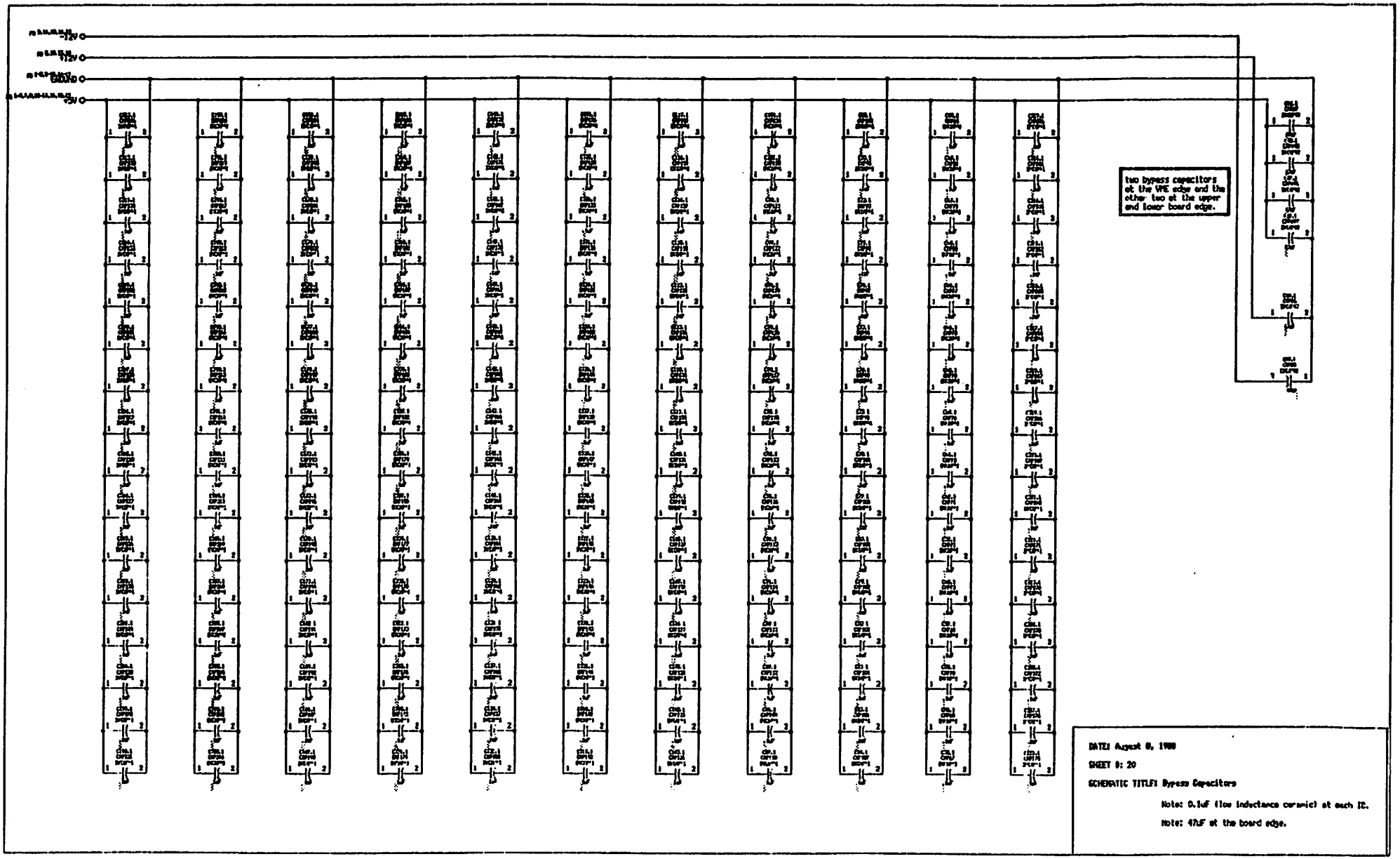


Pull up resistors for Open coll. outputs and for pullups to pots to be tied high.

DATE: September 2, 1988

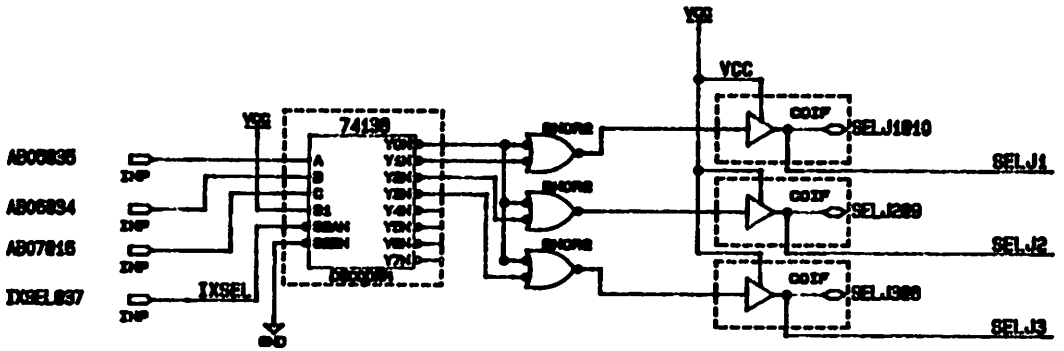
SHEET #: 19

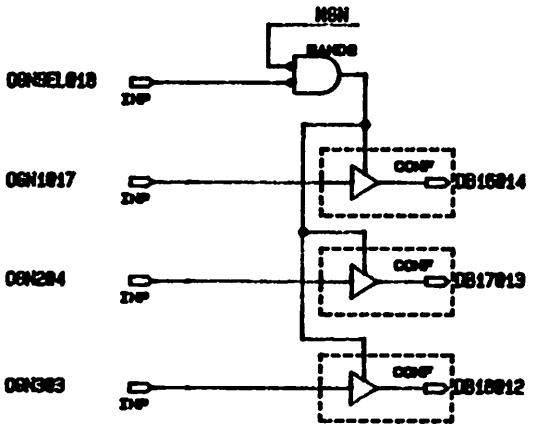
SCHEMATIC TITLE: Miscellaneous Circuits.
DSP Clock, Peripheral clocks, Reset, Pull up R-packs, R-pack for bese discharge resistors for tdA2301p, OC inverters for DSP interrupts, +15V connector, spare IC sockets (16, 20 & 24 pin)

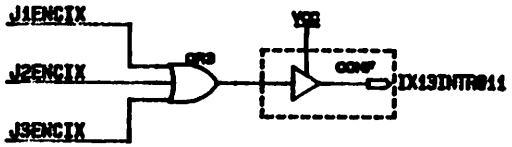

+12V
+12V
GND
GND
two bypass capacitors at the WE edge and the other two at the upper and lower board edge.
DATE: August 8, 1988
SHEET 9: 20
SCHEMATIC TITLE: Bypass Capacitors
Note: 0.1uF flow inductance ceramic at each IC.
Note: 47uF at the board edge.

Appendix C

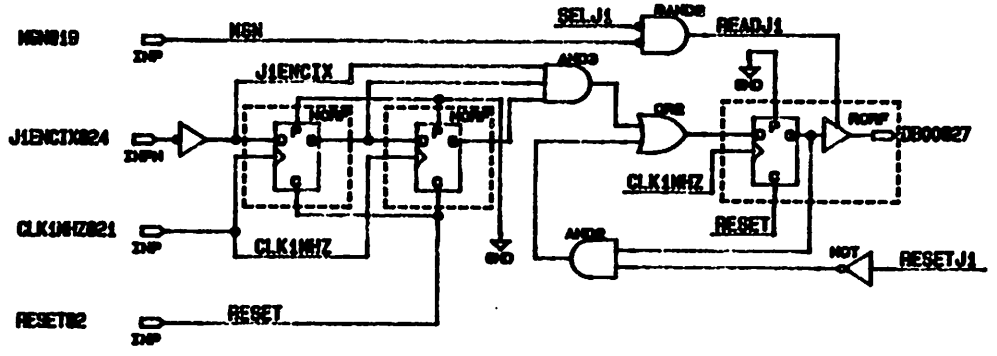
PLD Schematics

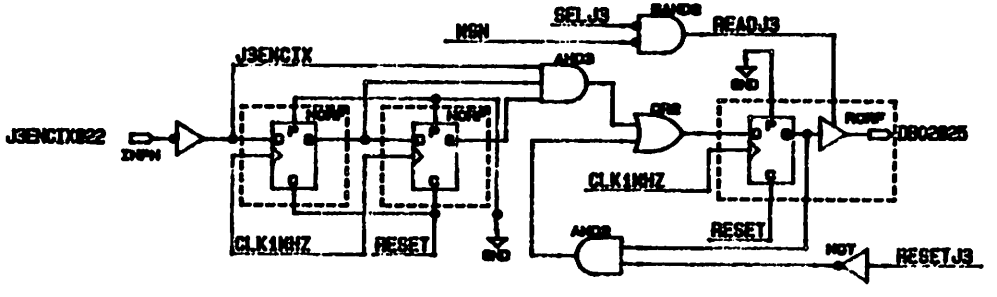


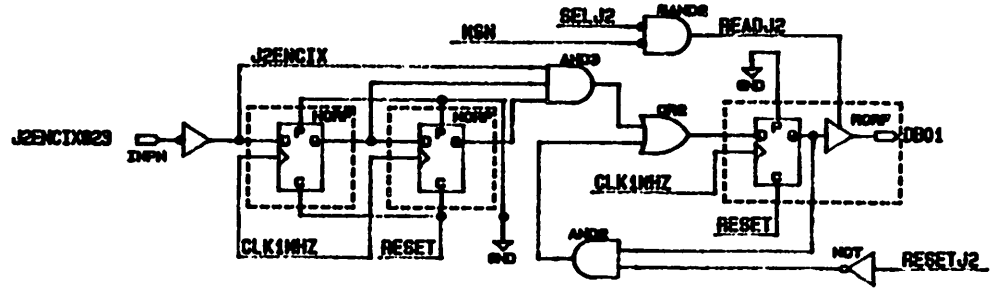


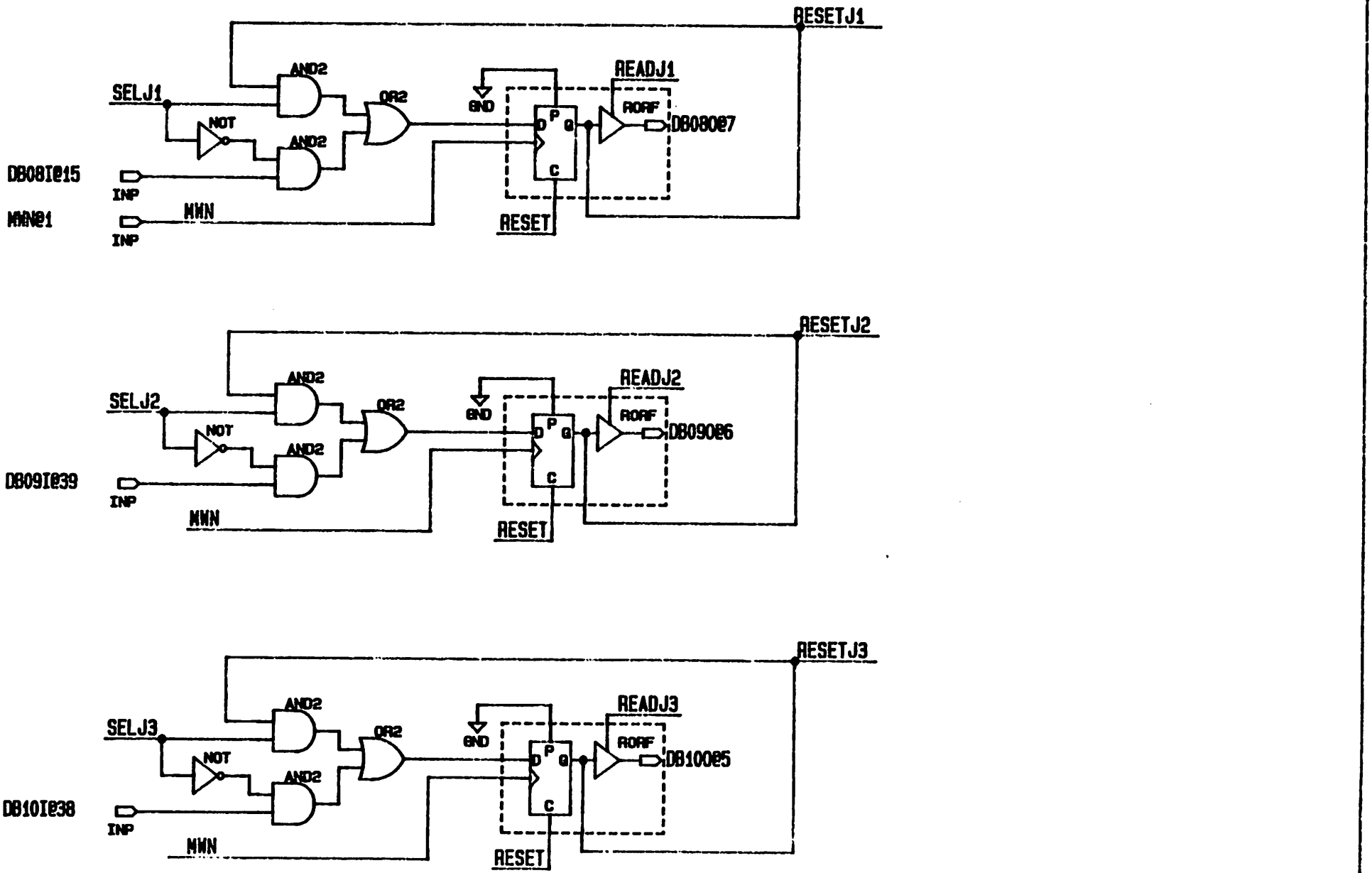


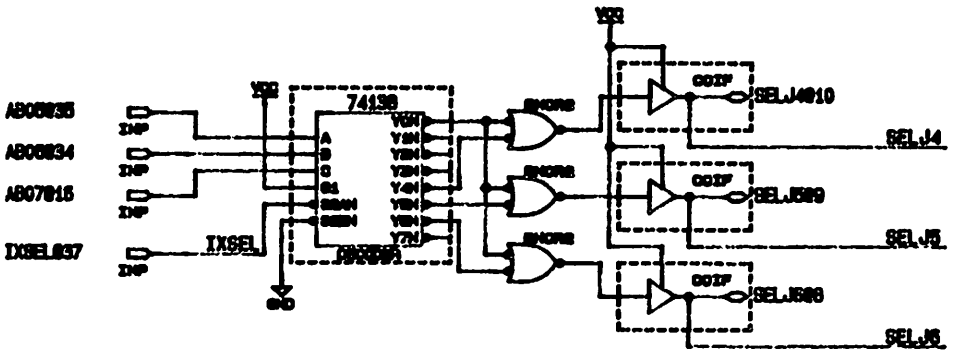
COMPANY	U C BERKELEY
TITLE	INDEX PULSE (J1-J3)
DRAWN	Gautam Doshi
SIZE B / SHLD EP910D	DRAWN 1.00 / REV A
DATE June 1988	SHEET 1 OF 2
TURBO ON	SECURITY OFF

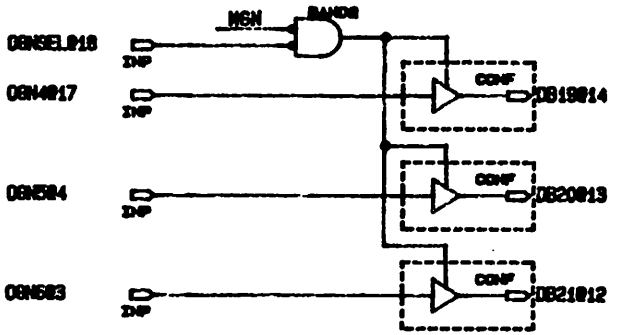

MSN019
INP
MSN
SELJ1
AND8
READJ1
J1ENCIX
AND3
J1ENCIX024
INPN
16BIP
16BIP
OR8
RORF
DB00827
CLK1MHZ
CLK1MHZ021
INP
CLK1MHZ
RESET
AND8
NOT
RESETJ1
RESET02
INP
RESET

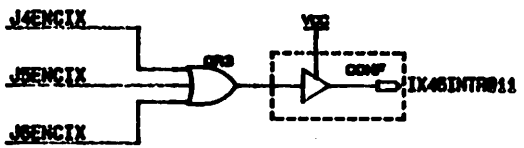

SELJ3
AND8
READJ3
MSN
J3ENCIX
AND3
J3ENCIX022
INPN
16BIP
16BIP
OR8
RORF
DB02825
CLK1MHZ
CLK1MHZ
RESET
AND8
RESET
NOT
RESETJ3


SELJ2
AND8
READJ2
MSN
J2ENCIX
AND3
J2ENCIX023
INPN
16BIP
16BIP
OR8
RORF
DB01
CLK1MHZ
CLK1MHZ
RESET
RESET
AND8
NOT
RESETJ2

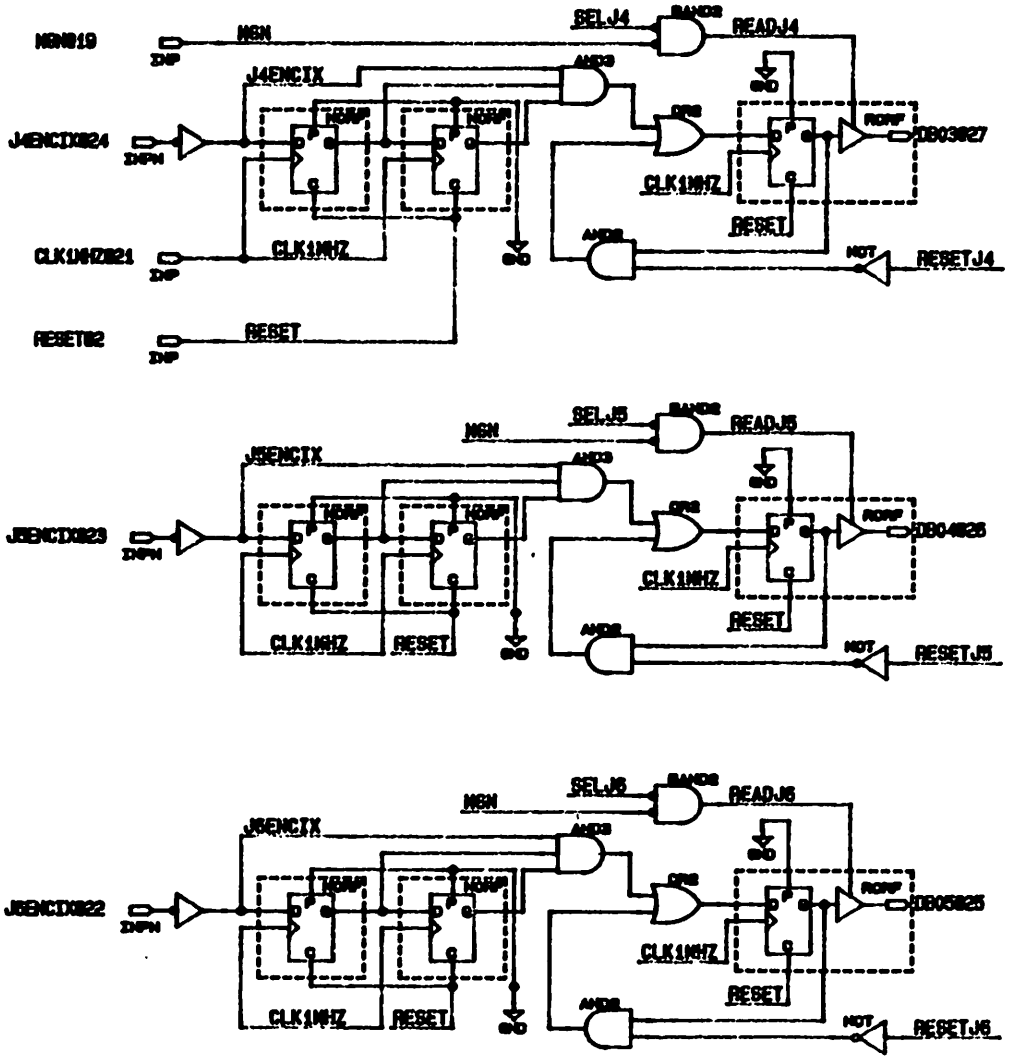

RESETJ1
AND2
SELJ1
OR2
GND
READJ1
NOT
AND2
RORF
DB08B07
DB08I@15
INP
C
MWN@1
MWN
INP
RESET
RESETJ2
AND2
READJ2
SELJ2
OR2
GND
NOT
AND2
RORF
DB09B06
DB09I@39
INP
C
MWN
RESET
RESETJ3
AND2
READJ3
SELJ3
OR2
GND
NOT
AND2
RORF
DB10B05
DB10I@38
INP
C
MWN
RESET

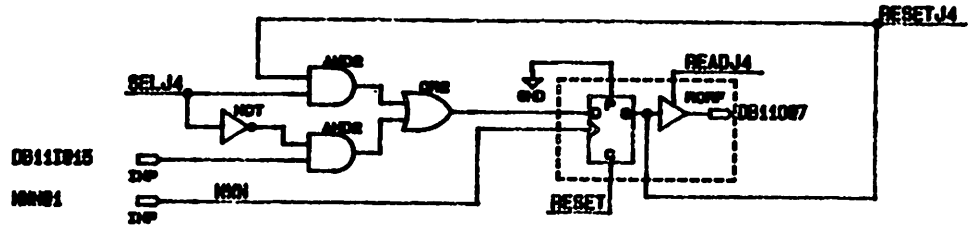


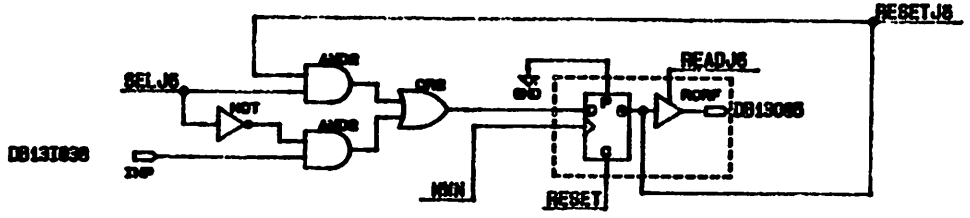


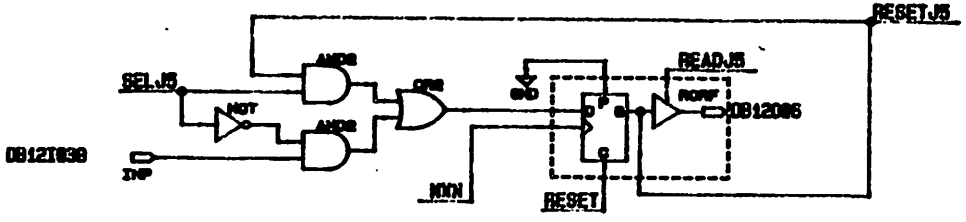


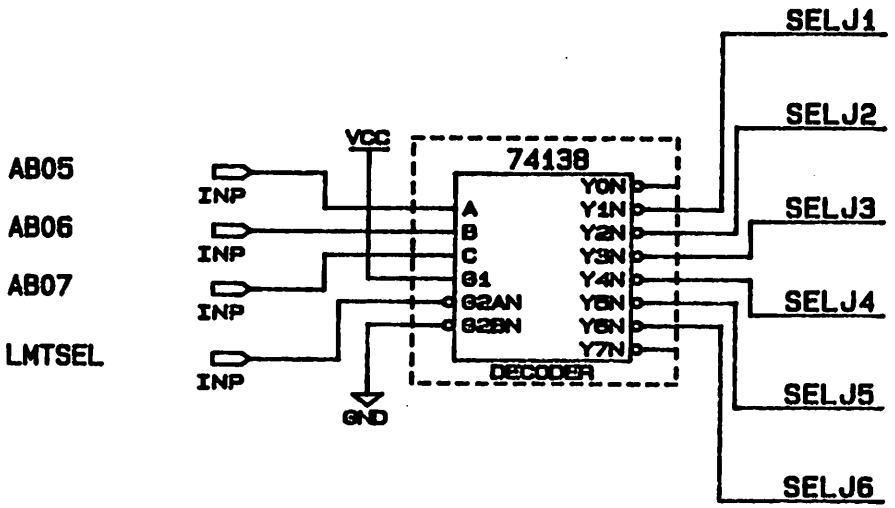
COMPANY	U C BERKELEY
TITLE	INDEX PULSE (J4-J6)
DRAWN BY	Gautam Doshi
SIZE B / BPLD EP910D	ALGRTHM 1.00 / REV A
DATE June 1988	SHEET 1 OF 3
TUNED ON	SECURITY OFF

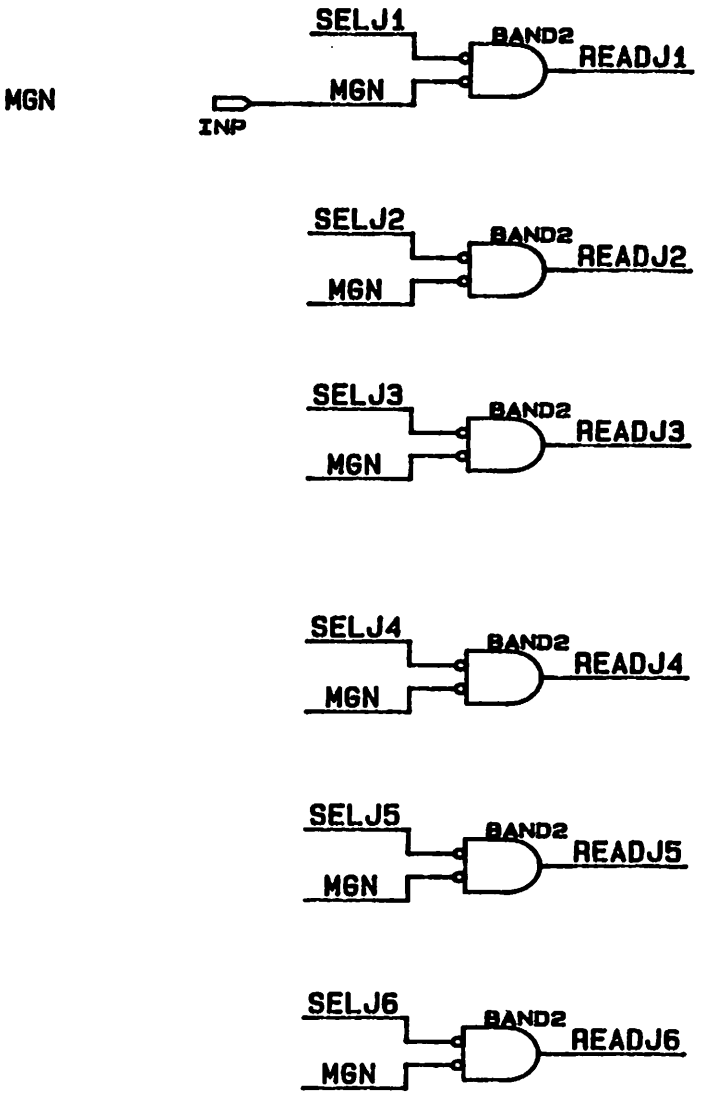

MSN19
MSN
SELJ4
READJ4
J4ENCIX
J4ENCIX824
CLK1MHZ824
RESET82
CLK1MHZ
RESET
CLK1MHZ
RESET
DB03827
RESETJ4
J5ENCIX
J5ENCIX823
CLK1MHZ
RESET
DB04826
RESETJ5
J6ENCIX
J6ENCIX822
CLK1MHZ
RESET
DB05825
RESETJ6


RESETJ4
SELJ4
NOT
AND2
OR2
AND2
DB11D915
INP
MMM1
INP
MMM
GND
READJ4
RDRF
DB11087
RESET

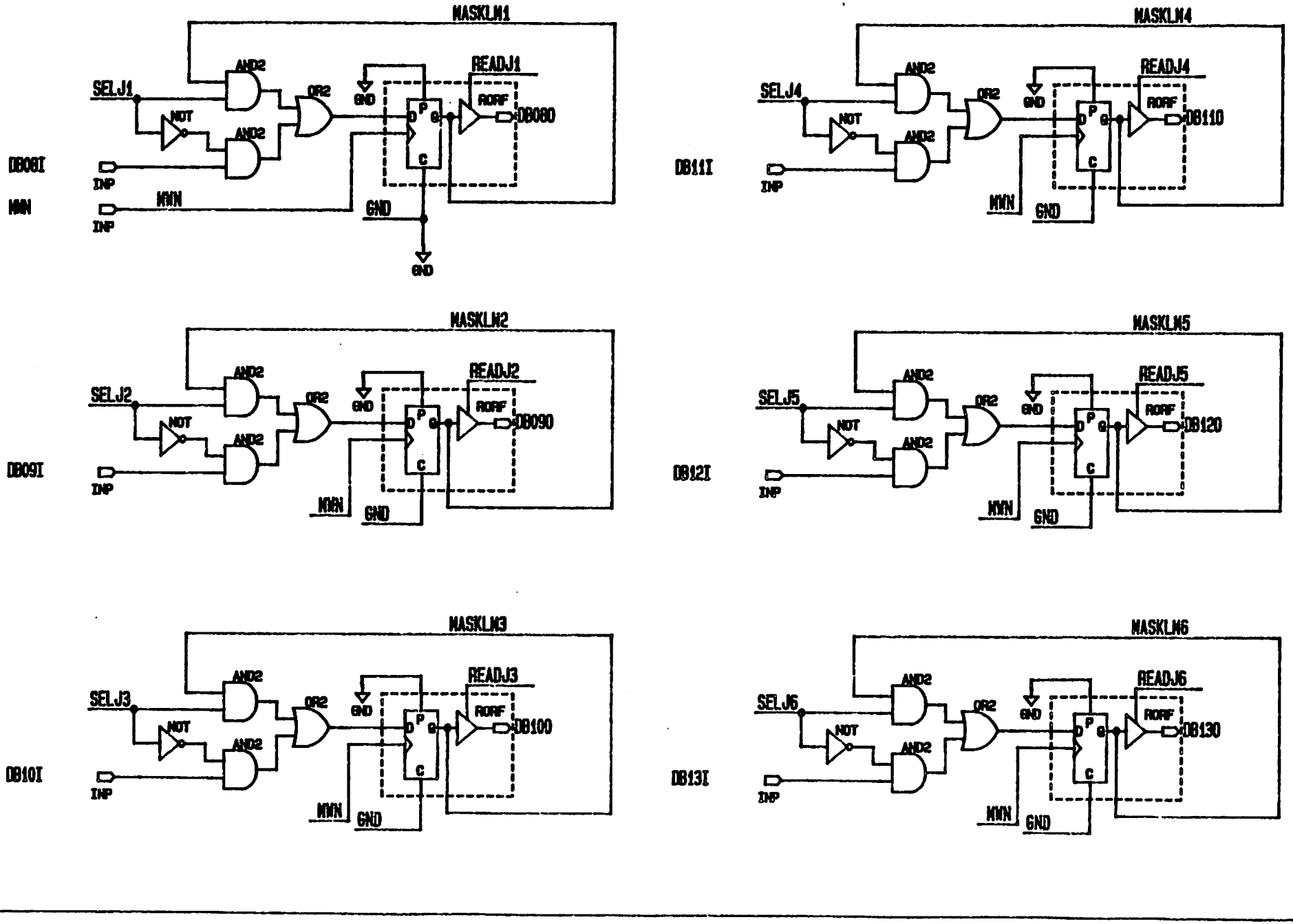

RESETJ5
SELJ5
NOT
AND2
OR2
AND2
DB13T838
INP
MMM
GND
READJ5
RDRF
DB13086
RESET

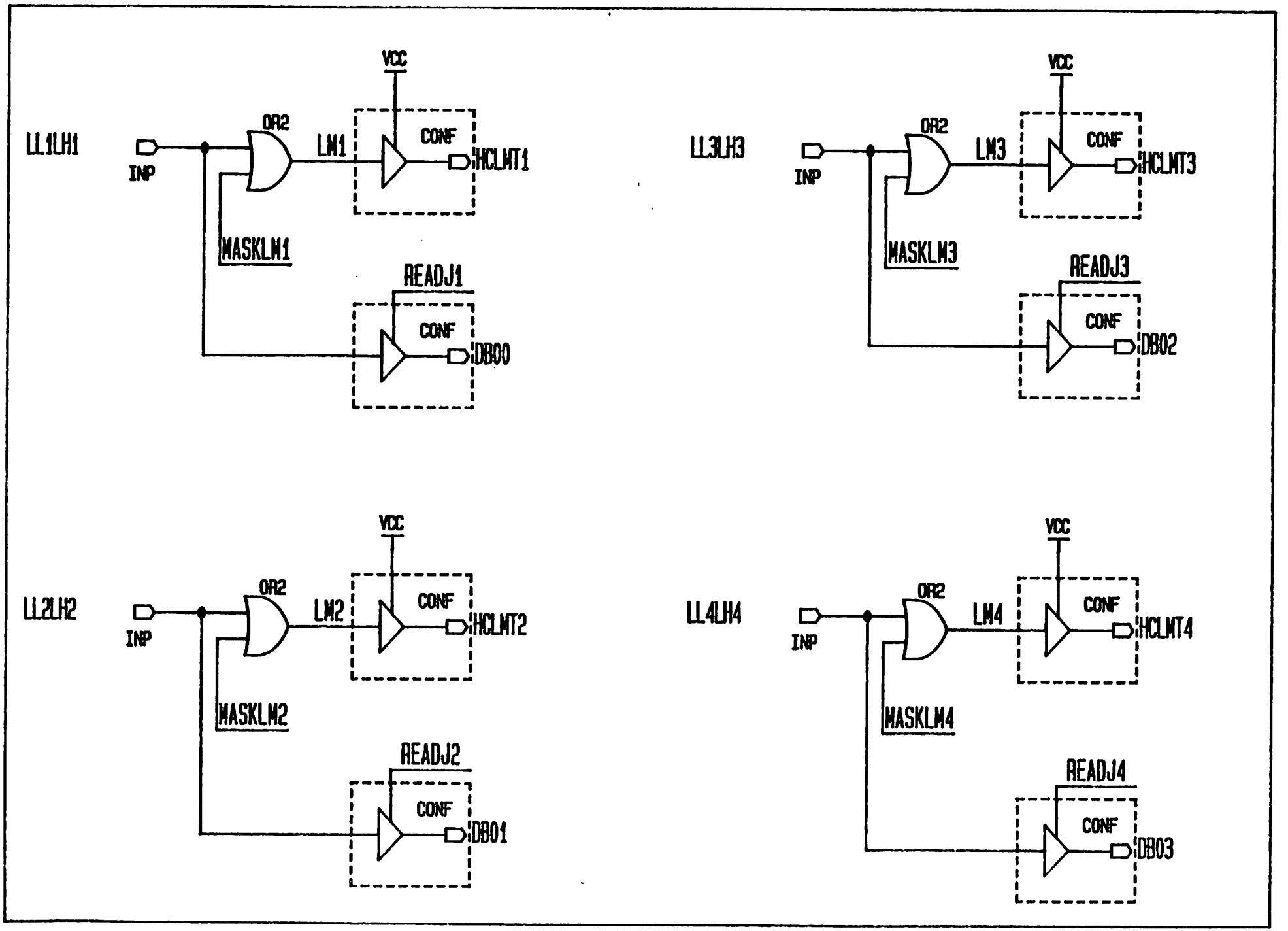

RESETJ5
SELJ5
NOT
AND2
OR2
AND2
DB12T839
INP
MMM
GND
READJ5
RDRF
DB12086
RESET

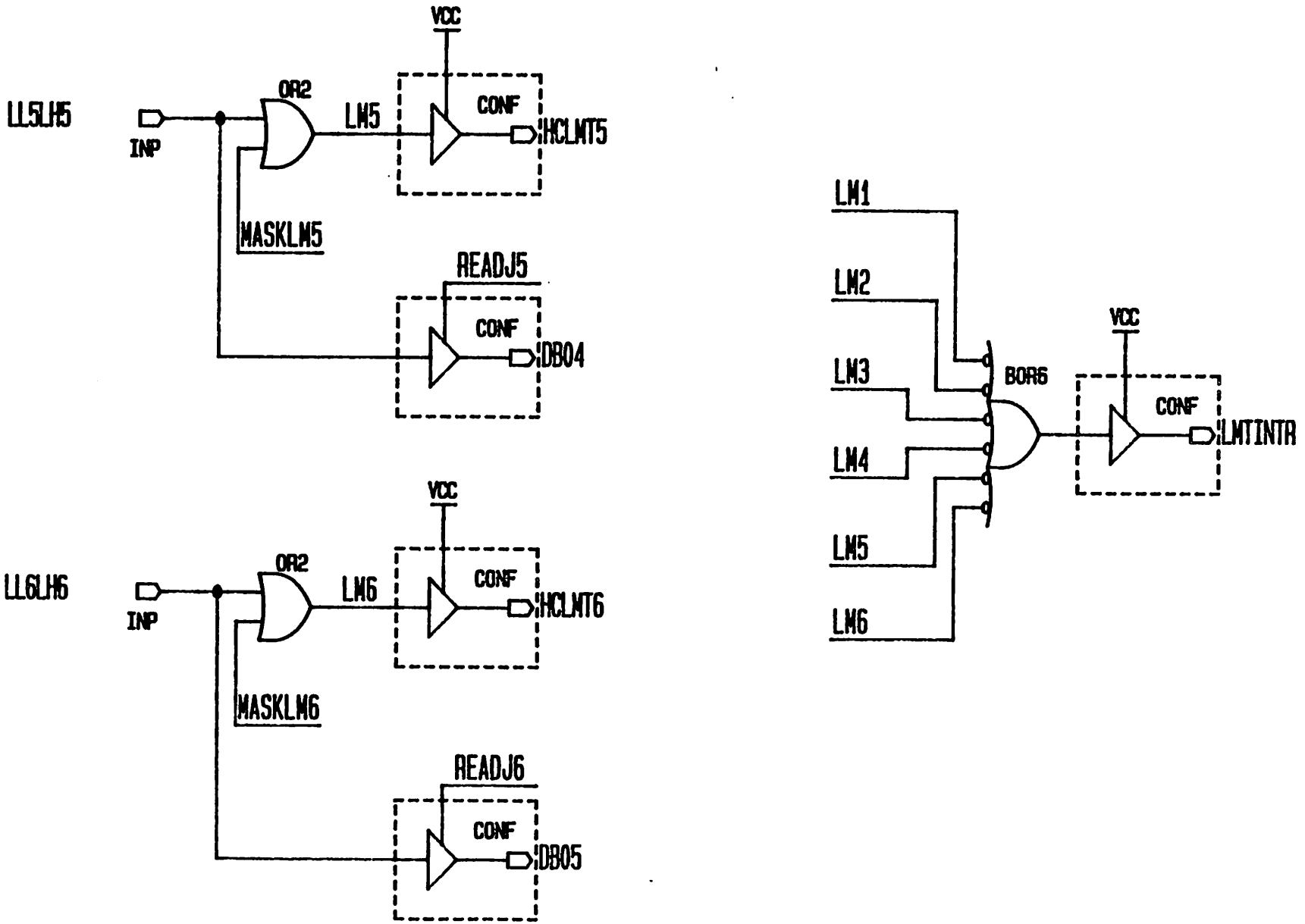


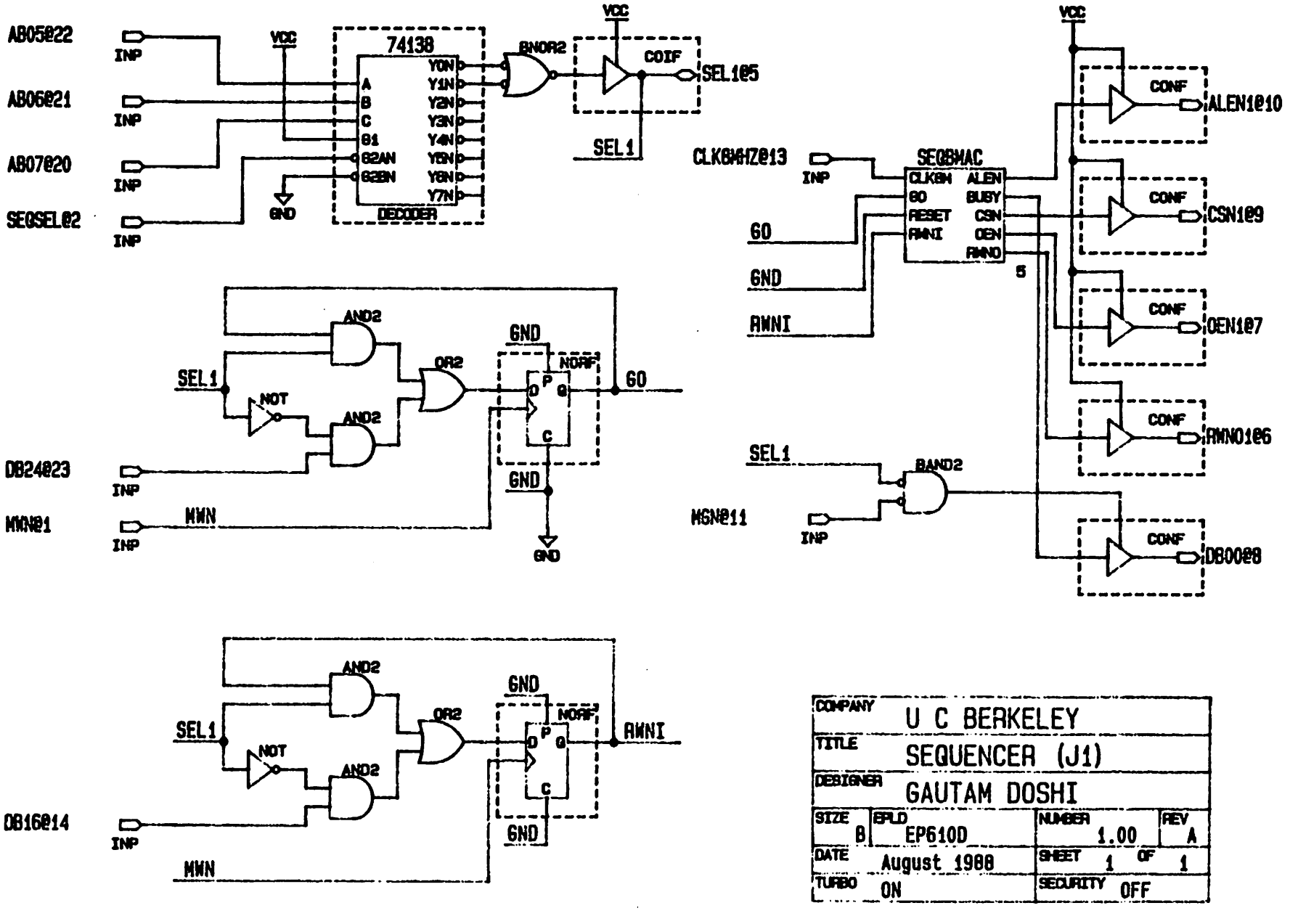


COMPANY	U C BERKELEY		
TITLE	LIMIT SIGNAL (J1-J6)		
DESIGNER	GAUTAM DOSHI		
SIZE B	EPLD EP910D	NUMBER 1.00	REV C
DATE June 1988		SHEET 1 OF 4	
TURBO ON		SECURITY OFF	

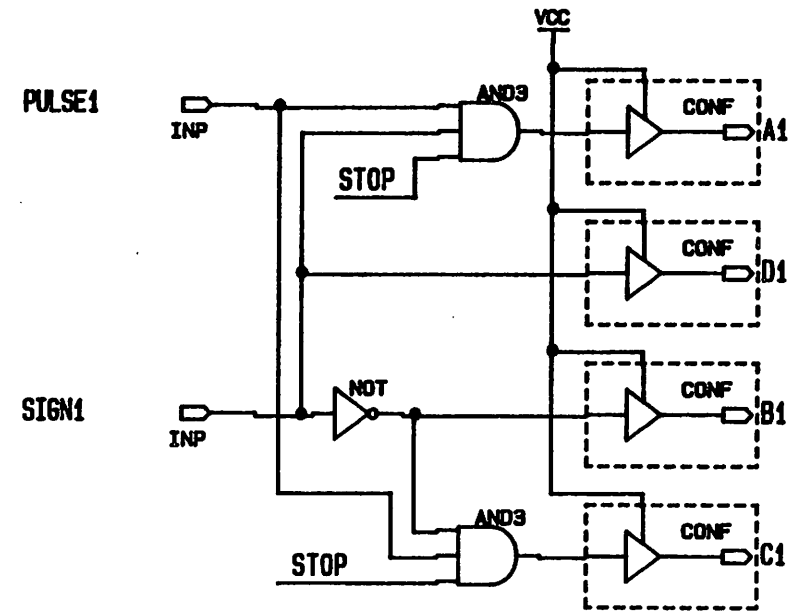

MASKLM1
READJ1
SELJ1
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB080
C
DB08I
INP
MWN
INP
MWN
GND
GND
MASKLM2
READJ2
SELJ2
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB090
C
DB09I
INP
MWN
GND
MASKLM3
READJ3
SELJ3
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB100
C
DB10I
INP
MWN
GND
MASKLM4
READJ4
SELJ4
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB110
C
DB11I
INP
MWN
GND
MASKLM5
READJ5
SELJ5
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB120
C
DB12I
INP
MWN
GND
MASKLM6
READJ6
SELJ6
AND2
OR2
NOT
AND2
GND
D
P
Q
RORF
DB130
C
DB13I
INP
MWN
GND

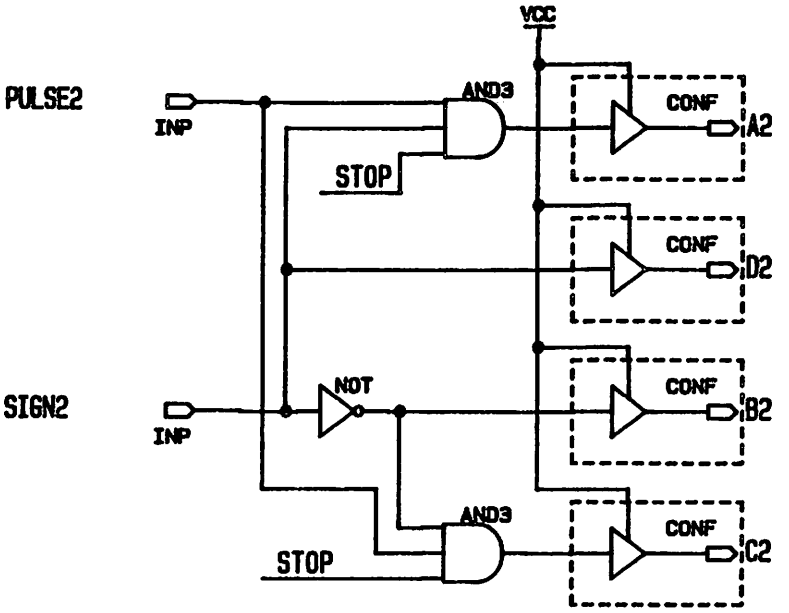

LL1LH1
INP
OR2
LM1
VCC
CONF
HCLMT1
MASKLM1
READJ1
CONF
DB00
LL2LH2
INP
OR2
LM2
VCC
CONF
HCLMT2
MASKLM2
READJ2
CONF
DB01
LL3LH3
INP
OR2
LM3
VCC
CONF
HCLMT3
MASKLM3
READJ3
CONF
DB02
LL4LH4
INP
OR2
LM4
VCC
CONF
HCLMT4
MASKLM4
READJ4
CONF
DB03

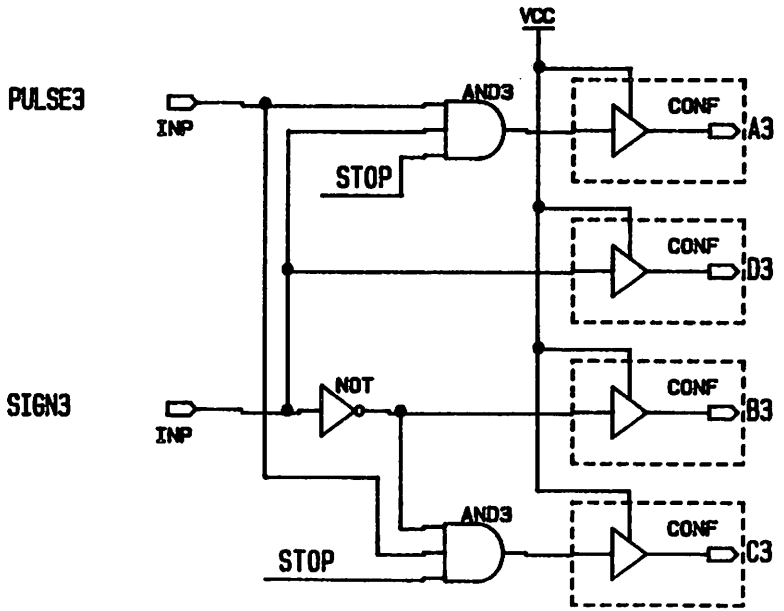

VCC
LL5LH5
INP
OR2
LM5
CONF
HCLMT5
MASKLM5
READJ5
CONF
DB04
VCC
LL6LH6
INP
OR2
LM6
CONF
HCLMT6
MASKLM6
READJ6
CONF
DB05
LM1
LM2
LM3
LM4
LM5
LM6
BOR6
VCC
CONF
LMTINTR

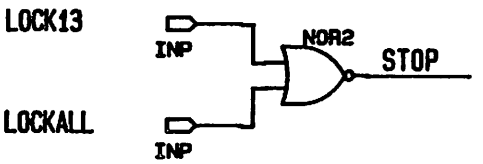


COMPANY	U C BERKELEY		
TITLE	SEQUENCER (J1)		
DESIGNER	GAUTAM DOSHI		
SIZE B	EPLD EP610D	NUMBER 1.00	REV A
DATE August 1988		SHEET 1 OF 1	
TURBO ON		SECURITY OFF	

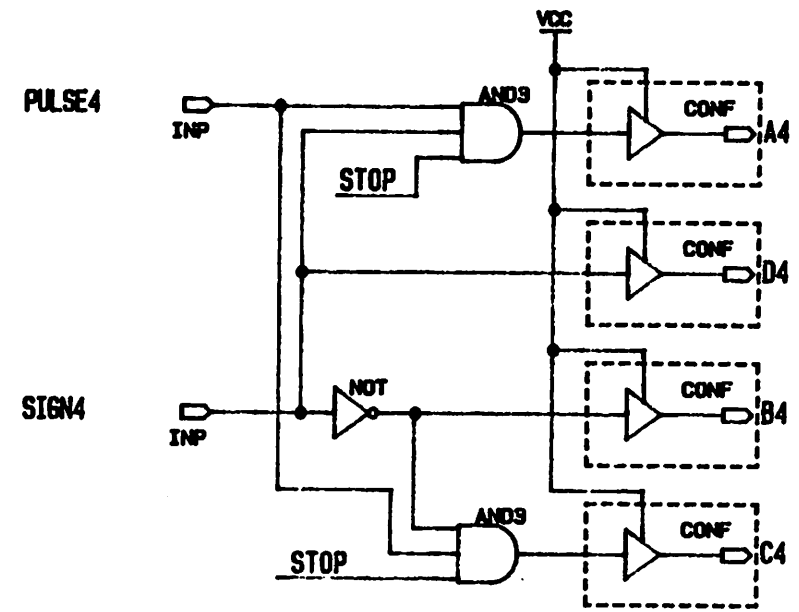


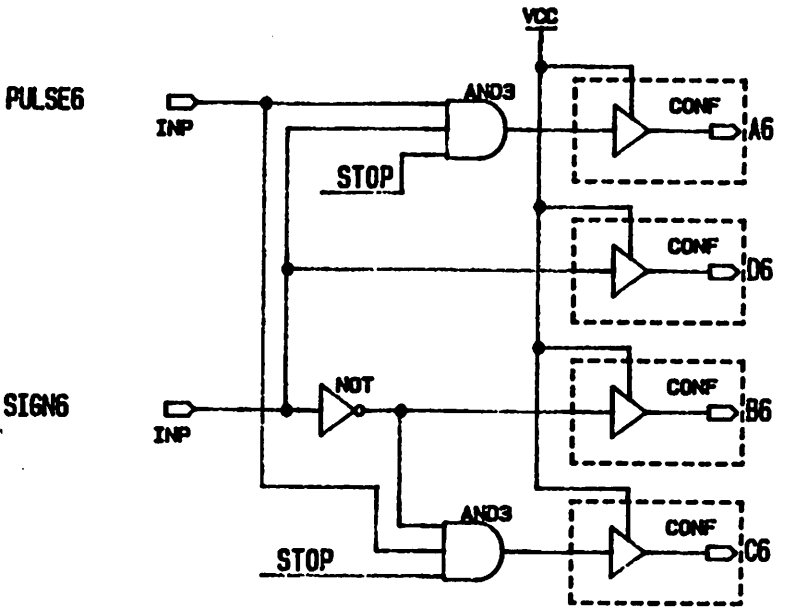


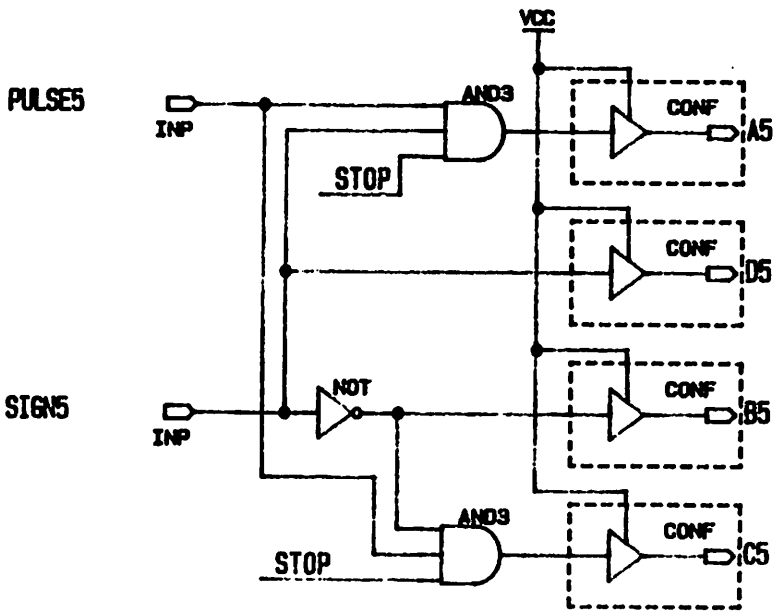


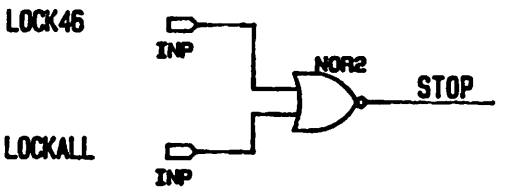


COMPANY	U C BERKELEY		
TITLE	PULSE N SIGN (J1-J3)		
DESIGNER	GAUTAM DOSHI		
SIZE B	EPLD EP600D	NUMBER 1.00	REV A
DATE	JULY 1988	SHEET 1 OF 1	
TURBO	ON	SECURITY	OFF

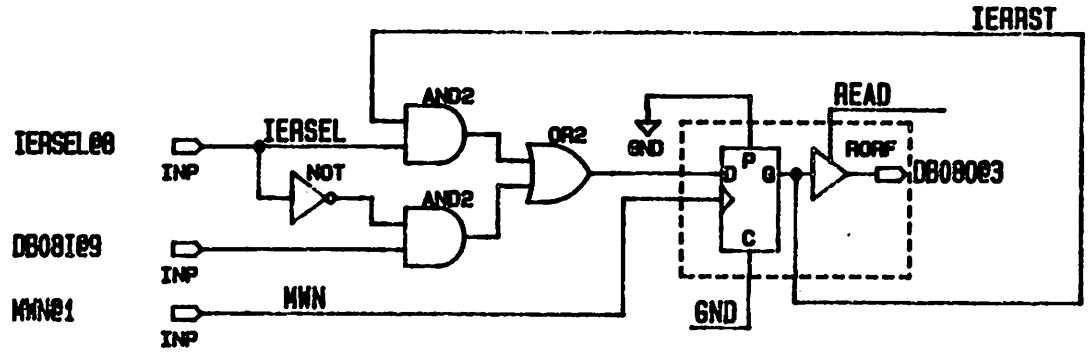


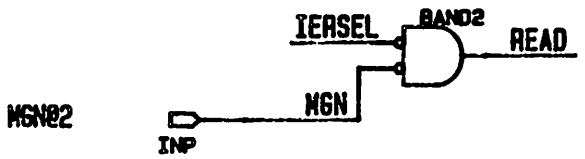


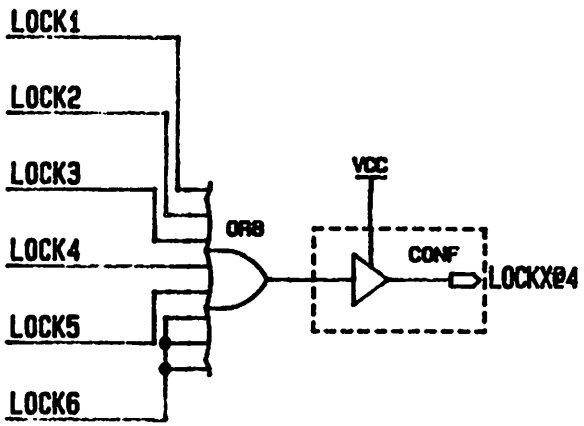




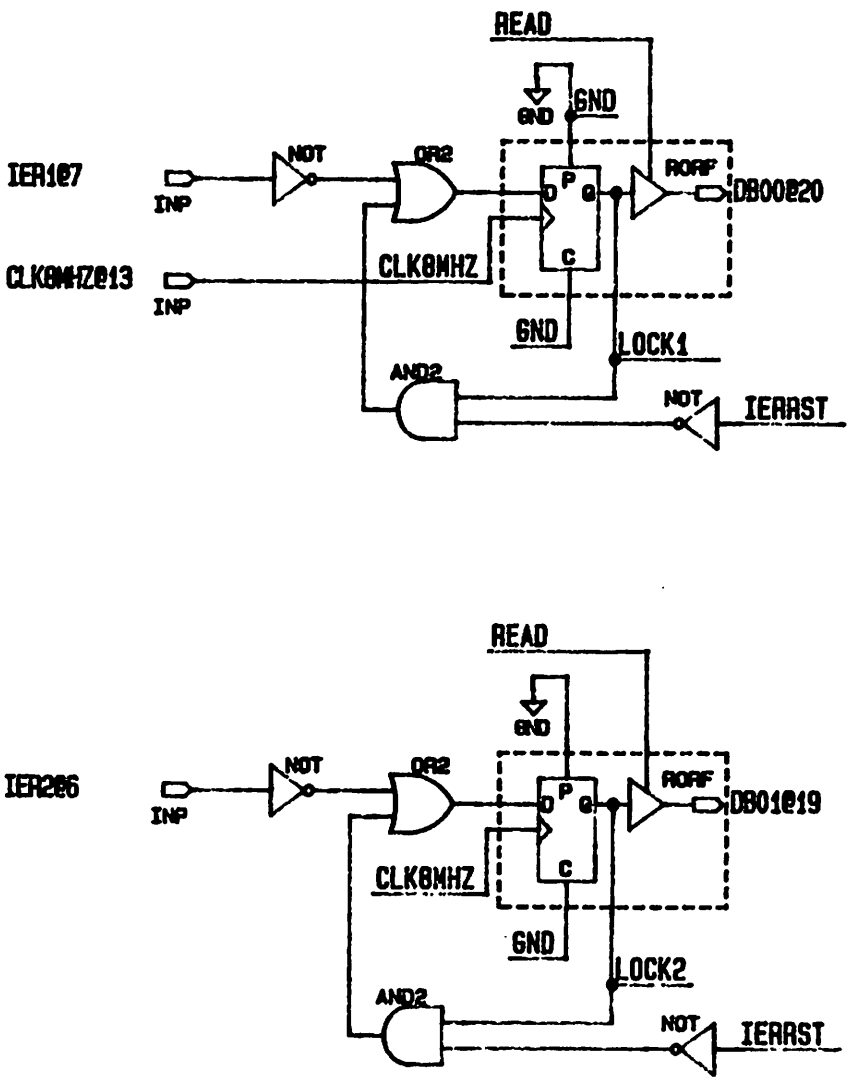
COMPANY	U C BERKELEY		
TITLE	PULSE N SIGN (J4-J6)		
DESIGNER	GAUTAM DOSHI		
SIZE B	EPLD EP600D	NUMBER 1.00	REV A
DATE JULY 1988		SHEET 1 OF 1	
TURBO ON		SECURITY OFF	

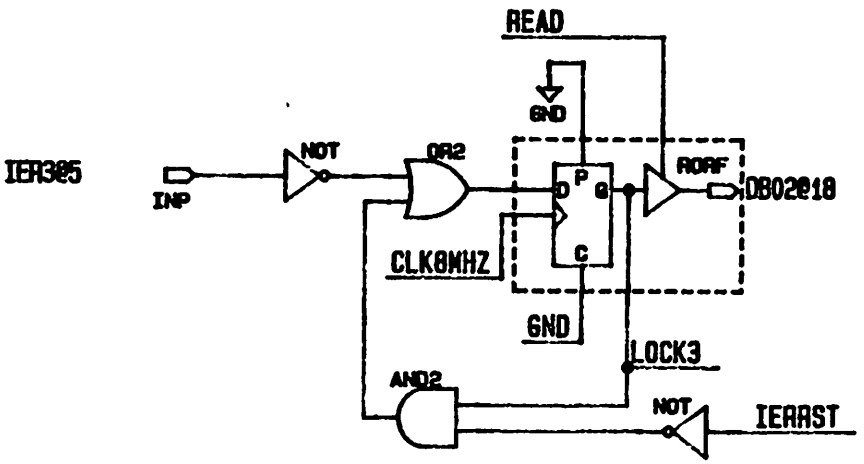




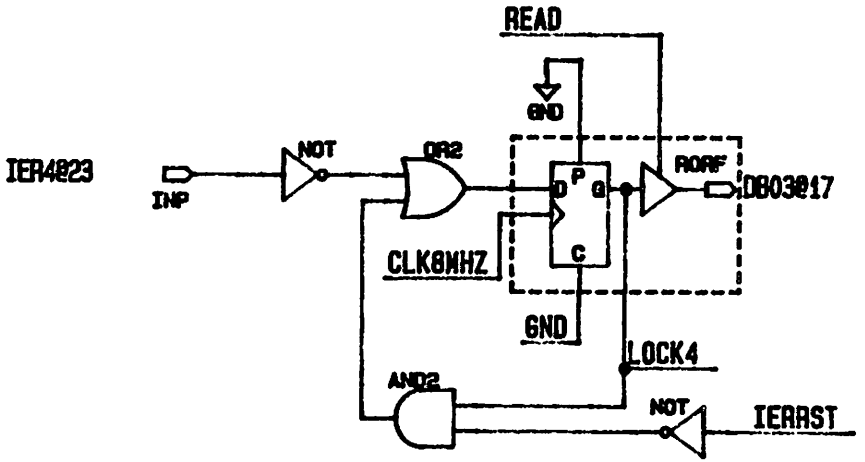


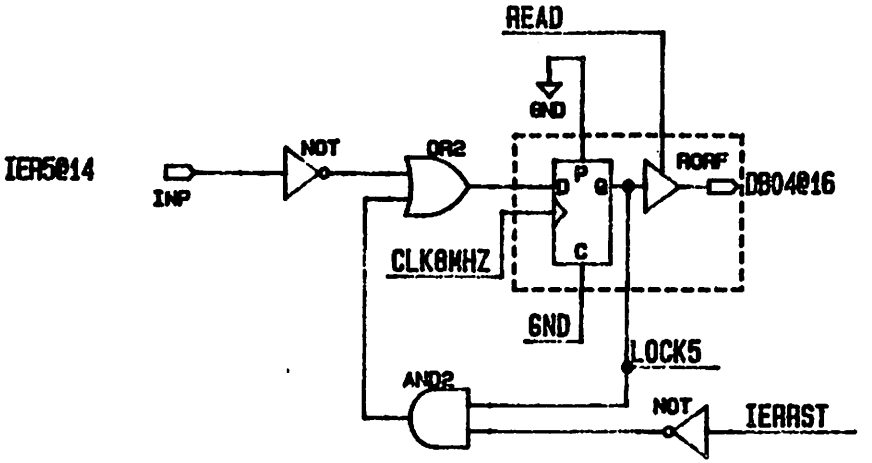
COMPANY	U C BERKELEY		
TITLE	CURRENT ERROR (J1-J6)		
DESIGNER	GAUTAM DOSHI		
SIZE B	EPLD EP610D	NUMBER 1.00	REV A
DATE June 1988		SHEET 1 OF 3	
TURBO ON		SECURITY OFF	

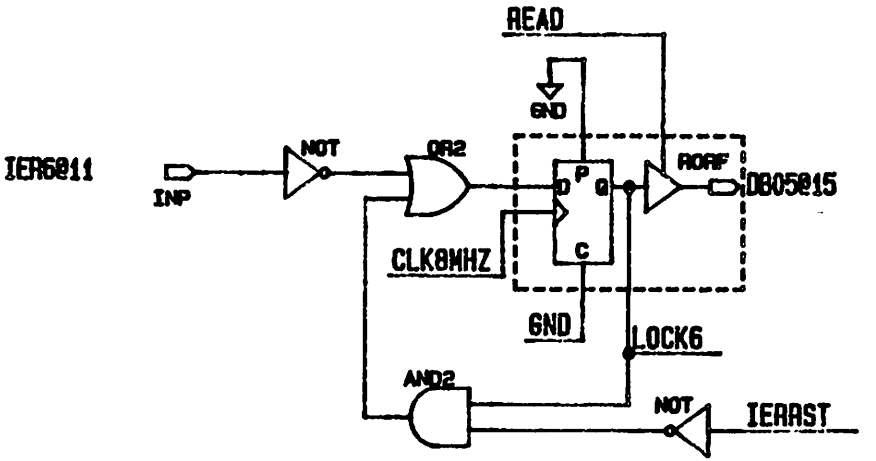

READ
GND
GND
IER107
NOT
OR2
INP
D
P
Q
RORF
DB00020
CLK8MHZ013
INP
CLK8MHZ
C
GND
LOCK1
AND2
NOT
IERRST

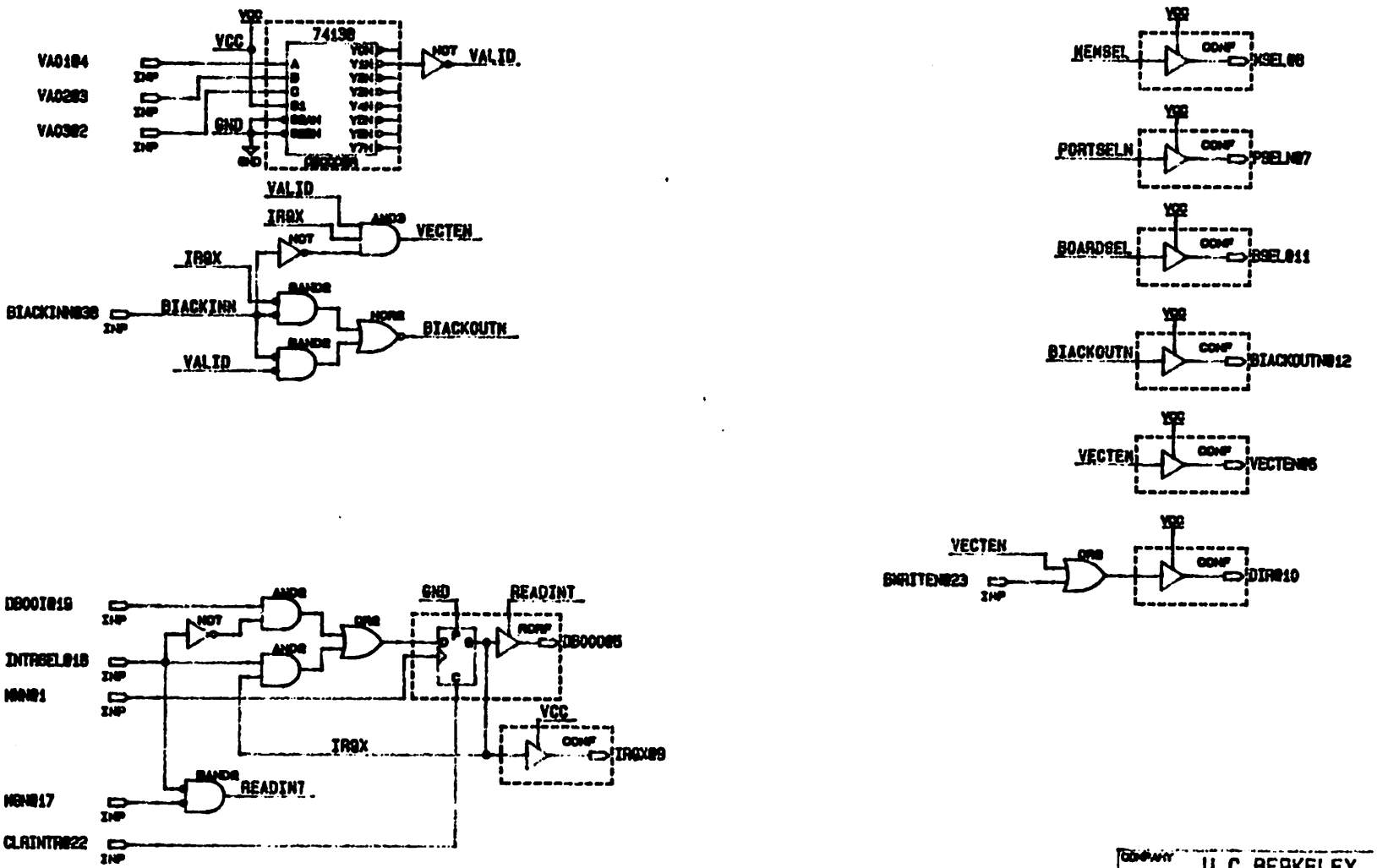

READ
GND
IER305
NOT
OR2
INP
D
P
Q
RORF
DB02018
CLK8MHZ
C
GND
LOCK3
AND2
NOT
IERRST

READ
GND
IER206
NOT
OR2
INP
D
P
Q
RORF
DB01019
CLK8MHZ
C
GND
LOCK2
AND2
NOT
IERRST

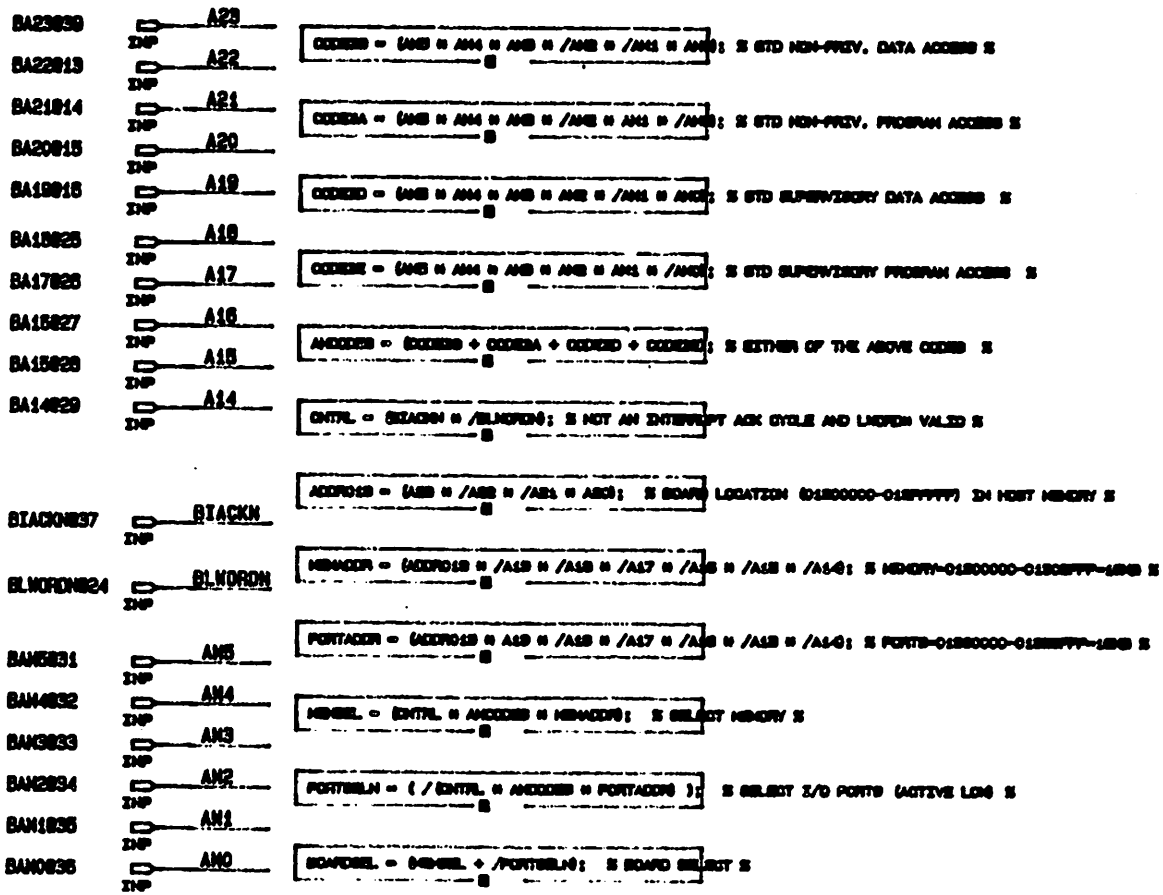

IER4@23
INP
NOT
OR2
READ
GND
D
P
Q
C
RORF
DB03@17
CLK8MHZ
GND
LOCK4
AND2
NOT
IERRST

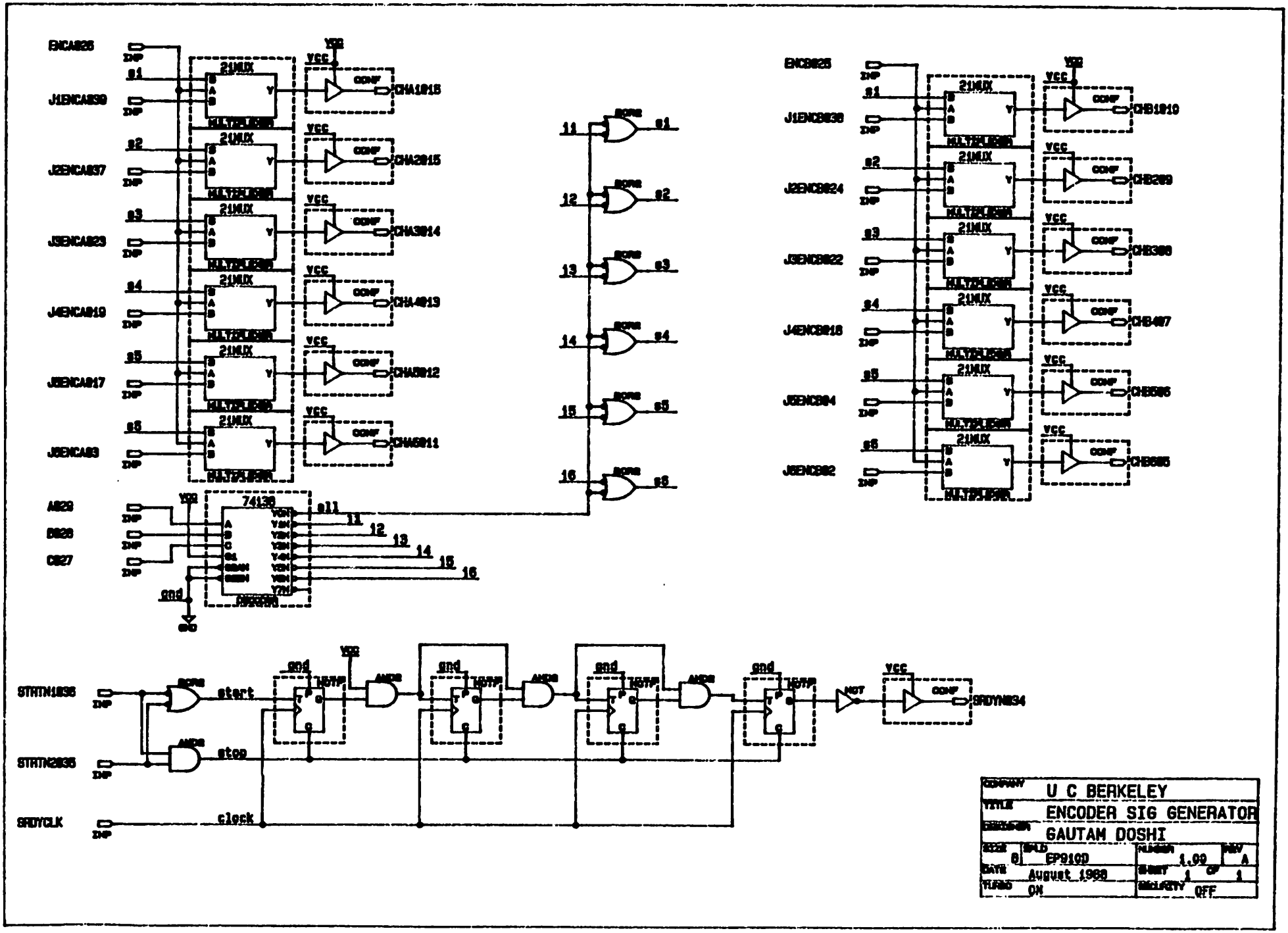

IER5@14
INP
NOT
OR2
READ
GND
D
P
Q
C
RORF
DB04@16
CLK8MHZ
GND
LOCK5
AND2
NOT
IERRST

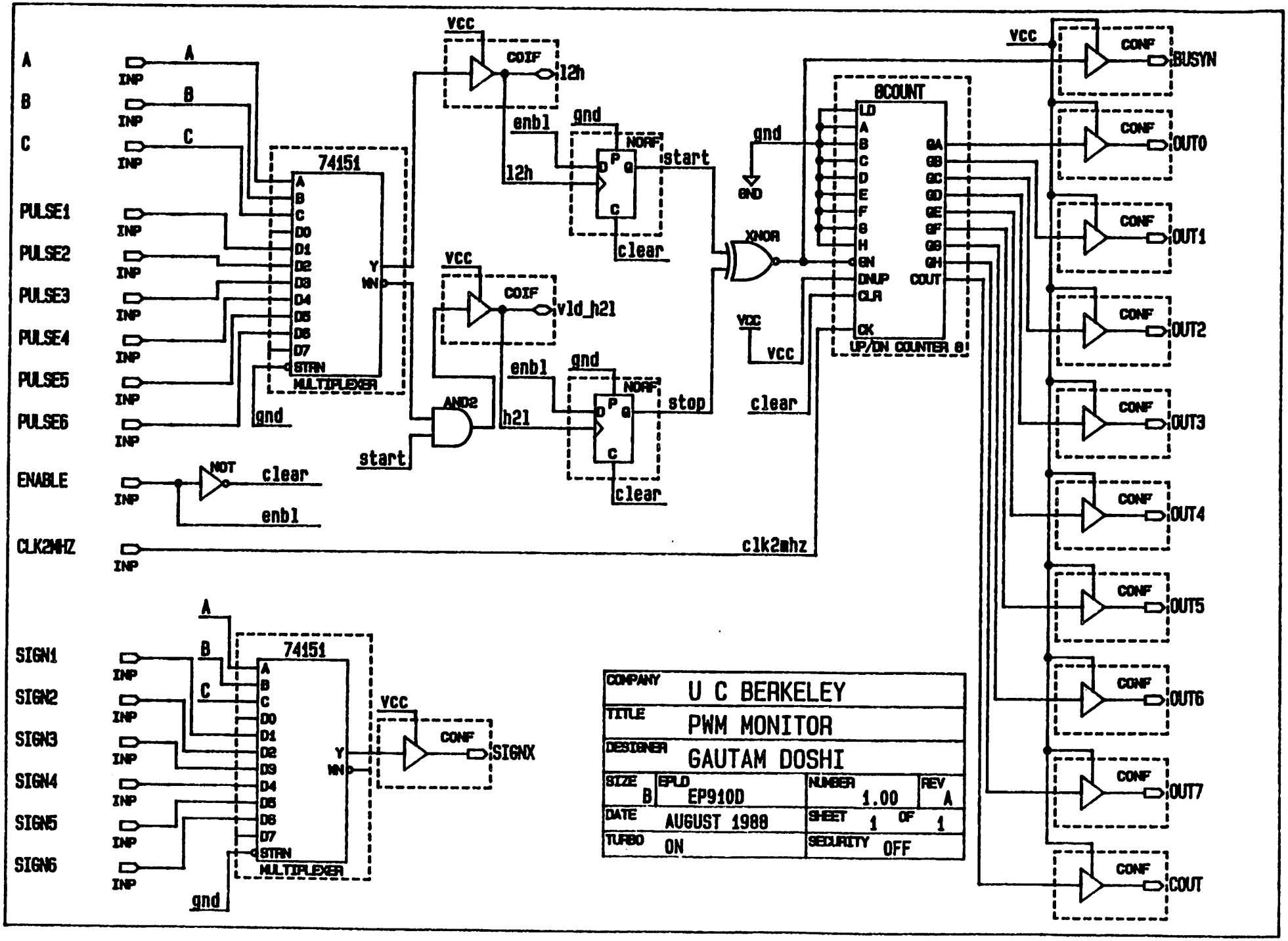

IER6@11
INP
NOT
OR2
READ
GND
D
P
Q
C
RORF
DB05@15
CLK8MHZ
GND
LOCK6
AND2
NOT
IERRST



COMPANY	U C BERKELEY
TITLE	VME INTERFACE
DRAWN BY	GAUTAM DOSHI
SIZE B	TYPE EP910
DATE	August 1988
TURBO	ON
NUMBER	1.00
REV	A
SHEET	1 OF 2
SECURITY	OFF

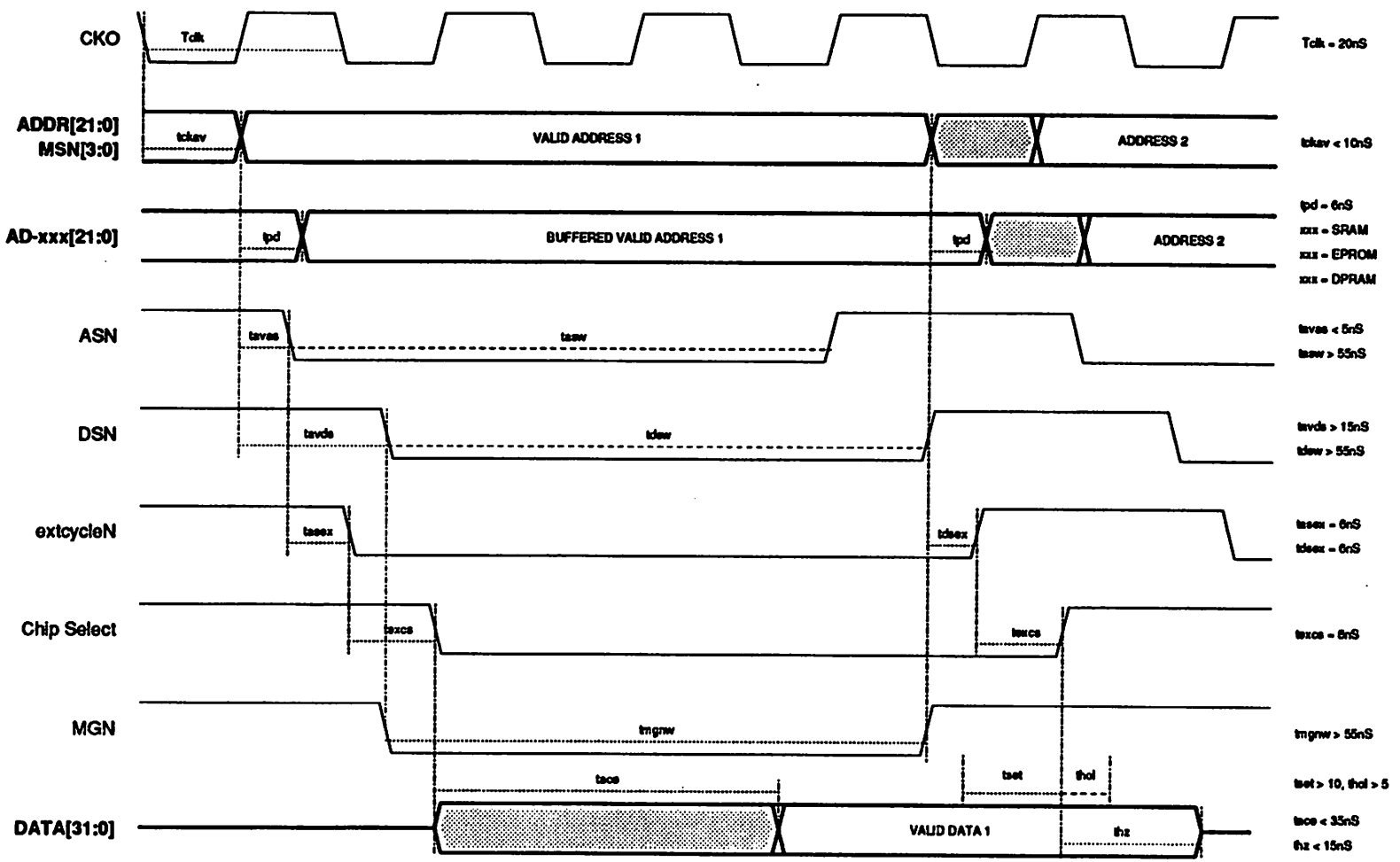

BA23823
A23
BA22813
A22
BA21814
A21
BA20815
A20
BA19816
A19
BA18825
A18
BA17826
A17
BA16827
A16
BA15828
A15
BA14829
A14
BIACKN837
BIACKN
BLWORDN824
BLWORDN
BAN5831
AN5
BAN4832
AN4
BAN3833
AN3
BAN2834
AN2
BAN1835
AN1
BAN0836
AN0
INP
ODDSEB = (AN5 # AN4 # AN3 # /AN2 # /AN1 # AN0); % STD NON-PRIV. DATA ACCESS %
ODDSEA = (AN5 # AN4 # AN3 # /AN2 # AN1 # /AN0); % STD NON-PRIV. PROGRAM ACCESS %
ODDSED = (AN5 # AN4 # AN3 # AN2 # /AN1 # AN0); % STD SUPERVISORY DATA ACCESS %
ODDSEC = (AN5 # AN4 # AN3 # AN2 # AN1 # /AN0); % STD SUPERVISORY PROGRAM ACCESS %
ANODDSE = ODDSEB # ODDSEA # ODDSED # ODDSEC; % EITHER OF THE ABOVE CODES %
CNTRL = BIACKN # /BLWORDN; % NOT AN INTERRUPT ACK CYCLE AND LWORDN VALID %
ADDR018 = (A18 # /A17 # /A16 # A15); % BOARD LOCATION (018000000-018FFFFFF) IN HOST MEMORY %
MEMACCR = (ADDR018 # A19 # /A18 # /A17 # /A16 # /A15 # /A14); % MEMORY=018000000-018CFFFF=1MB %
PORTACCR = (ADDR018 # A19 # /A18 # /A17 # /A16 # /A15 # /A14); % PORTS=018C00000-018CFFFF=1MB %
MEMSEL = CNTRL # ANODDSE # MEMACCR; % SELECT MEMORY %
PORTSELN = (/ CNTRL # ANODDSE # PORTACCR); % SELECT I/O PORTS (ACTIVE LOW) %
BOARDSEL = MEMSEL # /PORTSELN; % BOARD SELECT %


ENCA826
J1ENCA830
J2ENCA827
J3ENCA823
J4ENCA819
J5ENCA817
J6ENCA83
s1
s2
s3
s4
s5
s6
21MUX
VCC
CONF
CHA1815
CHA2815
CHA3814
CHA4813
CHA5812
CHA6811
11
12
13
14
15
16
A828
B828
C827
74138
gnd
g11
ENCB825
J1ENCB830
J2ENCB824
J3ENCB822
J4ENCB818
J5ENCB84
J6ENCB82
CHB1819
CHB269
CHB388
CHB487
CHB586
CHB685
STRTN1836
STRTN2835
SRDYCLK
start
stop
clock
AND8
NOT
SRDYNB34
U C BERKELEY
ENCODER SIG GENERATOR
GAUTAM DOSHI
EP910D
1.00
A
August 1988
ON
OFF


A
B
C
PULSE1
PULSE2
PULSE3
PULSE4
PULSE5
PULSE6
ENABLE
CLK2MHZ
INP
74151
MULTIPLEXER
gnd
NOT
clear
enbl
vcc
COIF
12h
enbl
NORF
start
clear
vld_h21
AND2
h21
stop
XNOR
8COUNT
LD
CLR
CK
CIN
CNT
CONF
BUSYN
OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
COUT
SIGN1
SIGN2
SIGN3
SIGN4
SIGN5
SIGN6
SIGNX
COMPANY U C BERKELEY
TITLE PWM MONITOR
DESIGNER GAUTAM DOSHI
SIZE B
EPLD EP910D
NUMBER 1.00
REV A
DATE AUGUST 1988
SHEET 1 OF 1
TURBO ON
SECURITY OFF

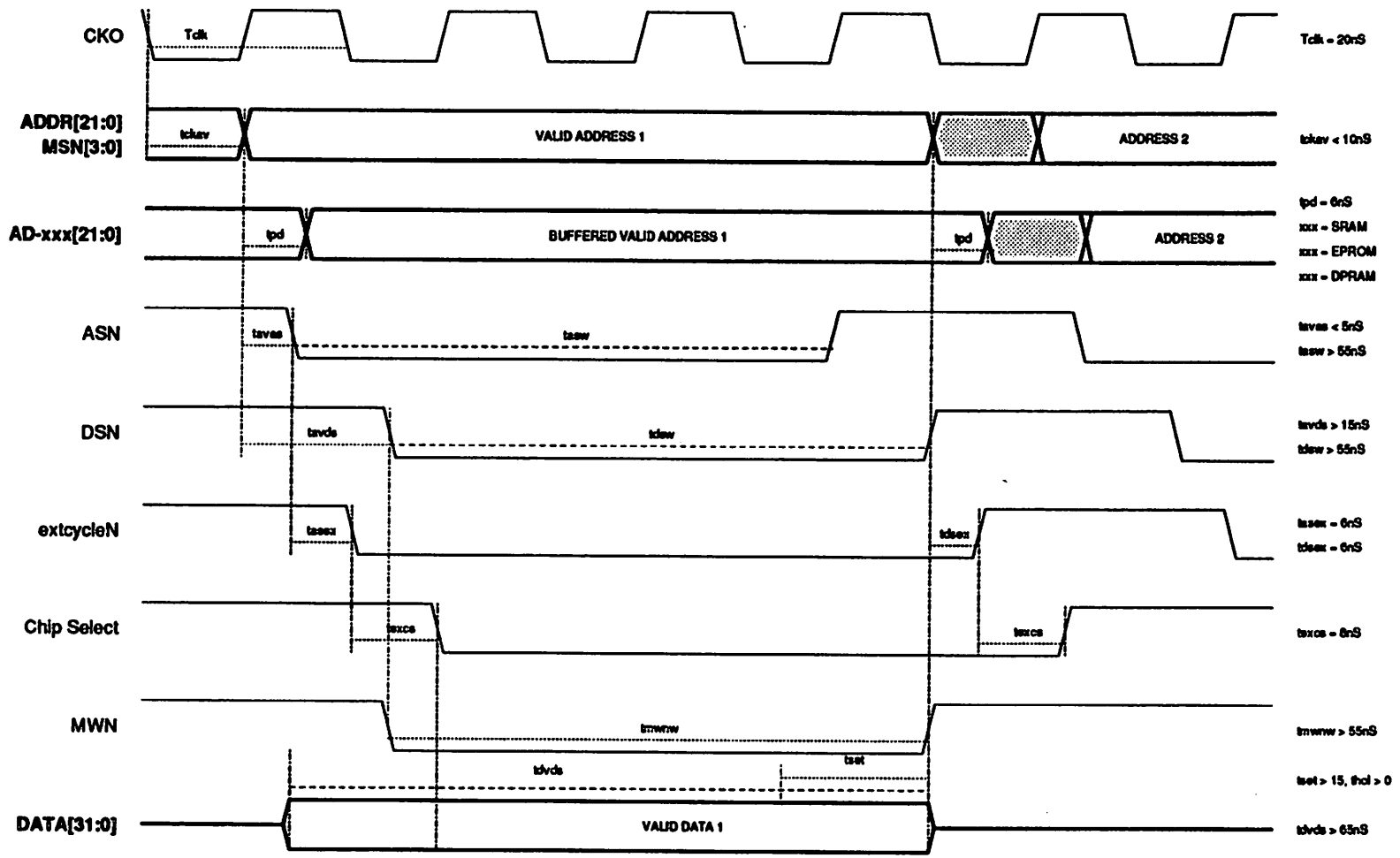
Appendix D

Timing Diagrams



EXTERNAL MEMORY READ TRANSACTION

DSP32C reading SRAM, EPROM or DPRAM - # of wait states = 2



EXTERNAL MEMORY WRITE TRANSACTION

DSP32C writing SRAM or DPRAM - # of wait states = 2

Appendix E

Parts List

PARTS LIST

Quantity	Component	Part Number	Description	Ref. Num.	Use
1	IxPls_Ogn13	EPLD910	EPLD910	U1	X
1	SN74ALS27	SN74AS27	Trip 3-in NOR	U2	X
10	74ALS646	SN74ALS646	Octal Bus Xcvr Reg	U3 - U12	
1	testenc	EPLD910	EPLD910	U13	X
2	74LS374	SN74AS374	Octal D type FF	U14 - U15	X
1	IxPls_Ogn46	EPLD910	EPLD910	U16	X
1	driver	MAX233	TTL-RS232 Driver	U17	X
1	duart	SCN2681	Dual Univ Asyn RT.	U18	X
5	SN74LS245	SN74AS245	Octal Bus Xsceiver	U19 - U23	
18	SN74ALS574	SN74AS574	Octal D-typeFF(3s)	U24 - U41	
32	JUMP_BG	--	Prototyping holes	U74 - U105	
1	7405	SN74AS05	Hex Inverter(oc)	U106	X
1	ml2208	ML2208 DAP	Data Acquis. Peri.	U107	X
1	74ALS04	SN74AS04	Hex Inverter	U108	X
2	74AS32	SN74AS32	Quad 2-in OR	U109 - U110	
1	Ierror16	EPLD610	EPLD610	U111	X
1	Sequencer4	EPLD610	EPLD610	U112	X
1	Sequencer5	EPLD610	EPLD610	U113	X
1	Sequencer6	EPLD610	EPLD610	U114	X
1	Sequencer1	EPLD610	EPLD610	U115	X
1	Sequencer2	EPLD610	EPLD610	U116	X
1	Sequencer3	EPLD610	EPLD610	U117	X
1	testpwm	EPLD910	EPLD910	U118	X
1	IC555	IC555	Timer chip	U119	X
1	VMEpld	EPLD910	EPLD910	U120	X
1	74ACT244	SN74ACT244	3-s Octal Buffer	U121	X
6	ULSTL082*1	TL082ACP	Dual OPAMP	U122 - U127	X
14	JUMP_1	--	Jumper	U128 - U141	X
1	74161	SN74LS161	Sync. 4bit counter	U142	X
8	74LS377	SN74LS377		U143 - U150	X
13	74ALS541	SN74ALS541	3-s Octal Buffer	U151 - U163	X
4	74LS138	SN74AS138	3 to 8 Decoder	U164 - U167	X
1	74LS08	SN74AS08	Quad 2-in AND	U168	XX00
3	fastPAL	PAL16L8-7CN	16-I/O, 8-O PAL	U169 - U171	
1	74S38	SN74S38	Quad 2-in NAND(oc)	U172	XX00
1	74AS02	SN74AS02	Quad 2-in NOR	U173	X

8	DPRAM_2Kx8	VT7132A-30PC	2Kx8 Dual Port RAM	U174 - U181	
1	JUMP_2x7	--	Jumper	U182	X
3	74AS74	SN74AS74	Dual D-type FF	U183 - U185	X
2	delayline		Digital Delay Line	U186 - U187	
1	74LS244	SN74AS244	3-S Octal Buffer	U188	X
4	td62501p	TD62501P	CE Xsistor Array	U189 - U192	
1	JUMP_2x6	--	Jumper	U193	X
1	DSP32C	WE-DSP32C R08	Digital Signal Proc	U194	X
8	EPROM_2Kx8	TMS27C292-3JL	2Kx8 Erasable PROM	U195 - U202	
8	SRAM_8Kx8	CY7C185-35PC	8Kx8 Static RAM	U203 - U210	
1	VREG_7905		Voltage Regulator	U211	X
2	74ALS00	SN74AS00	Quad 2-in NAND	U212 - U213	X
1	U74ALS08	SN74AS08	Quad 2-in AND	U214	XXXO
2	74ALS32	SN74AS32	Quad 2-in OR	U215 - U216	XXOO
1	74ALS10	SN74AS10	Trip 3-in NAND	U217	XXO
5	74LS32	SN74AS32	Quad 2-in OR	U218 - U222	XXOO
3	74AS138	SN74AS138	3 to 8 Decoder	U223 - U225	X
1	PulseNsign46	EPLD600/610	Prog. Logic Dev.	U226	X
1	PulseNsign13	EPLD600/610	Prog. Logic Dev.	U227	X
1	limitpld	EPLD910	Prog. Logic Dev.	U228	X
6	HCTL1000	HCTL-1000	Motion Cntl IC	U229 - U234	
1	C^1UDFTA		0.1 uF cap.	C1	X
1	C^47UDFTA		0.47 uF	C2	X
18	C1NDHCA		1 nF cap	C3 - C20	
25	C1^5NCHCA		1.5 nF Cap.	C21 - C45	
6	BYCAP47		47 uF LF-Bypass	C46 - C51	
1	CAPCER2u		2 uF Ceramic Cap	C52	X
1	CAPCER1u		1 uF Ceramic Cap	C53	X
176	BYCAP^1		0.1 uF HF-Bypass	C54 - C229	
12	POT_100K	3266W-1-104	100 Kohm Pot.	R1 - R12	
2	R1^0MJFDA		1 Mohm res	R13 - R14	
4	RNetDip15bus		10K Resistor Pack	R15 - R18	
6	R100KJFDA		100Kohm res.	R19 - R24	
6	R100_JFDA		100ohm res(5%)	R25 - R30	
12	R33KJFDA		33 Kohm res(5%)	R31 - R42	
24	R4^7KJFDA		4.7 Kohm res(5%)	R43 - R66	
25	R2^4KJFDA		2.4 Kohm res(5%)	R67 - R91	
1	duart_conn	501-1007ES T&B	10 pin connector	P2	X
1	pwm2_conn	501-3407ES T&B	34 pin connector	P3	X
1	power_conn		2 pin connector	P4	X
1	scb_conn	501-3407ES T&B	34 pin connector	P5	X

1	rob_conn	501-3407ES T&B	34 pin connector	P6	X
1	dsp_conn2		60 pin connector	P7	X
1	dsp_conn1		60 pin connector	P8	X
1	pwm1_conn	501-5007ES T&B	50 pin connector	P9	X
3	P096P0000001		96 pin connector	P10 - P12	
1	SWITCH1		Push Button Switch	S1	X
1	40MHzOSC	NCT070BSaronix	X-stal Clk Oscill.	Y1	X
x	xxx	SMO-xx-S6G T&B	xx pin sockets	xx	x
