

REVISIONS

REV	DESCRIPTION	DATE	APPROVED
0	Initial Release	March 1, 1983	<i>[Signature]</i>
A	Redefine VUA, VMA in Section 1.1	Aug. 10, 1983	<i>[Signature]</i>

This specification is preliminary. It is intended for use by CMS personnel during product development. Information provided within is made available to other organizations strictly for the purpose of preliminary evaluation and should not be used for design purposes. Information in this document is subject to change without notice.

CONTRACT NO.



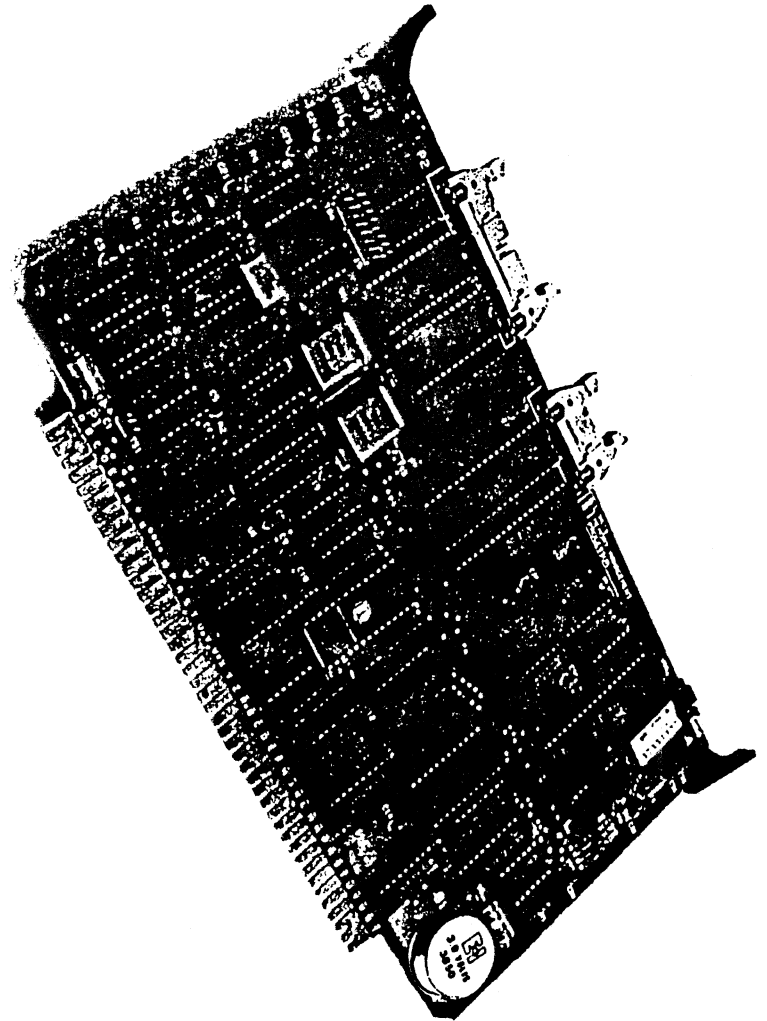
9639 MEMORY MANAGEMENT PROCESSOR
Performance Specifications

APPROVALS	DATE
PREPARED <i>[Signature]</i>	<i>[Date]</i>
CHECKED	
RELEASED	

SIZE
A

DWG. NO.

REV.



3277 FORTUNE AVENUE • LOS ANGELES, CALIFORNIA 90004 U.S.A.

1.0 GENERAL INFORMATION

The 9639 is an EXORbus compatible processor module specifically designed for use with the OS-9 Level 2 operating system. It provides memory management hardware with dynamic task allocation. The module also provides hardware to perform fast task switching with minimal load on the system software. Memory to memory transfers can be accommodated within a task or between tasks by Direct Memory Access. The 9639 is bus compatible with all EXORbus support modules with the exception that memory modules require four additional address inputs. The block diagram is illustrated in Figure 1.

1.1 I/O CHANNEL SUPPORT

The 9639 is generally compatible with all conventional I/O expansion modules capable of 2 MHz operation. It does not respond to the Memory Ready signal (bus pin R), however, and therefore cannot be used with modules that exercise that line. All I/O modules in the system must be adjusted to honor the VUA line on bus pin 10. This is not the case with memory modules, however, which should be set to respond to VMA on bus pin F.

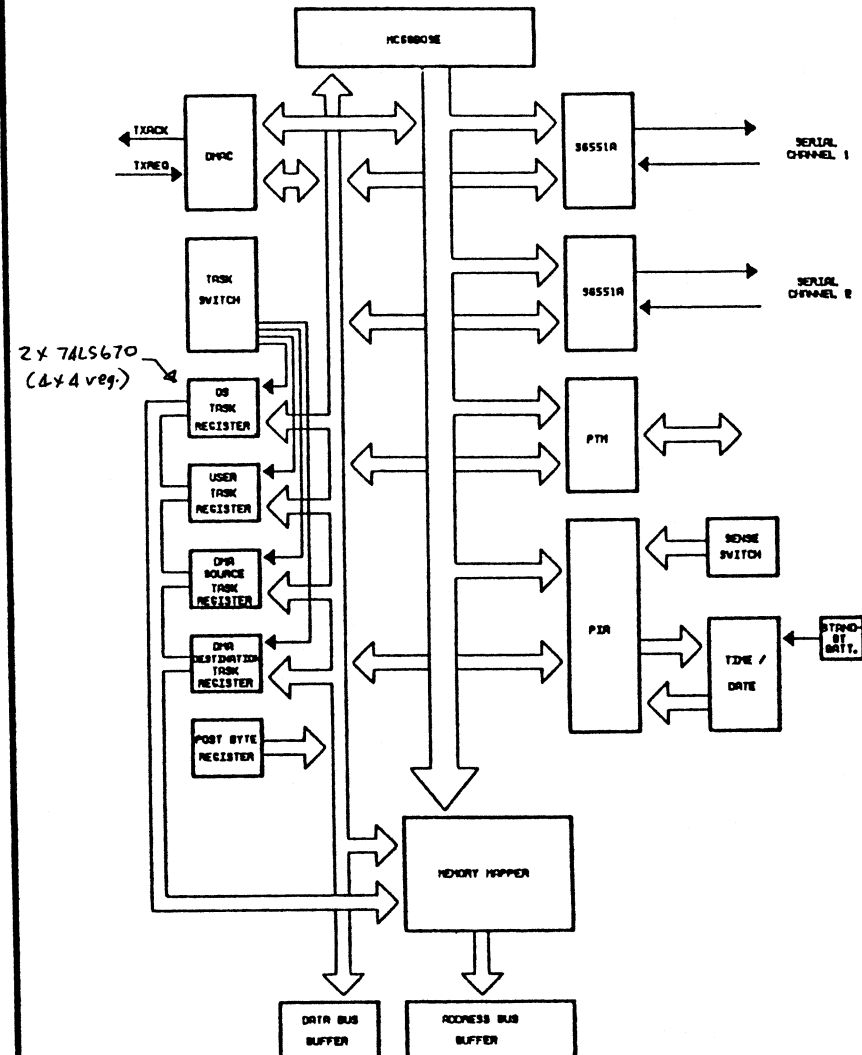
The 9639 provides two serial channels, a time/date subsystem, and a real time interrupt PTM on board. These devices are accessible only during execution of the OS task. Tasks other than the OS task cannot access any I/O devices in the system.

Disc data transfers can be performed by Programmed I/O or by Direct Memory Access. The DMA support is provided by a controller onboard the 9639. If DMA is used the disc controller must be chosen to support the DMA handshake protocol. Only one channel of DMA can be active at any given time.

Memory to memory transfers are also accommodated by the DMA control section of the 9639. Transfers are initiated by the OS task and are effective between blocks in the same or in different tasks.

1.2 LOCAL MEMORY

The 9639 supports either 4K or 8K of EPROM in a single socket. This memory is accessed only during execution of the OS task and appears at the top of the map. It contains the restart and interrupt vectors in the normally assigned locations. Local I/O devices overlay this EPROM from \$FFA0 through \$FFBF. The individual device addresses are shown in Figure 2. External memory at these addresses, if present on the bus, is superseded by the EPROM or local I/O section. The 9639 contains no RAM.



9639 BLOCK DIAGRAM

FIGURE 1

EPROM VECTORS	FFF0
EPROM	FFE0
RESERVED	FFD7
DMAC DATA CHAIN REGISTER	FFD6
DMAC INTERRUPT CONTROL REGISTER	FFD5
DMAC PRIORITY CONTROL REGISTER	FFD4
RESERVED	FFD2
DMAC DESTINATION CHANNEL CONTROL REGISTER	FFD1
DMAC SOURCE CHANNEL CONTROL REGISTER	FFD0
RESERVED	FFC8
DMAC DESTINATION BYTE COUNT REGISTER	FFC6
DMAC DESTINATION ADDRESS REGISTER	FFC4
DMAC SOURCE BYTE COUNT REGISTER	FFC2
DMAC SOURCE ADDRESS REGISTER	FFC0
RESERVED	FFBD
CONTROL SWITCHES	FFBC
USER TASK REGISTER	FFBB
OS TASK REGISTER	FFBA
DMA DESTINATION TASK REGISTER	FFB9
DMA SOURCE TASK REGISTER	FFB8
MC68B40 PTM	FFB0
S6551A ACIA #1	FFAC
S6551A ACIA #2	FFA8
TIME/DATE AND SENSE SWITCH PIA	FFA4
POSTBYTE REGISTER	FFA0
RESERVED EXTERNAL I/O SPACE	FF60
EPROM	F800
EPROM AND MAPPING RAM (SEE TABLE 1)	F000
OPTIONAL EPROM OR EXTERNAL RAM	E000
EXTERNAL RAM	0000

FIGURE 2: 9639 MEMORY MAP
(OS TASK ONLY)



CREATIVE MICRO SYSTEMS
3822 CERRITOS AVENUE • LOS ANGELES, CALIFORNIA 90008 U.S.A.

The MMU has enough capacity that the "DAT image" ^{never} has to be reloaded into the MMU when switching tasks !! Less overhead.
(128 maps of 16 x 4K blocks)

1.3 MEMORY MAPPING CIRCUIT

Memory mapping is accomplished through the use of a translation table written into a mapping RAM. The mapping RAM overlays the EPROM from \$F000 through \$F7FF. Write accesses to these addresses are directed to the RAM and read accesses are directed to the EPROM. During translation cycles the mapping RAM is presented with an eleven bit value consisting of address lines 12 through 15 from the processor and seven bits from the currently active task register. This value is translated into the eight high order address lines (12 through 19) for a 4K block of physical memory. These lines are used with the address lines 0 through 11 from the processor to form the 20 bit address required to access a location in the maximum 1.045 megabyte range. The structure of the mapping RAM is illustrated in Table 1.

1.4 TASK REGISTERS

The 9639 contains four task registers that appear as write-only locations in the local I/O address range. Each one of these registers is pertinent to a particular type of task designation. These are the USER task (register 3), the OS task (register 2) the destination task for DMA transfers (register 1) and the source task for DMA transfers (register 0). The operating system must have the task number (from 0 through 7F) placed in the appropriate register for any anticipated type of task prior to allowing a switch to the task. The path for task switching is prepared by the operating system, however the switch is actually performed by the task switch hardware.

(DMA done thru MMU very efficiently!)

1.5 TASK SWITCH SUPPORT

Task switching can be initiated by hardware interrupts, software interrupts, returns from interrupts (in OS task only), or by the onset of DMA activity. A special case handled by the task switch hardware involves SWI2. This instruction is always treated as a system call with a single postbyte. Like any other interrupt it causes a switch to the OS task, but in the stacking process the postbyte is placed in a register accessible to the OS task. The PC return address is incremented by one before it is stacked. This allows a structured return from the OS to avoid the postbyte after completion of the system call. (less S/w overhead)

how BIT map?
(cf U20, U25 last page)



CREATIVE MICRO SYSTEMS
3822 CERRITOS AVENUE • LOS ANGELES, CALIFORNIA 90008 U.S.A.

When executing number task number-- range--	The value in mapping RAM address--	Is the physical block for the task address
7F	F7FF	F000-FFFF
7F	F7FE	E000-EFFF
7F	F7FD	D000-DFFF
7F	F7FC	C000-CFFF
7F	F7FB	B000-BFFF
7F	F7FA	A000-AFFF
7F	F7F9	9000-9FFF
7F	F7F8	8000-8FFF
7F	F7F7	7000-7FFF
7F	F7F6	6000-6FFF
7F	F7F5	5000-5FFF
7F	F7F4	4000-4FFF
7F	F7F3	3000-3FFF
7F	F7F2	2000-2FFF
7F	F7F1	1000-1FFF
7F	F7F0	0000-0FFF
7E	F7EF	F000-FFFF
7E	F7EE	E000-EFFF
7E	F7ED	D000-DFFF
01	F012	2000-2FFF
01	F011	1000-1FFF
01	F010	0000-0FFF
00	F00F	F000-FFFF
00	F00E	E000-EFFF
00	F00D	D000-DFFF
00	F00C	C000-CFFF
00	F00B	B000-BFFF
00	F00A	A000-AFFF
00	F009	9000-9FFF
00	F008	8000-8FFF
00	F007	7000-7FFF
00	F006	6000-6FFF
00	F005	5000-5FFF
00	F004	4000-4FFF
00	F003	3000-3FFF
00	F002	2000-2FFF
00	F001	1000-1FFF
00	F000	0000-0FFF

TABLE 1: MAPPING RAM ORGANIZATION

CMS CREATIVE MICRO SYSTEMS
3877 CENTINOS AVENUE • LOS ANGELES, CALIFORNIA 90018 • U.S.A.

Task switching is accomplished by hardware selection of the appropriate task register. Prior to the selection the OS must have written into the register the number of the task that should execute after switching. Entry to user tasks are controlled by OS so there will always be an opportunity to install the desired number before the RTI that initiates the switch. Entry to the OS task can occur because of a software interrupt or a hardware interrupt. The OS task register should, of course, always have the OS task number in it, although it is permissible for OS to gain access to user task maps by installing another task number in the OS task register.

The task registers allocated for DMA transfers become effective during the DMA cycles and are always entered from OS. They can be written prior to initiation of the transfer operation. Task register 0 is reserved for the source task, involving memory reads, and register 1 is reserved for the destination task, involving memory writes. Transfers between memory and a peripheral will use only one task register. Memory to memory transfers will use both registers.

2.0 RECOMMENDATIONS FOR OS INSTALLATION

The 9639 has been designed to minimize the effort necessary to prepare an operating system for installation. Extensive hardware support for activities normally done by software has reduced the burden for the systems programmer.

2.1 SYSTEM INITIALIZATION

Following a RESET, the 9639 will be defaulted to the OS task. The restart vectors will be fetched from the EPROM and must point to the initialization routine. Since the hardware has selected the OS task, the processor will have access to the peripherals and the mapping RAM. The first initialization event must be to initialize the mapping RAM. This can be combined with the memory sizing process by the executing the following three step procedure.

Set up the mapping RAM for 32 tasks with 32K allocated for each task.

Reference to Table 1 illustrates that each location of the mapping RAM (Write Addresses F000 through F7FF) is intended to contain the physical number of a 4K block allocated for a particular address range of a specified task. Write Address locations F000 through F007, for example, determine which physical blocks will be used to comprise the lower 32K of memory for task 0. Locations F010 through F017 determine the blocks for the lower 32K of task 1, etc. Each 16 byte section of the mapping RAM, therefore, performs the translation for the entire 64K range that can be allocated to a particular task, and there

CMS CREATIVE MICRO SYSTEMS
3877 CENTINOS AVENUE • LOS ANGELES, CALIFORNIA 90018 • U.S.A.

are 128 such sections in the mapping RAM. Maps can be defined for up to 128 tasks if sufficient memory is available. In such a case the average task allocation would be 8K (2 blocks).

The portion of the memory initialization procedure that sets up the mapping RAM involves writing an ascending 8-bit value in the lower half of each 16 byte section in the range F000 through F1FF. The result of this operation is to set the mapping RAM so that the 32 tasks, each with 32K allocated, sequentially occupy all of the physical blocks of the maximum memory range.

Assign physical memory block 0 as a common block to appear in each of the 32 tasks at address 8000. Use this as temporary scratch RAM during the memory sizing search. Remember that this block will also appear at address 0000 of task 0.

This can be accomplished by writing the value 0 at F008, F018, F028 --- F1F8. A 4K block of memory will now be available at address 8000 regardless of which task number from 0 through 1F is placed in the OS task register. Since this same physical block was also allocated to task 0 at address 0000, the memory size search must be designed to prevent corruption of the scratch block.

Perform the memory size search on tasks 0 through 1F, searching the address range 0000 through 7FFF for each task. Select the task by writing the number in the OS task register.

At this point the memory size is known and task 0 has 36K allocated. Assuming that 0 will be the normal task number designating the OS task, the procedure should now allocate the desired amount of memory to the OS. All unassigned tasks should have their allocation section of the mapping RAM set to translate to a safety zone of physical memory, block FF for example.

2.2 ACTIVATION OF A NEW TASK

Inspection of Table 1 reveals a relationship between a task number and the section of the mapping RAM that performs address translation for that task. Bits 0 through 3 of the Write Address represent the logical block number of the task map translated by that location of the map RAM. Bits 4 through 11 of the Write Address are identical to the task number. The following rule is therefore derived.

A concatenation of the desired task number and the desired logical block number becomes the displacement above F000 of the mapping RAM location that must have the desired physical block number written to it during the allocation process.



CREATIVE MICRO SYSTEMS
3222 CERRITOS AVENUE • LOS ALAMITOS, CALIFORNIA 92644 U.S.A.

Activation of a new task is a three step process. First an available task number must be selected and the corresponding section of the mapping RAM must be loaded with the block numbers of free physical blocks of memory. If the task requires less than the full map of 16 blocks, the remaining logical blocks must be mapped to a safety zone, such as physical block FF.

After the allocation has been done, the new task memory must be loaded with executable code. This would normally be done by loading the OS buffer with blocks of code and then utilizing the memory-to-memory DMA feature to transfer the blocks to the new task. Another technique would be to map the recipient block of new task memory into an unused logical block of OS task memory, load the code, and remove the block from the OS task. Still another technique could be implemented if the loader and disc drivers and descriptors were located in the the onboard EPROM. In such an instance, 56K of the new task memory (all but the top 8K) could be mapped into OS by writing the task number in the OS task register. This would allow loading directly into the task memory which could then be removed from the OS task by rewriting the OS task number back in the OS task register. The code for executing this must of course reside entirely in the EPROM common block.

The third phase of new task activation is initiated by a FORK or CHAIN system call. At this time the OS must select the stack region for the new task and preload it in preparation for entry to the task. This can be accomplished by any of the methods suggested above for loading the executable module. The entry to the task occurs after the RTI is executed by the OS. Prior to executing the RTI the OS must set the stack pointer and the user control switch (bit 2 of location FFBC).

The user control switch can only be accessed by the OS task. Its purpose is to allow RTI opcodes to trigger a task switch to the USER task. In the case where a system call was generated by the OS the switch must be off prior to the RTI in order to prevent a task switch. After a RESET the user switch is off. Bits 0 and 1 of the control switch byte are used to start DMA cycles. These bits must not be inadvertently set while working with the user switch bit.

(cf. task switching schematics note)

2.3 RESPONSE TO HARDWARE INTERRUPTS

If a hardware interrupt (IRQ) occurs during execution of a user task, the machine state will be stacked prior to the task switch. The vector fetch will occur after the switch to the OS task. The service routine must save the stack pointer for the subsequent return. The user switch must be set to allow the return to switch to the user task. Of course the system could be set to return to a different task if desired.



CREATIVE MICRO SYSTEMS
3222 CERRITOS AVENUE • LOS ALAMITOS, CALIFORNIA 92644 U.S.A.

If the hardware interrupt occurs during execution of the OS task no task switch will occur. Prior to the return from the interrupt, the user switch must be cleared or the switch hardware will attempt to switch to the user task.

2.5 RESPONSE TO SWI AND SWI3

Interrupts generated by SWI and SWI3 are treated in the same fashion as hardware interrupts. It is important to remember the use of the user switch prior to returning from any interrupt.

2.6 RESPONSE TO SWI2

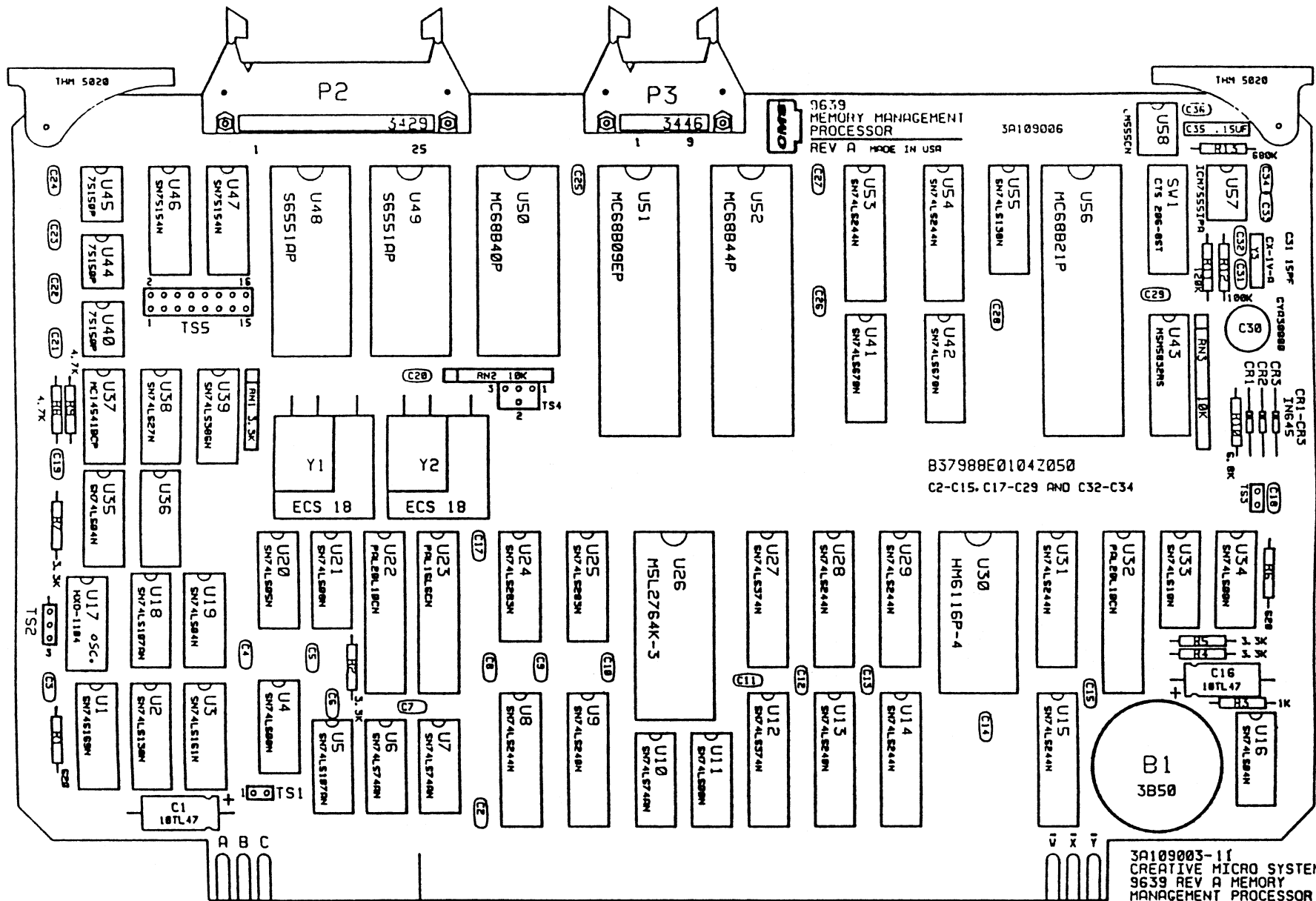
The SWI2 interrupt generates a unique response in the 9639. It stacks the processor registers in normal fashion except for the Program Counter. This register is incremented by one before being pushed on the stack. This allows a subsequent return to skip over the postbyte, assumed to follow every SWI2 opcode, and eliminates the need for manipulation of the stack to effect a structured return. Since a user task stack is not easily accessible to the OS task, the postbyte is placed in a register that can be directly read by the OS.

During the cycle in which the postbyte is latched, a hardware interrupt mask is set to prevent an interrupt from occurring during the first instruction cycle of the new task. At this time the OS task is in effect but the stack pointer is in an undefined condition. The first operation of the OS task should be to establish the stack. The hardware interrupt mask can then be cleared by clearing bit 3 of the control switch register.



CREATIVE MICRO SYSTEMS

3827 CERRITOS AVENUE • LOS ALAMITOS, CALIFORNIA 90730 U.S.A.



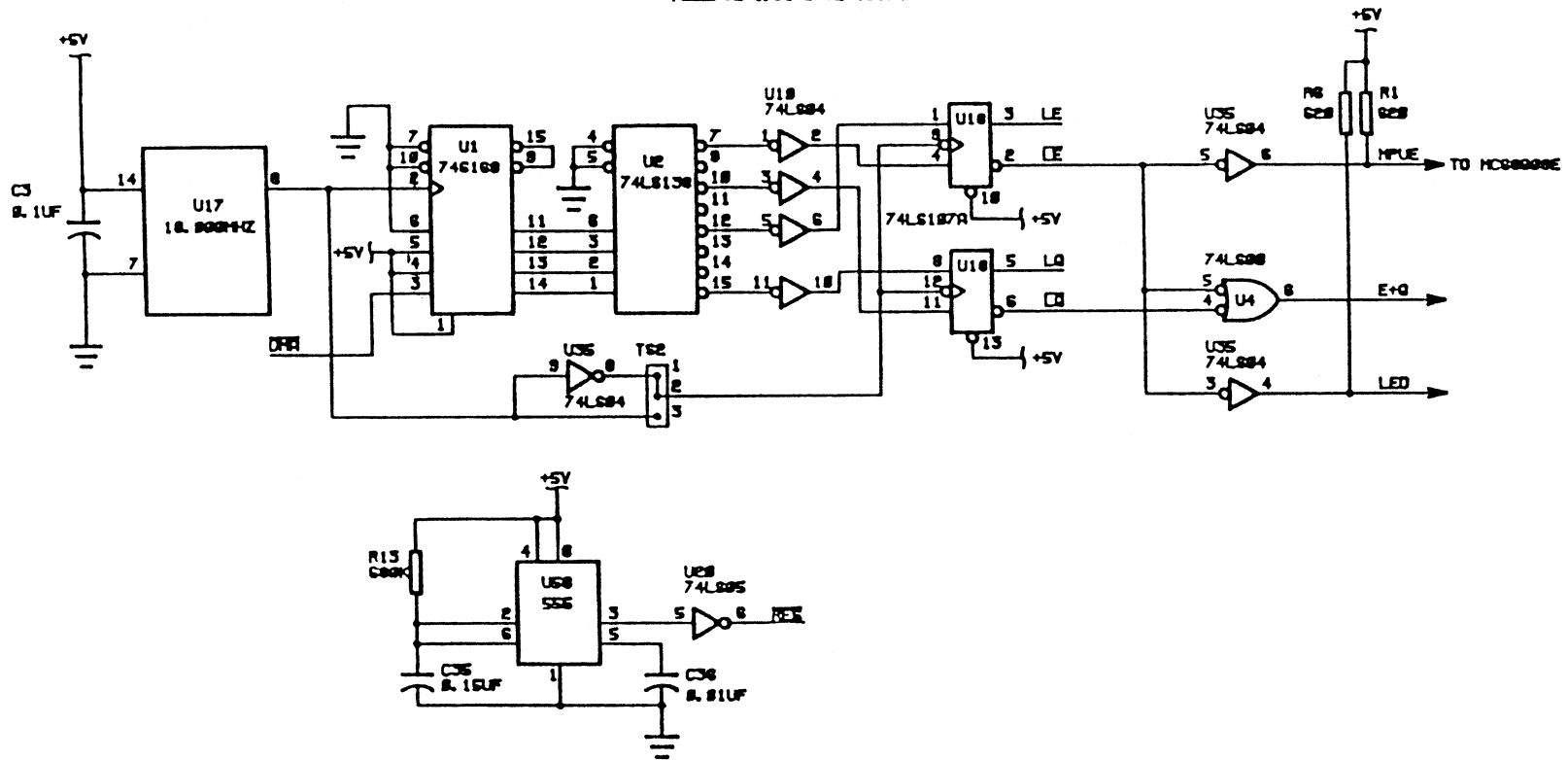
8. COMPONENT PLACEMENT BOARD STYLE #1

DRAWN BY A. G. L.

3A109003-11
 CREATIVE MICRO SYSTEMS
 9639 REV A MEMORY
 MANAGEMENT PROCESSOR
 RELEASE 1.03
 DATE: 840220

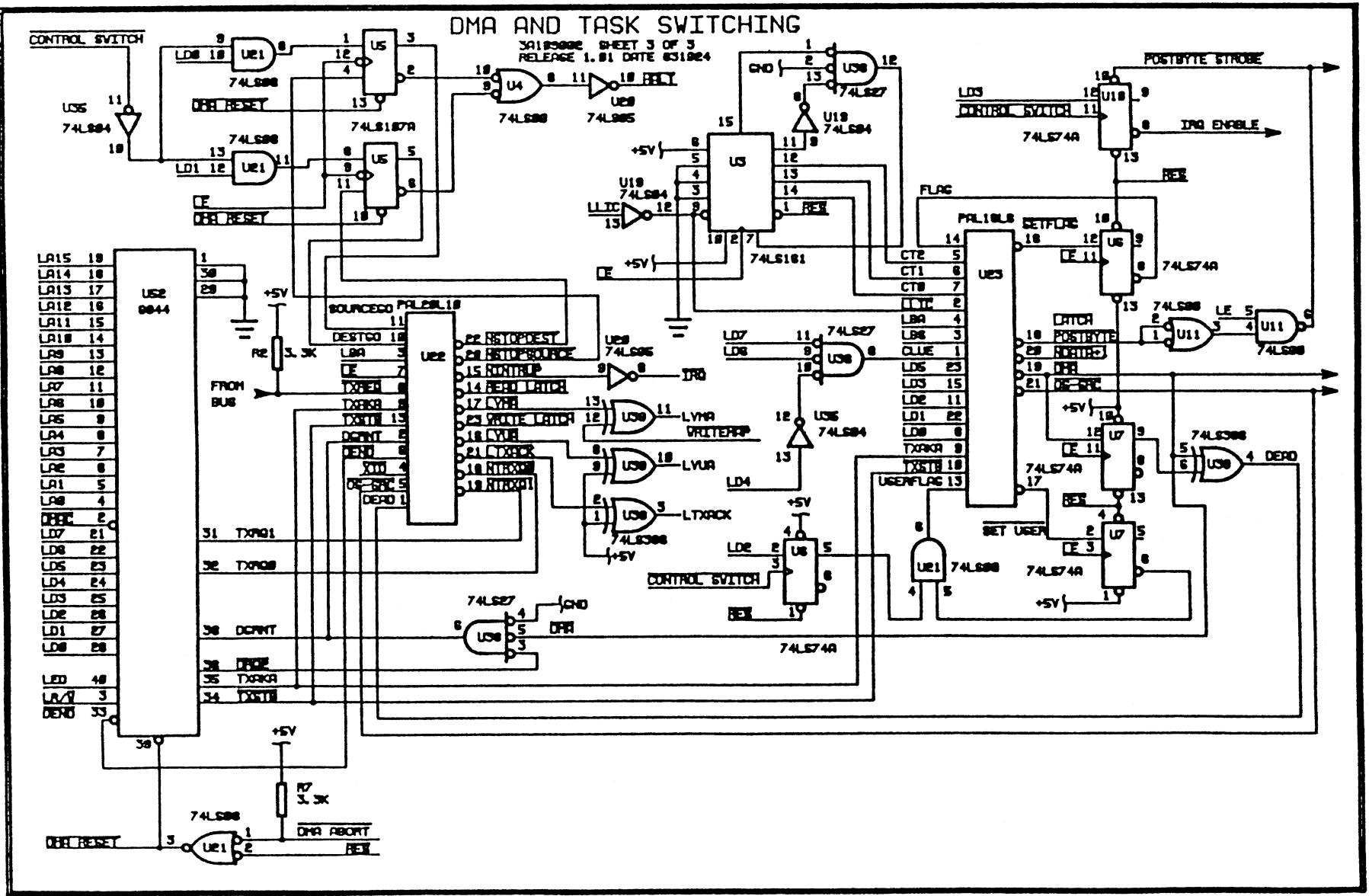
9639 CLOCK CIRCUIT SECTION

3A180002 SHEET 2 OF 3
RELEASE 1.81 DATE 831024



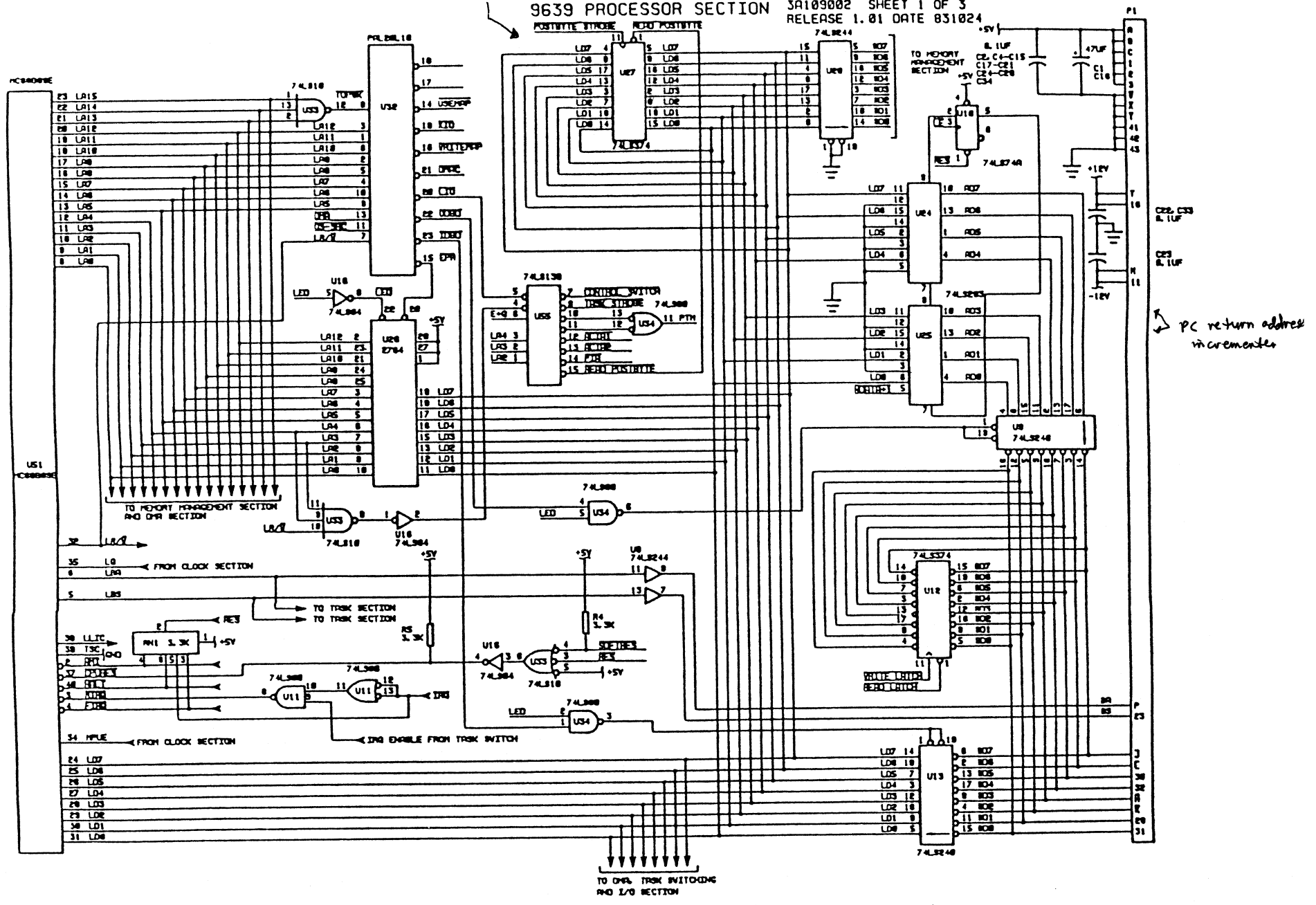
DMA AND TASK SWITCHING

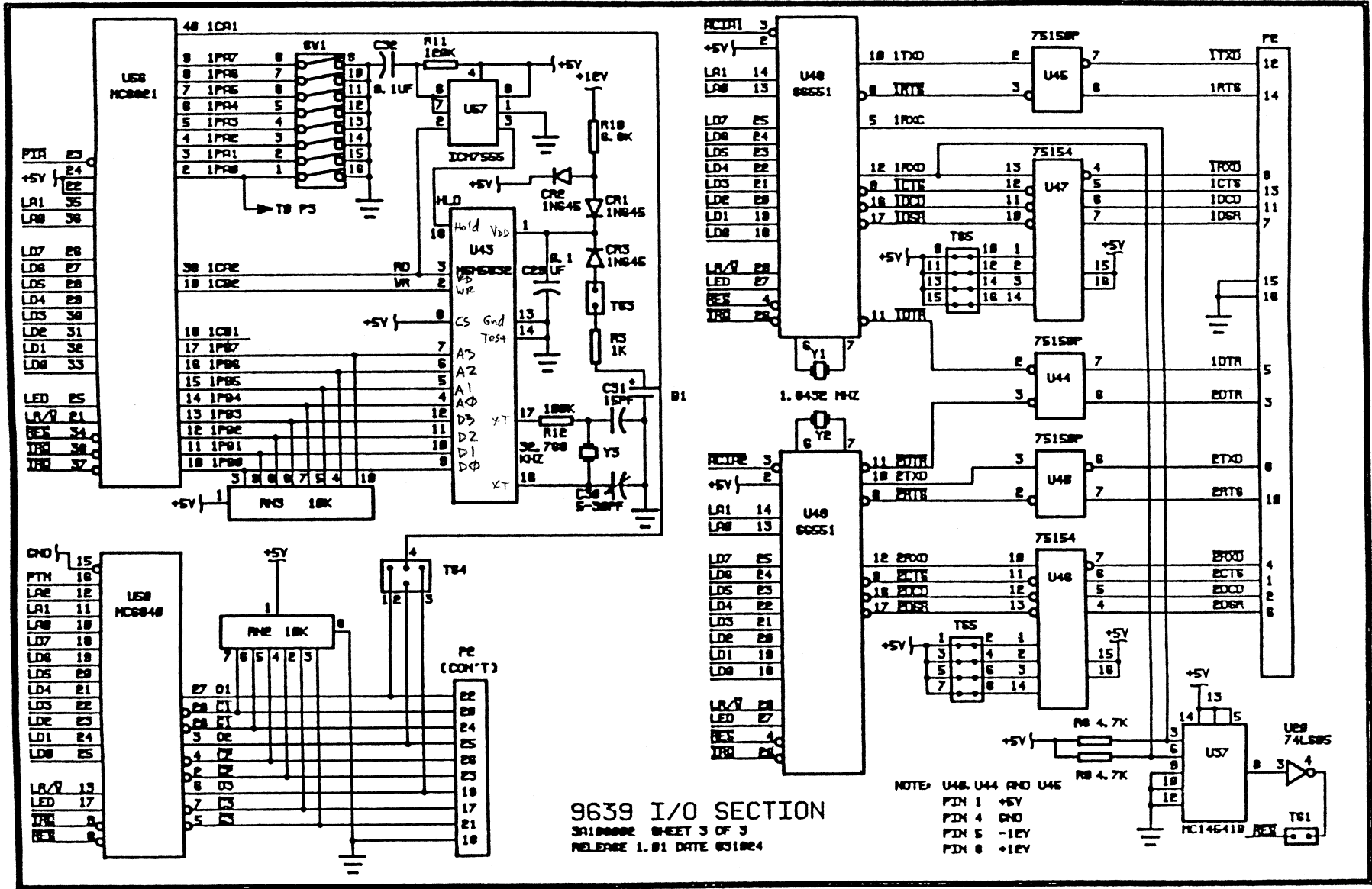
38110000 SHEET 3 OF 3
RELEASE 1.01 DATE 031984



74LS27 + PAL implements an "RTI" (#3B) code recognizer - It is enabled when a flag bit is set. When RTI found (no other #3B ^{subsequently} appear on the data bus) it resets the flag bit & switches from system to user task register.

9639 PROCESSOR SECTION 3A109002 SHEET 1 OF 3
 RELEASE 1.01 DATE 031024





9639 I/O SECTION
 3A100000 SHEET 3 OF 3
 RELEASE 1.01 DATE 031004

NOTE: U46, U44 AND U45
 PIN 1 +5V
 PIN 4 GND
 PIN 5 -12V
 PIN 8 +12V