1. INTRODUCTION

For fast response time and high data transfer rates, many computers provide direct memory access (DMA) channels. These provide a fast, intimate coupling between the computer memory and a high speed peripheral for the mass transfer of blocks of data. Rather than executing an interrupt routine for each word of data transferred, a DMA channel uses only one memory cycle for each word of a block transferred.

The DMA interface (controller) must be initialized by the CPU when it is ready to transfer a block of data. This is done by supplying the location and length of the block in memory and the direction of transfer to the controller. After enabling the controller, the CPU continues with other tasks, leaving the data transfer responsibility solely to the DMA controller.

Depending on the speed and type of the peripheral device, two types of DMA can take place: single transfer and burst mode. Both move a block of data, but each interacts differently with the CPU.

Single transfers are used for slower DMA peripheral devices. The DMA controller waits for the peripheral to be ready for one data transfer. When the peripheral has this data ready, the controller signals the CPU. When the CPU reaches the next available non-memory cycle, it suspends operation to permit the DMA controller access to memory. The DMA controller steals one memory cycle to transfer one byte or word of data between the peripheral and memory. At the completion of this cycle, the CPU resumes operation. These individual transfers continue at the rate determined by the peripheral until the entire block of data has been transferred.

The burst mode of DMA is used for very high speed devices. The DMA controller waits for the peripheral to be ready to transfer the whole block of data. When the peripheral is ready, the controller signals the CPU. When the CPU reaches the next available non-memory cycle, the CPU suspends operation to permit the DMA controller access to memory. The DMA controller then generates multiple, consecutive memory cycles to transfer the entire block of data before returning control to the CPU. The transfer rate is very high using burst mode DMA, but the DMA completely halts the processor until the data is transferred.

1.1 DESCRIPTION

The TMS 9911 Direct Memory Access Controller (DMAC) is a peripheral device designed for use with the Texas Instruments 9900 family of microprocessors. The TMS 9911 is designed to interface to the CPU via the communications register unit (CRU). The TMS 9911 is fabricated using N-channel, silicon gate, MOS technology. It is TTL-compatible on all inputs and outputs, including the power supply (+5 V) and single phase clock. Memory control signals and sequential memory addresses are generated by the TMS 9911 for two DMA channels, allowing two independent DMA devices to access memory autonomously with respect to the CPU. The number of DMA channels may be extended by use of multiple DMAC's. The interfaces between the DMAC and the CPU, system memory, and DMA peripheral devices require a minimum amount of additional hardware.

1.2 KEY FEATURES

- Two independent DMA channels
- Burst and single transfer capability
- Timing control provided for slow memories
- 1 MHz block transfer rate (assuming 3 MHz clock)
- DMA channel chaining

- Word or byte transfers
- TTL-compatibility including power supply and clock
- Standard 40-pin plastic or ceramic package
- Easily cascaded for prioritized expansion
- N-channel silicon gate MOS technology
- Low cost

1.3 TYPICAL APPLICATION

Figure 1 is a block diagram of a system with a TMS 9911 DMAC incorporated to control a DMA channel between a memory mapped peripheral and a 9900 family CPU. A brief discussion of this application follows. Detailed aspects of specific applications are given in subsequent sections of this manual.

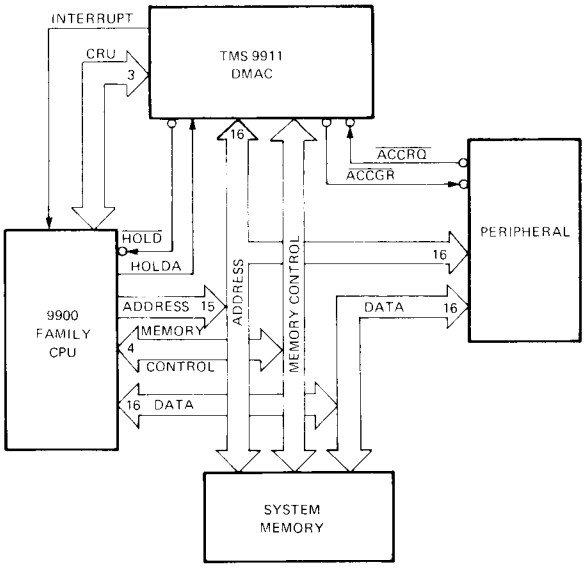


FIGURE 1 – TMS 9911 TYPICAL APPLICATION

The 9900 family CPU initializes the DMAC via its Communications Register Unit (CRU). The memory start address and the next memory address after the last address are stored into registers in the DMAC along with the direction of transfer. Then the DMAC is enabled, also using the CRU.

The peripheral indicates that it is ready for a transfer by use of the access request ($\overline{\text{ACCRQ}}$) line. If that channel is enabled, the DMAC will then try to get bus control by issuing a hold signal ($\overline{\text{HOLD}}$) to the CPU. At the next available non-memory cycle, the CPU relinquishes the bus and responds with a hold acknowledge ($\overline{\text{HOLDA}}$) signal. The DMAC signals the peripheral that it may transfer data by issuing an access grant ($\overline{\text{ACCGR}}$) signal. The DMAC then generates the proper memory address and memory control signals necessary for the data transfer directly between the peripheral and memory. This transfer may involve more than one memory cycle, depending on the type of DMA being performed (block or single transfer). After the data transfer has been completed, the DMAC releases its hold signal and the CPU continues execution of its instructions.

The DMAC generates an interrupt to signal the CPU that it has completed its last data transfer. This completes the DMA transfer until another is initiated by the CPU.

2. ARCHITECTURE

The TMS 9911 Direct Memory Access Controller (DMAC) is designed to provide two low-cost DMA channels for devices used with the 9900 family of microprocessors. Figure 2 is a block diagram of the DMAC's internal architecture. The DMAC consists of control logic, CRU interface, internal address and status registers, and increment and compare circuitry. Each channel contains a memory address register (MAR) and last address register (LAR). The MAR is incremented after each data transfer and compared to the LAR. Control logic generates the correct memory timing signals during transfers.

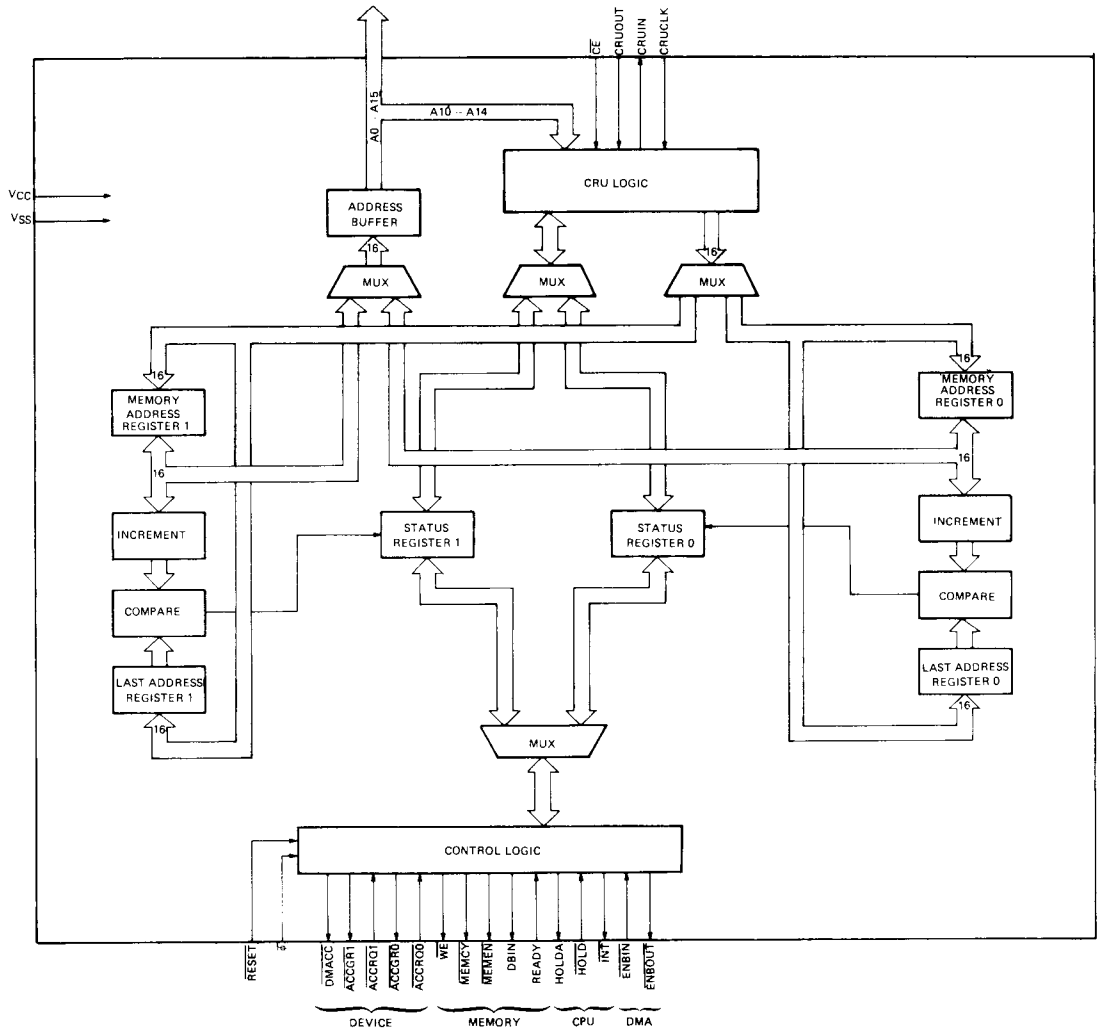


FIGURE 2 – TMS 9911 DMAC BLOCK DIAGRAM

2.1 CRU INTERFACE

The CPU communicates with the TMS 9911 DMAC by means of the Communications Register Unit (CRU). Figure 3 illustrates the CRU interface between the DMAC and a TMS 9900; Figure 4 illustrates the interface for a TMS 9980A or TMS 9981 CPU.

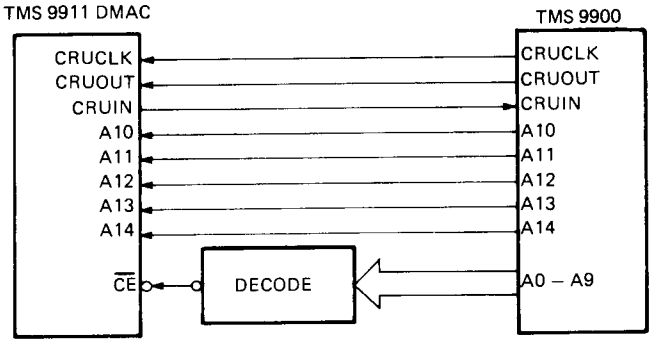


FIGURE 3 — TMS 9911 CRU INTERFACE (TMS 9900 SYSTEM)

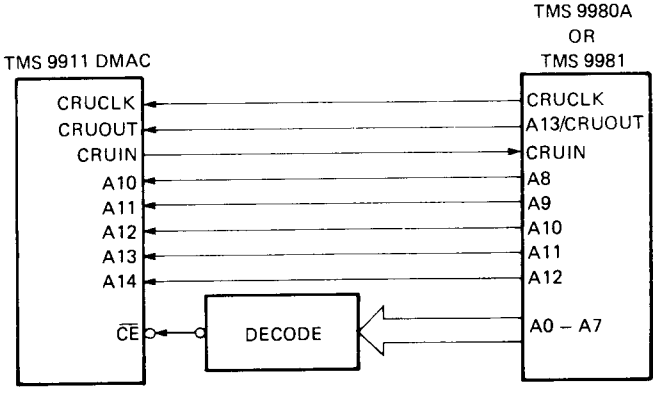


FIGURE 4 — TMS 9911 CRU INTERFACE (TMS9980A or 9981 SYSTEM)

The CRU signal pins (CRUOUT, CRUIN, CRUCLK) on the CPU and DMAC are tied directly together as shown. Data is serially output to the DMAC from the CPU via CRUOUT and serially input from the DMAC to the CPU via CRUIN. CRUCLK is used as a strobe for data transfers from the CPU to the DMAC (CRUOUT synchronization).

The least significant five bits of the CPU address bus are connected to the DMAC to specify which bit is being addressed. The most significant address bits are decoded to select the TMS 9911 via the chip enable ($\overline{CE}$). When $\overline{CE}$ is inactive (HIGH), the TMS 9911 sets CRUIN to high-impedance (allowing CRUIN to be used in an OR-tied bus). When $\overline{CE}$ is high, the DMAC still sees the bit select lines, but no command action is taken.

The CRU is a bit-addressable (4096 bits) synchronous serial interface over which a single instruction can transfer between 1 and 16 bits serially. Each of the 4096 bits of the CRU space has a unique address that can be read from and written to. During multiple-bit CRU transfers, the CRU address is incremented at the beginning of each CRU cycle to point to the next consecutive CRU bit.

When the 9900 CPU executes a CRU instruction, the processor uses the contents of workspace register 12 as a software base address. (Refer to the *TMS 9900 Microprocessor Data Manual* for a complete discussion of how CRU addresses are derived.) The CRU address is brought out on the 15-bit address bus; this means that the least significant bit of R12 is not brought out of the CPU. During CRU cycles, the memory control lines ($\overline{MEMEN}$, $\overline{WE}$, and DBIN) are all inactive; $\overline{MEMEN}$'s inactive state (HIGH) indicates the address is not a memory address and therefore is a CRU address or external instruction code. Also, when $\overline{MEMEN}$ is inactive (HIGH) and valid address is present, address bits A0-A2 must all be zero to constitute a valid CRU address; if address bits A0-A2 are other than all zeros, they are indicating an external instruction code. In summary, address bits A3-A14 contain the CRU address to be decoded, address bits A0-A2 must be zero, and $\overline{MEMEN}$ must be inactive (HIGH) to indicate a CRU cycle.

2.1.1 CPU Output To DMAC Via CRU

The TMS 9911 DMAC occupies 32 bits of CRU output space, of which 27 are used by the CPU to communicate command and control information to the DMAC. Table 1 shows the mapping between CRU address select line (A10-A14) and DMAC functions. Each CRU addressable output bit is described in the following paragraphs.

TABLE 1
TMS 9911 DMAC OUTPUT SELECT BIT ASSIGNMENTS

$ADDRESS_2$					$SELECT_{10}$	NAME	DESCRIPTION
A10	A11	A12	A13	A14			
1	1	1	1	1	31	SWRST	Software Reset
1	1	1	1	0	30	CHAIN	DMA Transfer Chain Mode
					29-26		Not Used
1	1	0	0	1	25	CHSEL	Channel Select
1	1	0	0	0	24	CHRET	Channel Return
					23		Not Used
					22-0		Channel N Control Functions

2.1.1.1 CRU Output Bits Common To Both DMA Channels

Bit 31 (SWRST) — Software Reset. Writing a one or zero to SWRST using a single-bit CRU instruction (SBO or SBZ) causes the DMAC to be reset to a known state in which interrupts and both channels are disabled, memory address register and last address register equal 0, all status bits are inactive, and CHSEL and CnASEL equal 0. The device must now be initialized by the CPU.

Bit 30 (CHAIN) — Chain. Writing a one to bit 30 causes the transfer control of both channels to be handled by the $\overline{ACCRQ0}/\overline{ACCGR0}$ handshake pair. Both channels must be enable. Channel 0 has first control, followed by channel 1 when channel 0 addresses are complete. Writing a zero to bit 30 causes each channel to operate independently. (See Section 2.3 for chain mode description.)

Bits 29-26 — Not used.

Bit 25 (CHSEL) — Channel Select. CHSEL is used to select which channel's control functions can be accessed. Writing a one to bit 25 allows channel 1 control functions to be written to bits 0-22 and read from bits 0-23. Writing a zero to bit 25 allows channel 0 control functions to be written to bits 0-22 and read from bits 0-23.

Bit 24 (CHRET) — Channel Return. When bit 24 is written to with either a one or a zero, the value saved on the one-bit CHSEL stack is written to bit 25 (CHSEL). This function facilitates DMAC interrupt processing of one channel while the other non-interrupting channel is being set up by writing to bit 24 at the end of the interrupt service routine.

Bit 23 — Not used.

2.1.1.2 CRU Output Bits That Apply To Only One Channel

The remaining CRU bits (22-0) apply only to the channel (n) selected by CHSEL (bit 25). Table 2 shows the mapping between CRU address select lines A10-A14 and channel control functions.

TABLE 2
TMS 9911 DMAC CHANNEL CONTROL FUNCTIONS

$ADDRESS_2$					$SELECT_{10}$	NAME	DESCRIPTION
A10	A11	A12	A13	A14			
1	0	1	1	0	22	IENBn	Interrupt Enable
1	0	1	0	1	21	OPCMPn	Operation Complete
1	0	1	0	0	20	WRDSLn	Word Select
1	0	0	1	1	19	MEMRDn	Memory Read
1	0	0	1	0	18	CNTNUn	Continue After Last Address
1	0	0	0	1	17	CHENBn	Channel Enable
1	0	0	0	0	16	CnASEL	Channel Address Select
0	1	1	1	1	15-0	CnA(0-15) or CnL(0-15)	Memory Address Register or Last Address Register
0	0	0	0	0			

Bit 22(IENBn) — Interrupt Enable. Writing a one to bit 22 causes the $\overline{INT}$ output to become active when channel n address is incremented to channel n last address (OPCMPn = 1). Writing a zero to bit 22 disables the $\overline{INT}$ output.

Bit 21 (OPCMPn) — Operation Complete. Writing a zero to bit 21 causes the channel n interrupt to be cleared. Writing a one to bit 21 has no effect.

Bit 20 (WRDSLn) — Word Select. Writing a zero to bit 20 causes the channel n address register to be incremented by one after each data transfer for byte structured data. Writing a one to bit 20 causes the channel n address register to be incremented by two after each data transfer for word structured data. Note that the byte mode of operation should only be used on systems which implement the LSB (A15) of the address bus.

Bit 19 (MEMRDn) — Memory Read. Writing a one to bit 19 causes each memory access through channel n to be performed with DBIN = 1 (memory read); writing a zero causes each memory access to be performed with DBIN = 0 (memory write).

Bit 18 (CNTNUn) — Continue. Writing a one to bit 18 prevents channel n from being automatically disabled when the last address has been reached. In this mode, each $\overline{\text{ACCRQ}}$ by channel n will cause a data transfer and increment of MARn (assuming the channel is enabled) regardless of the LARn value. Writing a zero to bit 18 causes channel n to be automatically disabled (CHENBn and CHACTn are reset) when the last address has been reached.

Bit 17 (CHENBn) — Channel Enable. Writing a one to bit 17 allows channel n to transfer data as requested by DMA device n. Writing a zero to bit 17 disables channel n operation ($\overline{\text{ACCRQ}}$n is ignored) and resets input bit 23 (CHACTn).

Bit 16 (CnASEL) — Address Select. Writing a one to bit 16 causes data loaded into addresses 0-15 to be directed to the channel n memory address register; writing a zero causes the data to be directed to the channel n last address register.

Bits 15-0 (CnA or CnL) — Memory Address Register (MAR) or Last Address Register (LAR). The value written to bits 15-0 is loaded as an address into the register specified by bit 16 (CnASEL). Bit 0 is the most significant address bit, bit 15 the least significant address bit. The value loaded into the memory address register defines the address for the first memory access and is subsequently incremented according to bit 20 (WRDSLn). The value loaded into the last address register defines the next address after the last transfer occurs. This value can be calculated by:

LAR = MAR + number of **bytes** transferred.

2.1.2 Input To CPU From DMAC Via CRU

The TMS 9911 DMAC occupies 32 bits of the CPU's input space, of which 29 are used. The CPU reads the 29 bits from the DMAC to sense the status of the device. Output bit 25 (CHSEL) is used to select which channel's control function values may be read. Table 3 shows the mapping between CRU bit addresses and TMS 9911 output data. Each CRU addressable DMAC input bit is described in the following paragraphs.

TABLE 3
TMS 9911 DMAC INPUT SELECT BIT ASSIGNMENTS

ADDRESS$_2$					SELECT$_{10}$	NAME	DESCRIPTION
A10	A11	A12	A13	A14			
1	1	1	1	1	31	DMAPRS	DMA Present
1	1	1	1	0	30	CHAIN	Chain Status
					29-28		Not Used
1	1	0	1	1	27	INT1	Channel 1 Interrupt
1	1	0	1	0	26	INT0	Channel 0 Interrupt
1	1	0	0	1	25	CHSEL	Channel Select Status
					24		Not Used
1	0	1	1	1	23	CHACTn	Channel n Active
1	0	1	1	0	22	IENBn	Interrupt Enable Status
1	0	1	0	1	21	OPCMPn	Operation Complete Status
1	0	1	0	0	20	WRDSLn	Word Select Status
1	0	0	1	1	19	MEMRDn	Memory Read Status
1	0	0	1	0	18	CNTNUn	Continue Status
1	0	0	0	1	17	CHENBn	Channel Enable Status
1	0	0	0	0	16	CnASEL	Channel Address Select Status
0	1	1	1	1		CnA(0-15)	Memory Address Register
					15-0	or	or
0	0	0	0	0		CnL(0-15)	Last Address Register

Bit 31 (DMAPRS) — DMAC Present. Indicates that DMAC is connected. Always set to one.

Bit 30 (CHAIN) — Chain. Chain indicates the status of output bit 30 (CHAIN).

Bits 29-28 — Not used.

Bit 27 (INT1) — Channel 1 Interrupt. INT1 is set to one after the transfer of data through channel 1 has been completed. INT1 is reset when a zero is written to bit 21 (OPCMPn) while CHSEL = 1.

Bit 26 (INT0) — Channel 0 Interrupt. INT0 is set to one after the transfer of data through channel 0 has been completed. INT0 is reset when a zero is written to bit 21 (OPCMPn) while CHSEL = 0.

Bit 25 (CHSEL) — Channel Select. CHSEL indicates which channel's control functions can be accessed. CHSEL must be set to one for channel 1 and reset to zero for channel 0.

Bit 24 — Not used.

Bit 23 (CHACTn) — Channel Active. CHACTn indicates if a block transfer has been started but is not complete. CHACTn is set to one when channel n becomes active and is reset to zero when CHENBn is reset. When in chain mode, CHACT1 is set to one only after CHENB0 and CHACT0 are reset (channel 0 inactive).

Bit 22 (IENBn) — Interrupt Enable. IENBn indicates the status of CRU control bit 22 (IENBn).

Bit 21 (OPCMPn) — Operation Complete. OPCMPn is set to one when the channel n memory address register is incremented to the value of the channel n last address register regardless of the status of CNTNUn. OPCMPn is reset to zero when a zero is written to output bit 21 (OPCMPn).

Bit 20 (WRDSLn) — Word Select. WRDSLn indicates the status of CRU control bit 20 (WRDSLn).

Bit 19 (MEMRDn) — Memory Read. MEMRDn indicates the status of CRU control bit 19 (MEMRDn).

Bit 18 (CNTNUn) — Continue. CNTNUn indicates the status of output bit 18 (CNTNUn).

Bit 17 (CHENBn) — Channel Enable. CHENBn is set to a one when channel n is enabled by writing a one to CRU control bit 17 (CHENBn). CHENBn is reset to zero when channel n is disabled either by writing a zero to control bit 17 (CHENBn) or when the last address has been reached while CNTNUn = 0.

Bit 16 (CnASEL) — Address Select. CnASEL indicates the status of output bit 16 (CnASEL).

Bits 15-0 (CnA or CrL) — Address Register or Last Address Register. When CnASEL = 1, inputs bits 15-0 contain the value of the channel n memory address register (CnA). When CnASEL = 0, input bits 15-0 contain the value of the channel n last address register. In both cases bit 0 is the most significant address bit; bit 15 the least significant address bit.

2.2 DEVICE OPERATION

Operation of the TMS 9911 DMAC is diagrammed in Figure 5. This figure shows the relevant sequences by which a DMAC gains control of memory bus, contends with system memory not being ready, transfers control to another DMAC, and relinquishes control back to the CPU. Circled labels apply to the discussion of device operation in the following paragraphs.

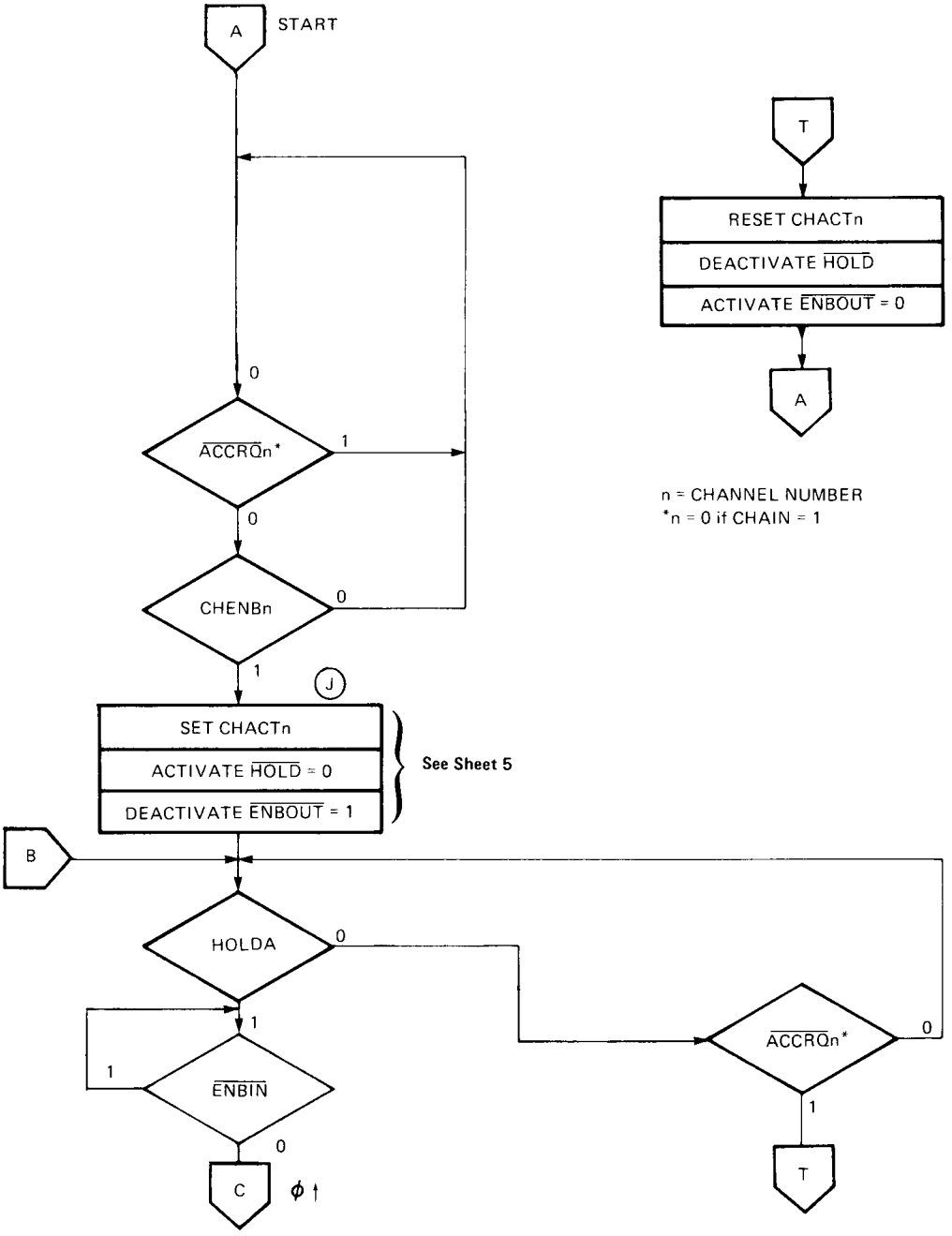


FIGURE 5 (SHEET 1 OF 5) – DMAC CONTROL FLOW

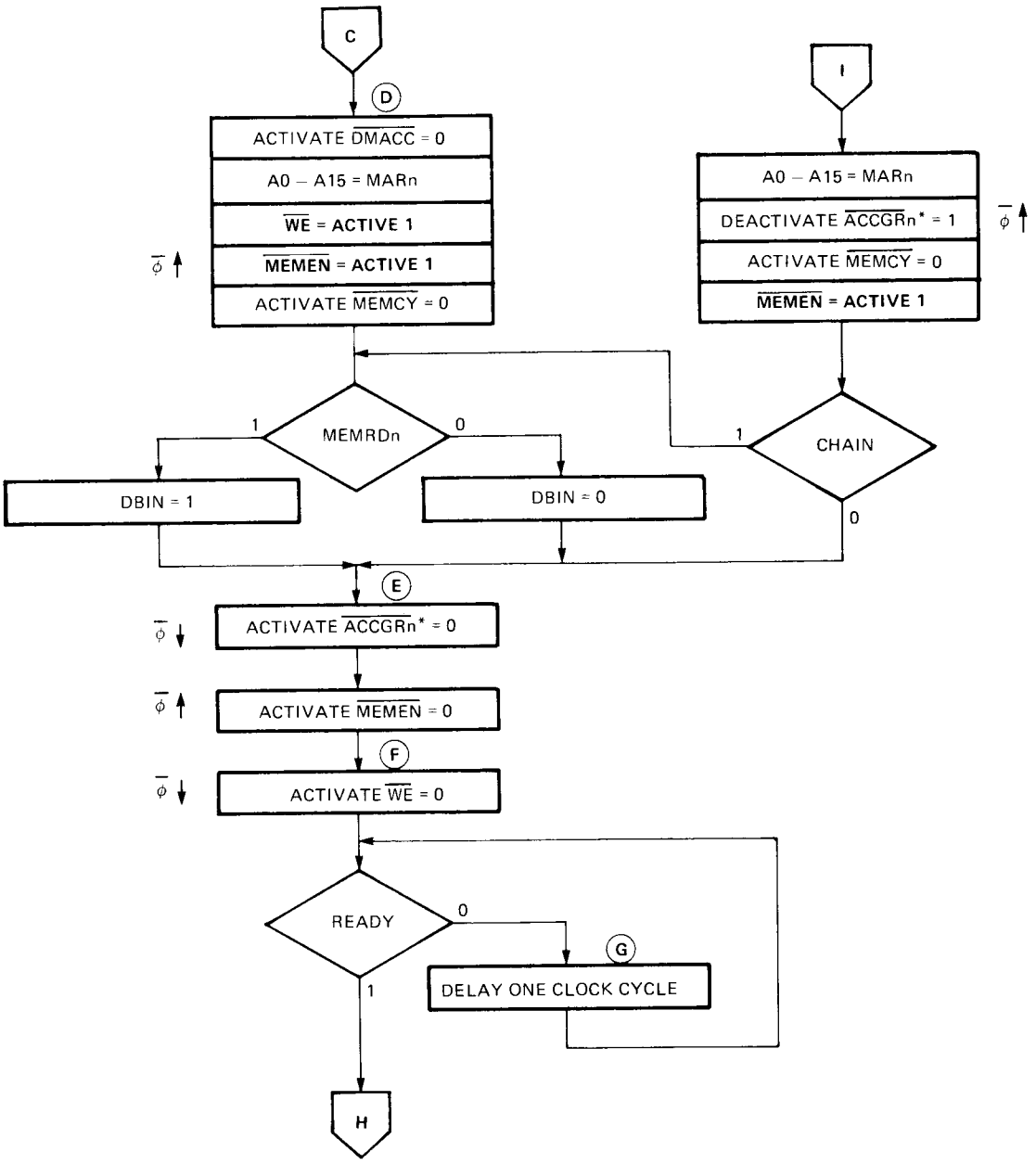


FIGURE 5 (SHEET 2 OF 5) -- DMAC CONTROL FLOW

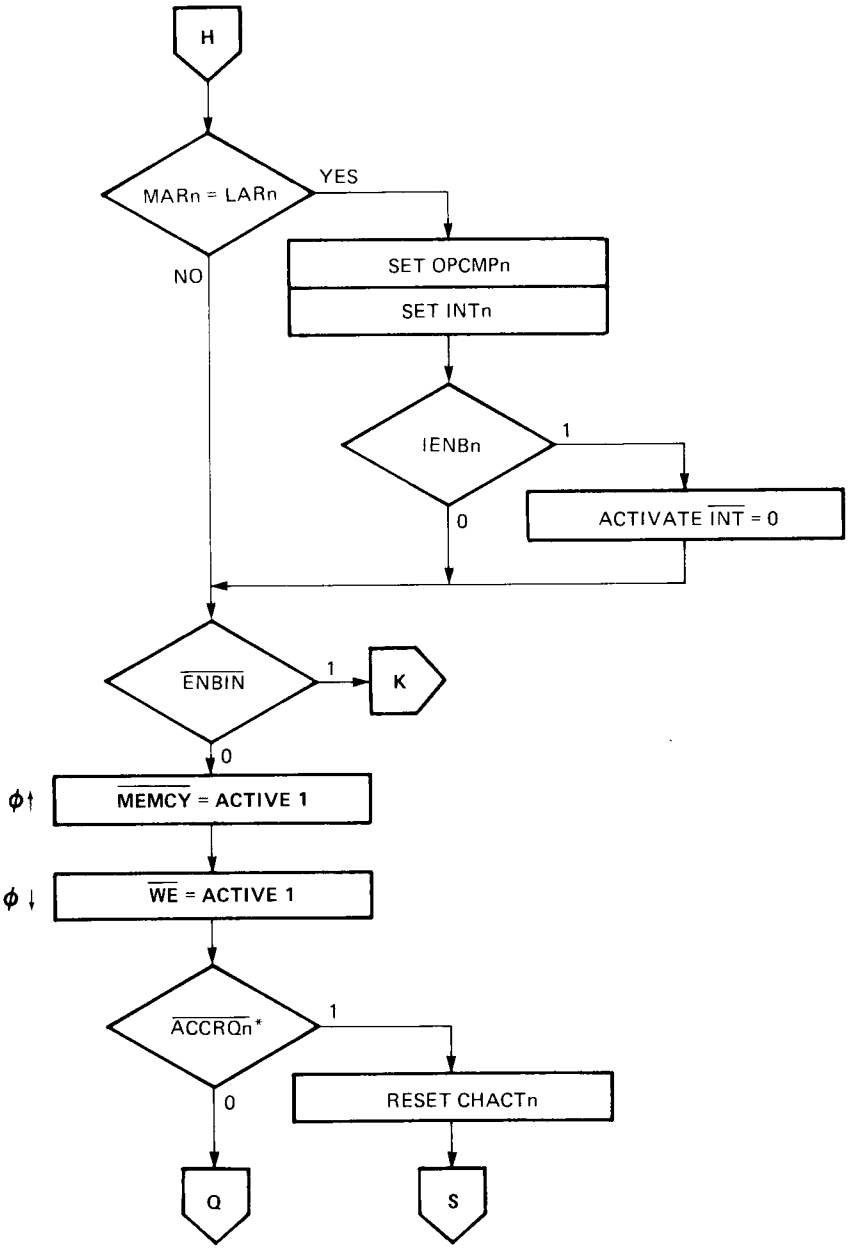


FIGURE 5 (SHEET 3 OF 5) – DMAC CONTROL FLOW

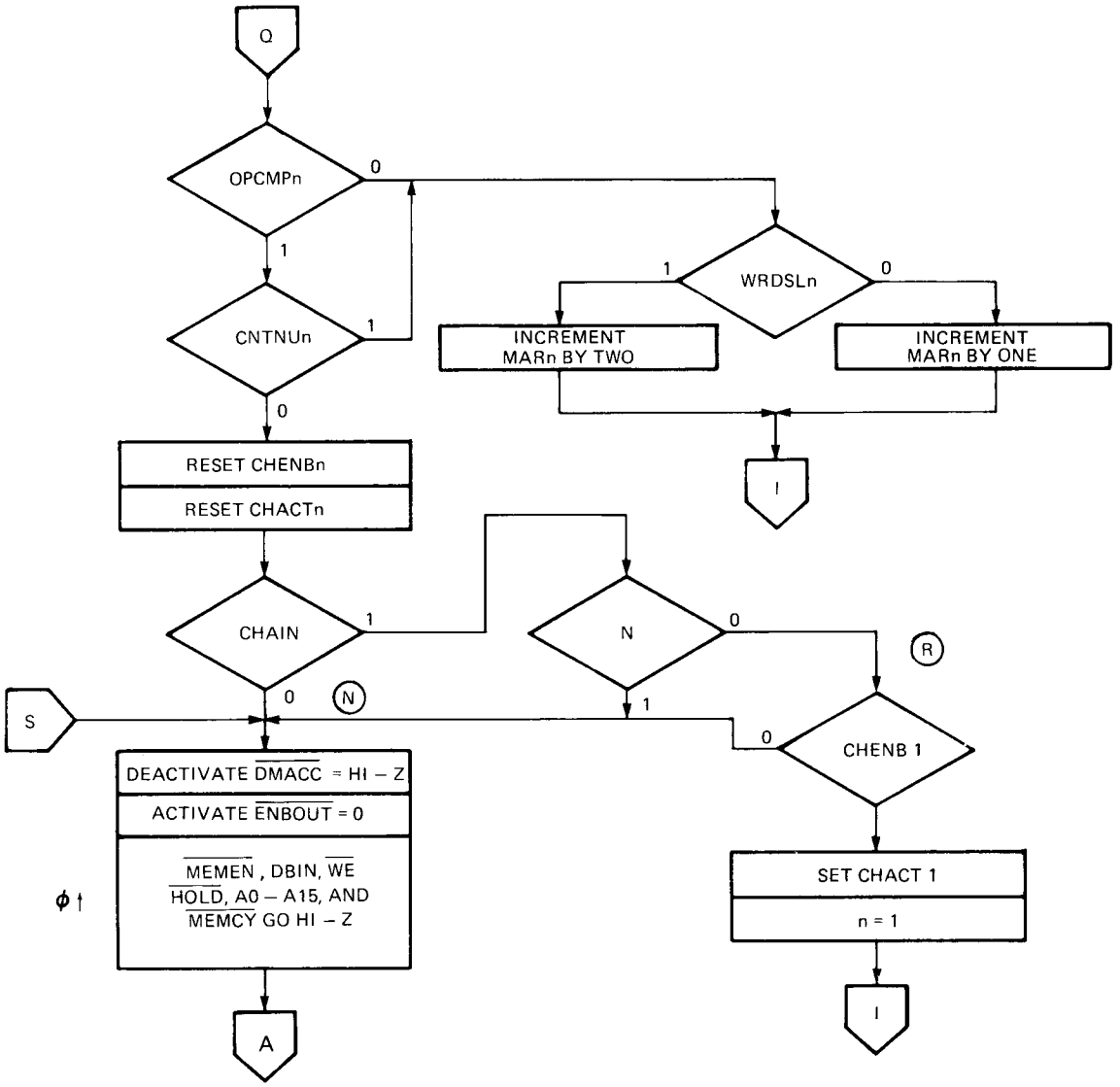


FIGURE 5 (SHEET 4 OF 5) – DMAC CONTROL FLOW

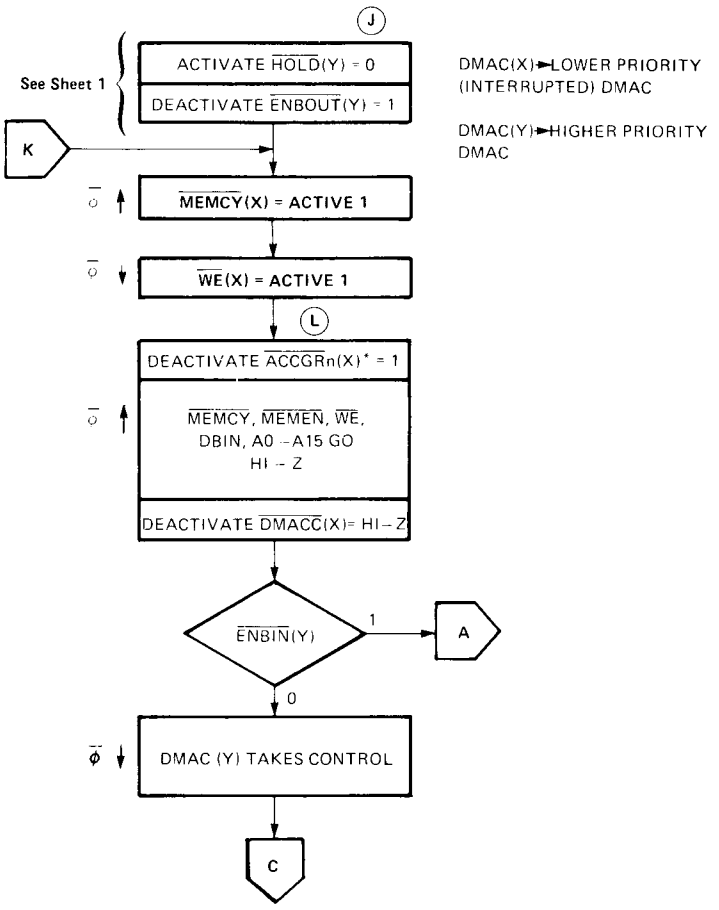


FIGURE 5 (SHEET 5 OF 5) – DMAC CONTROL FLOW

2.2.1 DMAC Acquisition of Bus Control

Figure 6 shows the timing relationships that occur when the DMAC requests and receives control of the system bus. This sequence begins with the DMAC acquistion of an $\overline{ACCRQ}$ signal (point A) from DMA peripheral device (DMAPD). If the channel is enabled, this signal is recognized by the DMAC, which then issues a $\overline{HOLD}$ signal (active low) to the CPU, sets CHACTn, and deactivates its $\overline{ENBOUT}$ signal (active low) to the lower priority DMAC's to inform them that it is contending for access to the system buses (point B). The $\overline{HOLD}$ signal is received by the CPU. When the CPU has finished its current memory cycle, it responds by issuing a HOLDA signal (point C). This signal instructs the DMAC that the CPU is relinquishing control of the bus and the DMAC may take control. The maximum time from the receipt of $\overline{HOLD}$ until the CPU responds with HOLDA is given by:

$$t_{max} = (8 + 3W)\, t_c(\varnothing)$$

where W is the number of wait states in a worst-case CPU memory cycle.

During the waiting period between the issue of $\overline{HOLD}$ and the receipt of HOLDA, the DMAC will sample its incoming $\overline{ENBIN}$ to ensure that it may still be a contender for bus control. The DMAC accepts the HOLDA signal and on the following rising edge of $\overline{\phi}$ starts its memory cycle (point D).

2.2.2 DMAC Memory Control Sequence

After the HOLDA and $\overline{ENBIN}$ conditions are correct (as described above), the DMAC address bus and control signals ($\overline{MEMEN}$, DBIN, $\overline{WE}$, and $\overline{MEMCY}$) go active from their HI-Z state. The level of DBIN is determined by the state of the MEMRDn status bit for the active channel. $\overline{WE}$ and $\overline{MEMEN}$ are intially high and $\overline{MEMCY}$ low (point D). At the next falling edge of $\overline{\phi}$, $\overline{ACCGR}$, which was high inhibiting DMAPD access, goes low instructing the DMAPD to output data (point E).

On the following rising edge of $\overline{\phi}$, $\overline{MEMEN}$ goes low. At the subsequent falling edge of $\overline{\phi}$, $\overline{WE}$ falls, enabling the system memory (dependent on DBIN) or DMAPD (dependent on $\overline{DMACC}$) to write data (point F). READY is sampled at this time to ensure that it is a valid time to read or write data to the system memory. If READY = 0 (indicating a slow memory access), then the memory cycle is extended until READY = 1. This is illustrated as a WAIT system clock cycle in Figure 6 (point G).

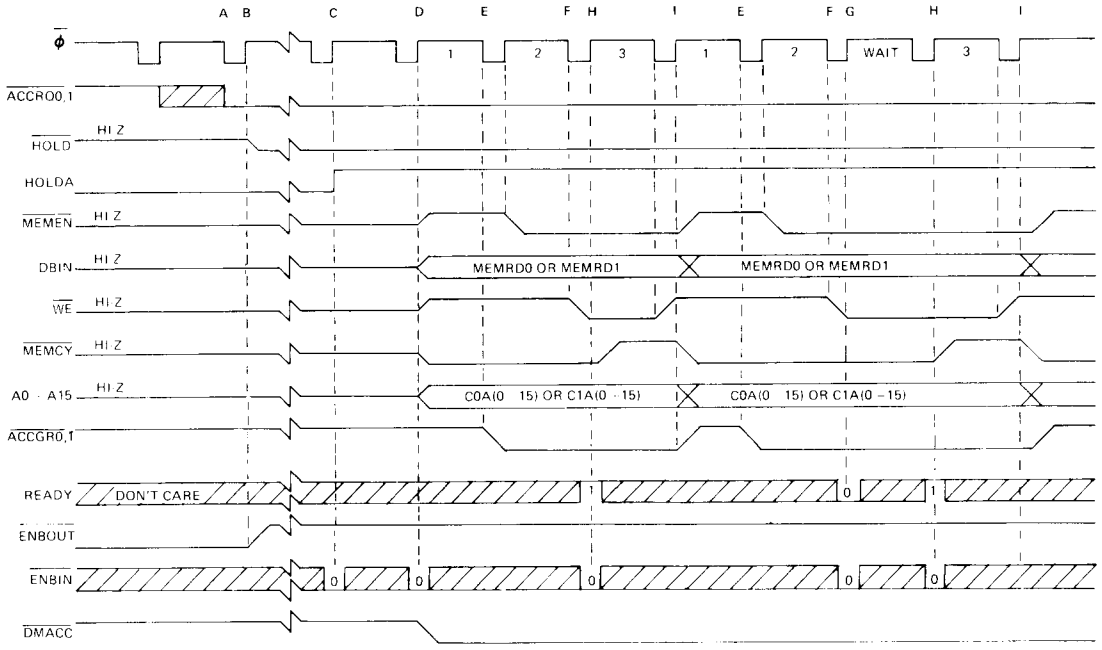


FIGURE 6 -- CONTROL SEQUENCE - DMAC TAKEOVER OF COMMAND AND MEMORY CYCLE WITH ONE WAIT STATE

At the next rising edge of $\overline{\phi}$, $\overline{MEMCY}$ goes high to indicate that the last system clock cycle in the memory cycle has started (point H). At the same time, the memory address register is compared to the last address register; if equal, the OPCMPn status flag is set and an interrupt is generated if enabled. At the falling edge of $\overline{\phi}$, $\overline{WE}$ goes high to disable any writing that is taking place and on the subsequent rising edge the current memory cycle is complete. $\overline{ACCGR}$ returns high to inhibit access of DMAPD, $\overline{MEMCY}$ returns low (since the last clock cycle is finished), and $\overline{MEMEN}$ returns high (point I).

If another memory cycle is to be allowed, A0-A15 will output a new incremented address. If in chain mode and control has been exchanged to the other channel, DBIN will be output in accordance with the new MEMRD state of the new active channel. Otherwise DBIN will continue in its original state.

2.2.3 DMAC to DMAC Control Transfer

Figure 7 shows the timing relationships that occur when a higher priority DMAC requires access while a lower priority DMAC is currently active. Assume the lower priority DMAC is DMAC(X) and the higher priority DMAC is DMAC(Y).

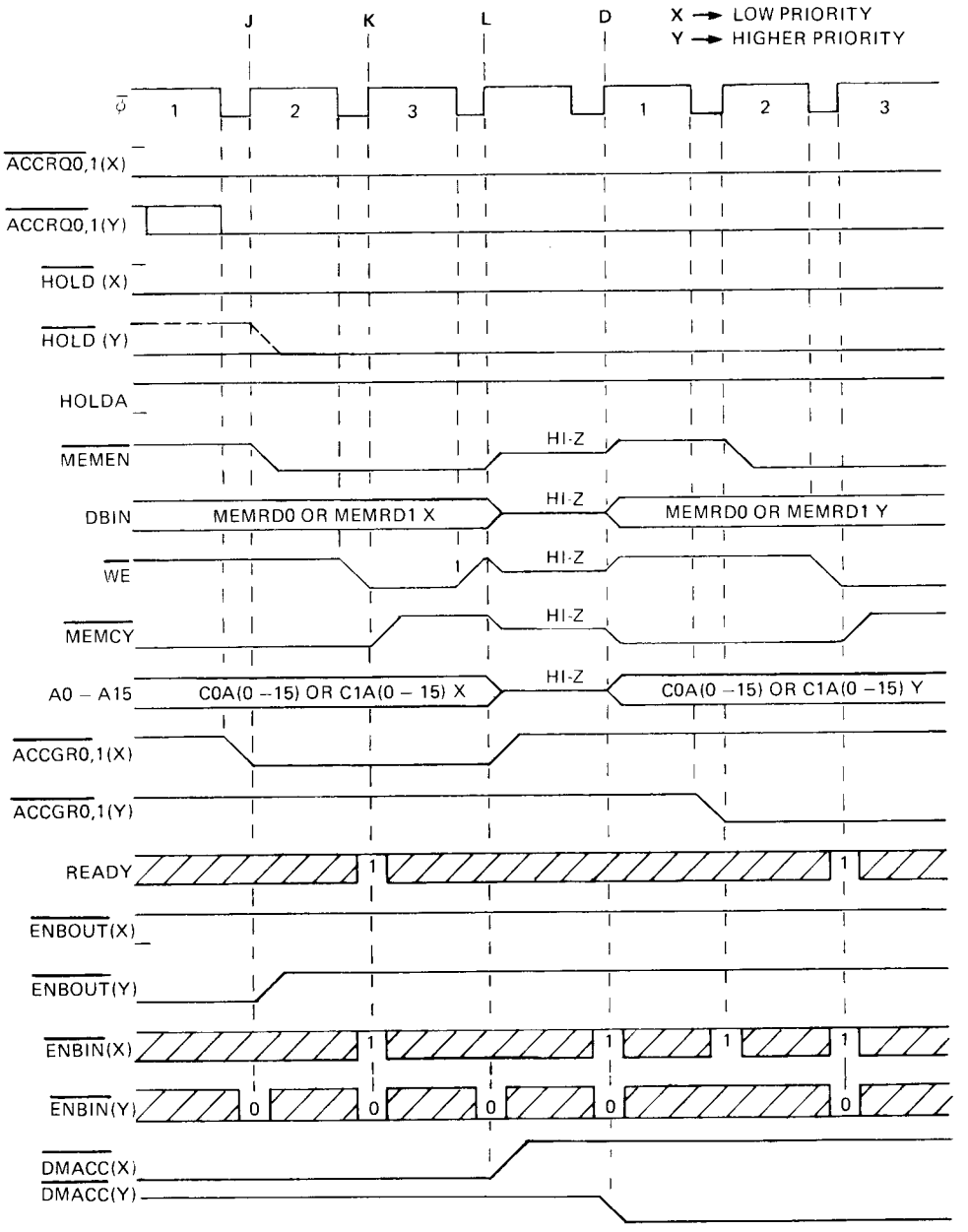


FIGURE 7 – CONTROL SEQUENCE – TRANSFER OF CONTROL FROM DMAC (X) TO DMAC (Y)

When DMAC(Y) receives an $\overline{\text{ACCRQ}}$(Y) from its DMAPD, it issues a $\overline{\text{HOLD}}$ and disables its $\overline{\text{ENBOUT}}$ on the subsequent rising edge of $\bar{\phi}$ (point J). DMAC(X) samples its $\overline{\text{ENBIN}}$ now to be a one at the rising edge of clock cycle 3 (point K), and prepares to place its control signals to their relevant conditions (i.e., memory control signals to HI-Z and others to their respective high or low state). At the end of system clock cycle 3 (point L), DMAC(X), which is temporarily disabled, releases control to DMAC(Y), which takes over control using the $\overline{\text{MEMCY}}$ signal, (delayed for one clock cycle) as an enable (point D). DMAC(X) does not relinquish its $\overline{\text{HOLD}}$ or $\overline{\text{ENBOUT}}$ since it is still contending for access. Therefore, when DMAC(Y) has completed its use of the bus and has relinquished its $\overline{\text{HOLD}}$ and $\overline{\text{ENBOUT}}$, DMAC(X) will reassume control as if executing a normal start from the CPU.

2.2.4 Return Of Control From DMAC To CPU

There are two situations in which the DMAC will stop contending for bus access. The first is when the required block transfer has been completed. This condition only occurs when the value contained in the MAR exactly equals the value contained in the LAR and CNTNU equals 0. Anything other than equality has no effect on OPCMP and therefore will not stop DMAC transfers or cause the DMAC to generate an interrupt. Note that this may cause problems on word transfers when the LAR is incorrectly initialized.

The second condition to halt DMA is when the DMAPD limits the transfer by releasing its $\overline{\text{ACCRQ}}$. The $\overline{\text{ACCRQ}}$ signal must change within 100 ns of the receipt of $\overline{\text{ACCGR}}$ (as shown in Section 4.4) to avoid additional memory cycles. Figure 8 shows the timing relationships that occur as the CPU regains control of the system bus.

During the last memory cycle of the last DMAC contending for access, the $\overline{\text{HOLD}}$ signal is removed by the active channel at the rising edge of $\bar{\phi}$ (point N). $\overline{\text{MEMEN}}$, DBIN, $\overline{\text{WE}}$, $\overline{\text{MEMCY}}$, and A0-A15 go to HI-Z state. Now, the CPU (sampling that $\overline{\text{HOLD}}$ is gone) takes command again. The CPU instructs the DMAC that it has taken control by deactivating HOLDA (point P).

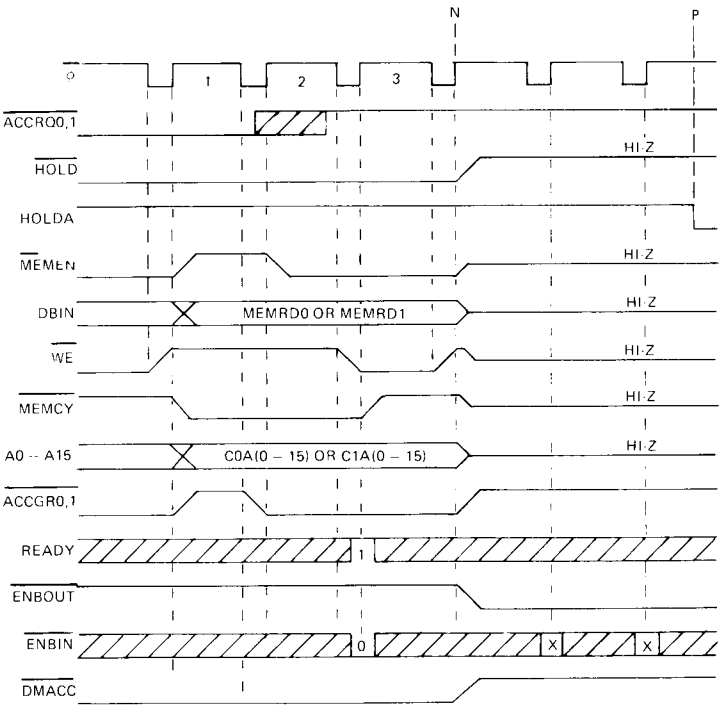


FIGURE 8 – CONTROL SEQUENCE -- DMAC RELINQUISHES COMMAND TO CPU

2.3 CHANNEL PRIORITY AND CHAINING

Within one DMAC, channel 0 has priority over channel 1 when both channels are contending for memory bus access. Priority of channels in different DMAC's is established by the ENBIN/ENBOUT signal string. (See Section 3.1.)

The TMS 9911 contains a chain option which allows both channels to operate under the control of one DMAPD. This chain option may be used to allow the DMAC's two channels to alternately select the block memory locations for data transfer. This function will normally be used when the blocks of data are at noncontiguous memory locations as shown in Figure 9, since a single channel can control contiguous block transfers. The chain controls should be set before either channel is enabled. The ACCRQ0/ACCGR0 handshake pair controls all chain mode transfers, and both channels must be enabled. Since channel 0 will have priority over channel 1, channel 0 will control the addresses and memory control signals that are output until OPCMPO is true. At that point (point R) control will switch to channel 1. Channel 0 must not be re-enabled until the channel 1 operation is complete, or the channel priority convention will cause channel 0 to take control prematurely.

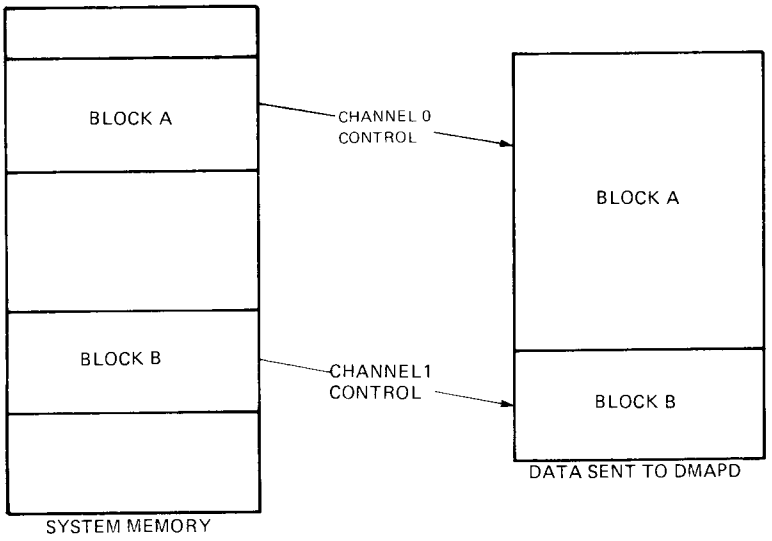


FIGURE 9 – DMAC CHAINING

2.4 INTERRUPT GENERATION

Figure 10 illustrates the logical equivalent of the DMAC interrupt section. The interrupt output ($\overline{INT}$) is active (LOW) whenever either channel interrupt is enabled and that channel's address register is incremented to the last address value. The interrupt signal, once activated, stays active until the CPU writes to bit 21 (OPCMPn).

The TMS 9911 interrupt lines are normally routed to a TMS 9901 Programmable Systems Interface. However, multiple DMAC's may be wire-ORed (by use of open-collector buffers) as a single $\overline{INT}$ input to a TMS 9901 with polling by the host CPU to determine the interrupt source.

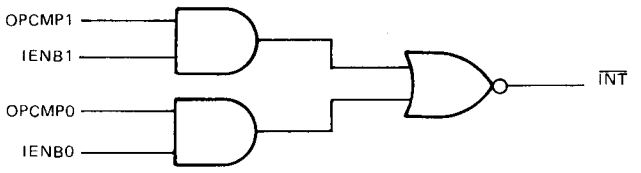


FIGURE 10 – INTERRUPT OUTPUT GENERATION

2.5 POWER-UP CONSIDERATIONS

During hardware reset, $\overline{RESET}$ must be active (LOW) for a minimum of two clock cycles after power is applied to force the TMS 9911 to a known state. $\overline{RESET}$ will disable both DMA channels, set all address registers to zero, set all control bits to their respective inactive state, reset CHSEL to zero, and reset both CnSEL's to zero. The system software must then activate the required status bits and enable any appropriate interrupts. Then when any DMA transfers are required, the memory address register and last address register of the channel involved must be loaded and the channel enabled.

2.6 TERMINAL ASSIGNMENTS

Table 4 defines the TMS 9911 DMAC terminal assignments and the function of each.

TABLE 4
TMS 9911 TERMINAL ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O	DESCRIPTION
			CPU CONTROL
$\overline{HOLD}$	28	I/O	Hold Request. $\overline{HOLD}$ is an open-drain output which becomes active when one of the access request signals is active and corresponding channel is enabled. $\overline{HOLD}$ is Input/Output (only when $\overline{ENBOUT}$ is active) used to set the internal $\overline{HOLD}$ condition according to: HOLD(INT) = ACCRQn · (HOLD + HOLDA)
HOLDA	27	IN	Hold Acknowledge. HOLDA is an input from the CPU in response to the $\overline{HOLD}$ signal which allows the DMAC to access memory.
CRUIN	31	OUT	CRU Data In (to CPU). Serial data output pin from DMAC to CRUIN pin of the CPU.
CRUOUT	30	IN	CRU Data Out (from CPU). Serial data input line to DMAC from CRUOUT pin of the CPU.
CRUCLK	29	IN	CRU Clock. When active (high) and $\overline{CE}$ is low indicates valid data on the CRUOUT line for DMAC input.
$\overline{CE}$	26	IN	Chip Enable. When $\overline{CE}$ is inactive (high), DMAC CRU address decoding is inhibited, preventing execution of any DMAC command function. CRUIN remains at HI-Z when $\overline{CE}$ is inactive.
$\overline{INT}$	25	OUT	Interrupt Output. $\overline{INT}$ is active (low) when either channel has transferred the specified number of data bytes and the interrupt for that channel is enabled.
			MEMORY CONTROL
$\overline{MEMEN}$	36	OUT	Memory Enable. $\overline{MEMEN}$ is at high impedance except when the DMAC is accessing memory. $\overline{MEMEN}$ is low during all except the first clock cycle of each DMAC memory cycle.
DBIN	37	OUT	Memory Data Read Enable. DBIN is at high impedance except when the DMAC is accessing memory. During DMAC memory cycles DBIN indicates the direction of transfer; DBIN = 1 for memory reads and DBIN = 0 for memory writes.
$\overline{WE}$	38	OUT	Write Enable. $\overline{WE}$ is at high impedance except when the DMAC is accessing memory. $\overline{WE}$ performs the function of strobing write data from the data bus. Its timing is identical to that of the $\overline{WE}$ output of the 9900 family CPU's. Since the DMAC is always writing to either memory or DMA peripheral device, $\overline{WE}$ is produced in every DMAC memory cycle.
READY	34	IN	Memory Transfer Ready. READY is sampled at the beginning of clock cycle 3 during each DMAC memory cycle. If READY = 0, the memory cycle is extended until READY = 1, at which time the cycle continues to completion.
$\overline{MEMCY}$	35	I/O	Memory Cycle. $\overline{MEMCY}$ is at high impedance except when the DMAC is accessing memory. $\overline{MEMCY}$ is low during all except the last clock cycle of each DMAC memory cycle. As an input $\overline{MEMCY}$ is used to avoid bus conflicts during DMAC to DMAC control transfer.
			DEVICE CONTROL
$\overline{ACCRQ0}$	1	IN	DMA Device 0 Access Request. $\overline{ACCRQ0}$ is asserted by the DMA device connected to channel 0 when it wishes to access memory.
$\overline{ACCGR0}$	2	OUT	DMA Device 0 Access Granted. $\overline{ACCGR0}$ is active (low), except for the first clock cycle of each memory cycle, while the DMAC is performing a data transfer involving the DMA device connected to channel 0.

DUAL-IN-LINE PACKAGE
(TOP VIEW)

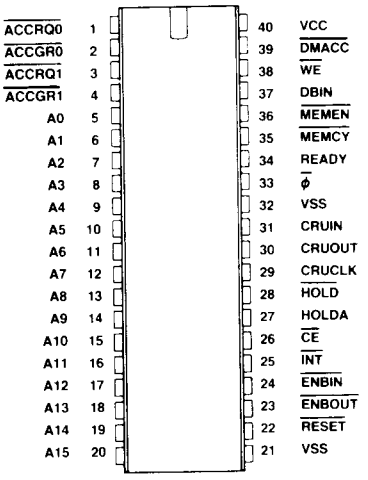


TABLE 4
TMS 9911 TERMINAL ASSIGNMENTS AND FUNCTIONS (CONCLUDED)

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{\text{ACCRQ1}}$	3	IN	DMA Device 1 Access Request. $\overline{\text{ACCRQ1}}$ is asserted by the DMA device connected to channel 1 when it wishes to access memory.
$\overline{\text{ACCGR1}}$	4	OUT	DMA Device 1 Access Granted. $\overline{\text{ACCGR1}}$ is active (low), except for the first clock cycle of each memory cycle, while the DMAC is performing a data transfer involving the DMA device connected to channel 1.
$\overline{\text{DMACC}}$	39	OUT	DMA Accessing Memory, $\overline{\text{DMACC}}$ is active (low when the DMAC is active. DMACC is de active (High) when the DMAC is not accessing memory, DMACC can be used to control buffers when used. (See Section 3.4.)
			DMA CONTROL
$\overline{\text{ENBIN}}$	24	IN	Enable Input. When $\overline{\text{ENBIN}}$ = 0, the DMAC is allowed to contend for bus access. When $\overline{\text{ENBIN}}$ = 1, the DMAC is inhibited from contending for bus access.
$\overline{\text{ENBOUT}}$	23	OUT	Enable Output. $\overline{\text{ENBOUT}}$ is active (low) when the DMAC is not contending for bus access. $\overline{\text{ENBOUT}}$ is inactive (high) when the DMAC is contending for or has control of the bus.
			ADDRESS BUS
A0 (MSB)	5	OUT	Address Bus. A0-A15 outputs are at high impedance while the DMAC is not accessing memory. During memory access A0-A15 output the memory address. A10-A14 are inputs selecting the address of the bit to or from which the CPU is transferring data via the CRU. A15 is not normally implemented on 9900 family systems, but is needed to specify which byte of a word is being transferred during byte structured data transfer.
A1	6	OUT	
A2	7	OUT	
A3	8	OUT	
A4	9	OUT	
A5	10	OUT	
A6	11	OUT	
A7	12	OUT	
A8	13	OUT	
A9	14	OUT	
A10	15	I/O	
A11	16	I/O	
A12	17	I/O	
A13	18	I/O	
A14	19	I/O	
A15 (LSB)	20	OUT	
			CLOCK SIGNALS
$\overline{\phi}$	33	IN	Clock Input. Provided by the $\overline{\phi 1}$ output of the TMS 9904 clock generator or the $\overline{\text{CKOUT}}$ output of the TMS 9980A or 9981 (in which case, $\overline{\text{HOLD}}$ should be synchronized to avoid changing during $\phi 1$).
			DEVICE RESET
$\overline{\text{RESET}}$	22	IN	Device Reset. When $\overline{\text{RESET}}$ is active (low), the DMAC is reset to a known state where both channels are disabled.
			POWER REQUIREMENTS
V_{CC}	40	IN	Supply voltage. 5.0 volts dc ± 5%
V_{SS}	21,32	IN	Voltage Reference. Pin 21 provides ground (O V) reference for the TMS 9911 logic. Pin 32 provides ground reference for all signal buffers.

3. DEVICE APPLICATION

This section discusses some of the design considerations in the use of the TMS 9911 and describes the software interface between the CPU and the DMAC.

3.1 MULTIPLE DMAC's

Figure 11 shows the connection of four DMAC's that are cascaded into a prioritized chain. The highest priority DMAC has its $\overline{\text{ENBIN}}$ input grounded so that it will never be denied access to the bus. The $\overline{\text{ENBOUT}}$ output of this DMAC is connected to the $\overline{\text{ENBIN}}$ input of the second highest priority DMAC. This signal is only active (low) when the first DMAC is not contending for bus access. For third and following DMAC's in the priority chain, the $\overline{\text{ENBIN}}$ input is derived by a gating of both the $\overline{\text{ENBIN}}$ and $\overline{\text{ENBOUT}}$ of the previous DMAC.

Each of these DMAC's is only enabled if the previous DMAC is enabled and not contending for bus access. Due to the external gating, maximum length of the priority chain is limited only by the propagation delay of the external gate. This delay can be as low as 5ns per link in the priority chain. The total delay of the chain must not exceed the clock cycle time, $t_C(\phi)$.

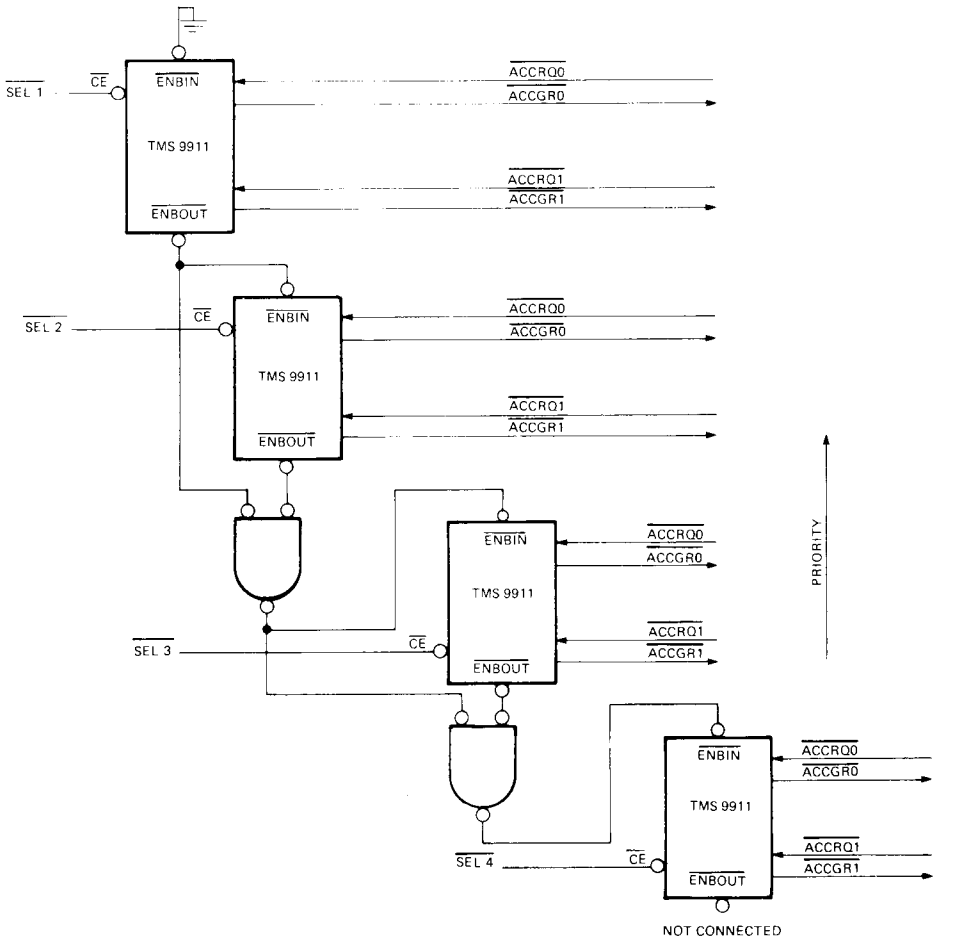


FIGURE 11 – CONNECTION FOR MULTIPLE DMA CONTROLLERS

3.2 CONTROL SIGNALS FOR DEVICES AND MEMORY ON COMMON BUS STRUCTURES

A device which connects to a DMA channel is usually able to decode control lines from both the CPU and DMAC. This section describes the logic to generate the proper control signals for such DMA devices and memory. DMA only devices do not need this extra gating. Generally, implementation of these controls minimize the levels of gating $\overline{WE}$ must pass through.

3.2.1 DBIN Generation For Peripheral Device

The following values occur during CPU controlled data transfers to a peripheral device:

$\overline{ACCGR}$ = $\overline{DMACC}$ = inactive
DBIN = active for reading from the device or memory
DBIN = inactive for writing to the device or memory

The following values occur during DMAC controlled data transfers involving the peripheral device:

$\overline{ACCGR}$ = $\overline{DMACC}$ = active
DBIN = inactive for reading from the device (writing to memory)
DBIN = active for writing to the device (reading from memory)

These values supply memory with the proper DBIN signal for both cases, but do not supply the device with a consistent control signal for direction of drive. Figure 12A shows the logic necessary to supply a consistent signal such that: DBIN = 1 for reading from the device and DBIN = 0 for writing to the device. This conforms to the CPU-generated values. The TMS 9909 Floppy Disk Controller and TMS 9914 General Purpose Interface Bus Adaptor already contain internal gating to perform this function. TMS 9909 and TMS 9914 DBIN and $\overline{WE}$ inputs should be connected directly to the CPU/DMAC bus signals.

3.2.2 $\overline{WE}$ Generation For Memories And Devices

Since either the memory or the peripheral device is being written to, a $\overline{WE}$ pulse is output by the TMS 9911 on every DMA data transfer. For memories which require $\overline{WE}$ to remain inactive (high) during memory reads, Figure 12B shows the external logic to block the unwanted signal. For devices which require $\overline{WE}$ to remain inactive (high) during device reads, the same logic must be applied, which is shown in Figure 12C as a combination of Figures 12A and B. Again, the TMS 9909 and 9914 already contain internal gating to perform this function.

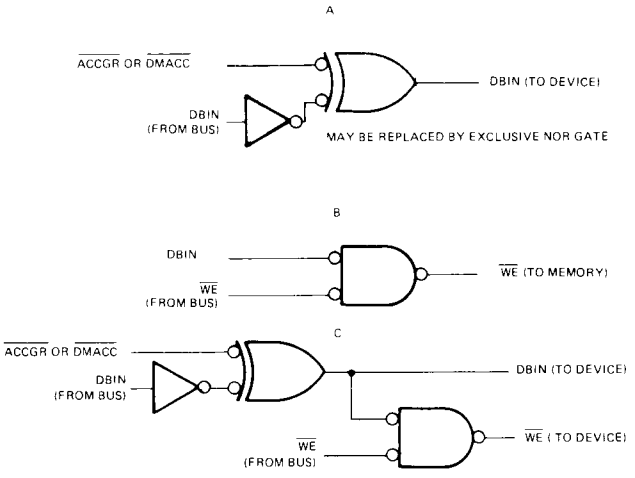


FIGURE 12 – EXTERNAL CONTROL SIGNAL LOGIC

3.3 DATA TRANSFER RATES

The fastest data transfer rate occurs when the TMS 9911 DMAC gains control of the bus and transfers a block of data as rapidly as memory will allow. This minimum data transfer period is:

$$t_{min} = (3 + W_{DMA})\, t_c(\phi)$$

where W_{DMA} is the number of wait states per memory cycle under DMAC control.

The fastest guaranteed rate that the DMAC can gain bus access, transfer a single byte or word, and release the bus, depends on the DMA transfer time plus the maximum delay between activation of $\overline{HOLD}$ and receipt of HOLDA. This maximum data transfer period is:

$$t_{max} = (11 + W_{DMA} + 3\,W_{CPU})\, t_c(\phi)$$

where W_{CPU} is the number of wait states per memory cycle under CPU control.

Rates between the two above are possible only if the DMAC does not release control of the bus back to the CPU between transfers. Figure 13 shows logic to keep $\overline{HOLD}$ active during slow block transfers which allow transfer rates with a period equal to any multiple of $t_c(\phi)$ between t_{min} and t_{max}. Extreme caution must be exercised to ensure that the CPU is not indefinitely halted (i.e, IENB must be set). Figure 14 shows the allowable transfer rates. For a clock frequency of 3 MHz and no wait states, this latch will be required for transfer rates betweeen 272 kHz and the maximum transfer rate of 1 MHz.

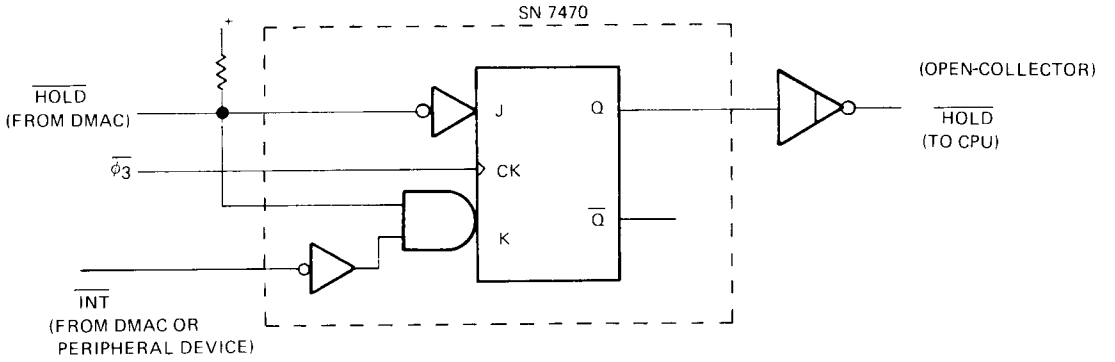


FIGURE 13 – FLIP - FLOP INTERCONNECT FOR SLOW BLOCK TRANSFER

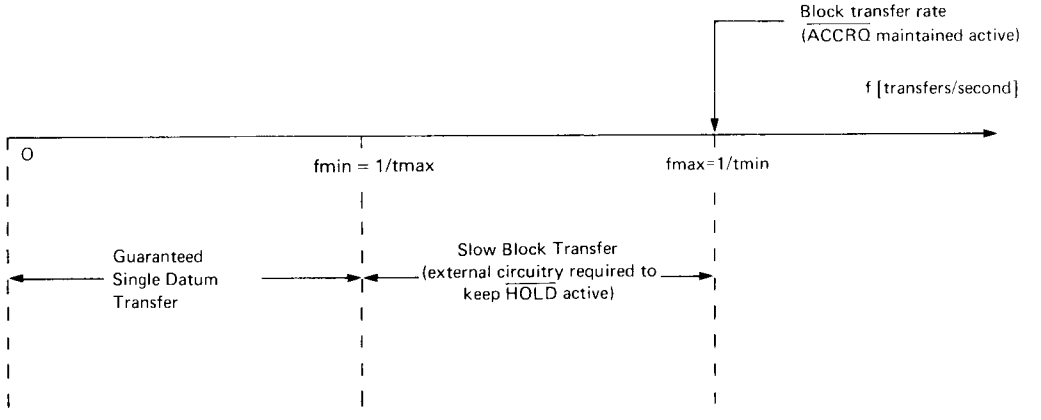


FIGURE 14 – ALLOWABLE DATA TRANSFER RATES

3.4 BUFFERING TMS 9911 SIGNALS

When the TMS 9911 is used in systems involving signal buffering (as in board-to-board signal transfer), signal direction and conditions must be observed by the buffers. Table 5 characterizes the DMAC signals in terms of direction of drive and signal conditions (depicted in parentheses). Conditional signals must be present only when valid to avoid bus conflict.

TABLE 5
DMAC SIGNAL DIRECTIONS

INPUT (CONDITION)	OUTPUT (CONDITION)
$\overline{\text{ACCRQ0}}$	$\overline{\text{ACCGR0}}$
$\overline{\text{ACCRQ1}}$	$\overline{\text{ACCGR1}}$
READY	A0-A9 ($\overline{\text{DMACC}}$ = Active)
$\overline{\phi}$	A15 ($\overline{\text{DMACC}}$ = Active)
HOLDA	$\overline{\text{DMACC}}$
$\overline{\text{CE}}$	$\overline{\text{WE}}$ ($\overline{\text{DMACC}}$ = Active)
$\overline{\text{ENBIN}}$	DBIN ($\overline{\text{DMACC}}$ = Active)
$\overline{\text{RESET}}$	$\overline{\text{MEMEN}}$ ($\overline{\text{DMACC}}$ = Active)
CRUCLK	CRUIN ($\overline{\text{CE}}$ = Active)
CRUOUT	$\overline{\text{INT}}$
A10-A14 ($\overline{\text{DMACC}}$ = Inactive)	$\overline{\text{ENBOUT}}$
$\overline{\text{MEMCY}}$ ($\overline{\text{DMACC}}$ = Inactive)	A10-A14 ($\overline{\text{DMACC}}$ = Active)
$\overline{\text{HOLD}}$ ($\overline{\text{ENBOUT}}$ = Active)	$\overline{\text{MEMCY}}$ ($\overline{\text{DMACC}}$ = Active)
	$\overline{\text{HOLD}}$ ($\overline{\text{ENBOUT}}$ = Inactive)

Figure 15 shows a typical buffer arrangement for use with the DMAC and a memory mapped peripheral. Some of this gating can be eliminated for a DMA only device. The proper $\overline{\text{WE}}$ (Section 3.2.2) and $\overline{\text{ENBOUT}}$ (Section 3.1) signals are generated and passed to the bus. Discrete open collector logic is used to buffer $\overline{\text{HOLD}}$ using $\overline{\text{ENBOUT}}$ as a directional control. The other bi-directional signals (A10-A14 and $\overline{\text{MEMCY}}$) are buffered with a bi-directional buffer. Conditional outputs are buffered using tristate buffers enabled only when the condition is valid. The $\overline{\text{ACCGR}}$n, $\overline{\text{ACCRQ}}$n, and $\overline{\text{CE}}$ signals are assumed to be of local interest, so are not buffered. The remaining signals are buffered using uni-directional buffers in the corresponding direction.

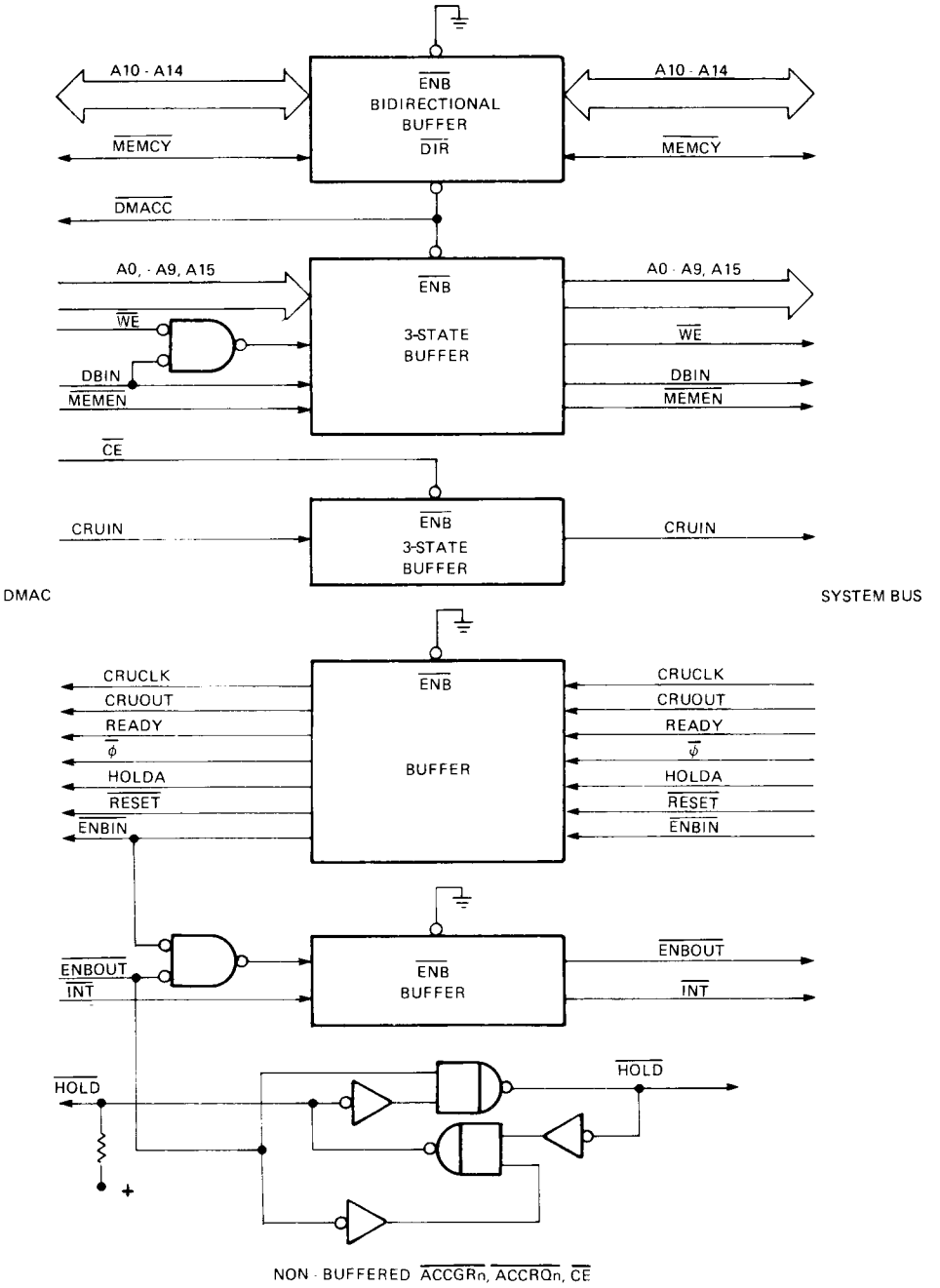


FIGURE 15 – OFF - BOARD SIGNAL BUFFERING

3.5 SOFTWARE INTERFACE

Figure 16 lists a sample portion of TMS 9900 Assembly Language used to initialize the TMS 9911 DMAC. This example sets up channel 1 of the DMAC for a block transfer from memory to the DMA peripheral device connected to channel 1. The interrupt output from the DMAC is also enabled so that the DMAC will interrupt the processor once the block has been transferred assuming that the interrupt interface and the processor interrupt mask are properly set. The sample interrupt service routine simply resets the interrupt without disturbing the current CHSEL status and sets a done flag. Both routines assume that the DMAC is connected to bits $E0_{16}$-FF_{16} of the CRU address space.

```
                * ABSTRACT:
                *       ROUTINE TO INTIALIZE CHANNEL 1 OF TMS 9911 DMAC FOR
                *       DATA TRANSFER.
                *
                * CALLING SEQUENCE:
                *       R1,R2(BLOCK START ADDRESS,NUMBER OF WORDS)
                *       BL    INIT                CALLING METHOD
                *       B     *R11                RETURN METHOD
                *
                * EXTERNAL REFERENCES:
                *       NONE
                *
                * LOCAL DATA:
                *       NONE
                *
                * ENTRY POINT:
     0000'  INIT    EQU   $           ***** ENTRY *****
                    DEF   INIT
0000 020C           LI    R12,>1C0            LOAD CRU SOFTWARE BASE ADDRESS
0002 01C0
                *                               (HARDWARE BASE ADDRESS >E0)
0004 1D1F           SBO   31                  RESET DMAC
0006 1D19           SBO   25                  SELECT CHANNEL 1
0008 1D14           SBO   20                  SELECT WORD TRANSFER
000A 1E13           SBZ   19                  SELECT MEMRD=0 (MEMORY WRITE)
000C A082           A     R2,R2               COMPUTE BYTES TO TRANSFER
000E A081           A     R1,R2               CALCUTATE LAR CONTENTS
0010 3002           LDCR  R2,0                LOAD LAR
0012 1D10           SBO   16                  LOAD MAR
0014 3001           LDCR  R1,0
0016 1D16           SBO   22                  ENABLE INTERRUPTS
0018 1D11           SBO   17                  ENABLE CHANNEL 1
001A 045B           B     *R11                RETURN
```

FIGURE 16 (SHEET 1 OF 2) — SOFTWARE INTERFACE

```
                * ABSTRACT:
                *       ROUTINE TO SET DONE FLAG AFTER BLOCK TRANSFER HAS
                *       COMPLETED.
                *
                * CALLING SEQUENCE:
                *       NO PARAMETERS PASSED
                *       ENTRY METHOD IS BY INTERRUPT VECTOR
                *       RTWP                        RETURN METHOD
                *
                * EXTERNAL DATA:
                *       REF  DONE                   DONE FLAG
                *
                * LOCAL DATA:
                *       NONE
                *
                * ENTRY POINT:
     0000'  INTSVC EQU  $          ***** ENTRY *****
                   DEF  INTSVC
0000 020C          LI   R12,>1C0            LOAD CRU SOFTWARE BASE ADDRESS
0002 01C0
0004 1D19          SBO  25                  SELECT CHANNEL 1
0006 1E15          SBZ  21                  RESET INTERRUPT
0008 0720          SETO @DONE               SET DONE FLAG
000A 0000
000C 1D18          SBO  24                  RESTORE CHANNEL NUMBER
000E 0380          RTWP                     RETURN
```

FIGURE 16 (SHEET 2 OF 2) — SOFTWARE INTERFACE

3.6 TYPICAL TMS 9911 DMAC APPLICATIONS

The following are some specific TMS 9911 DMAC applications that can be used as examples of typical DMAC interfaces. The designs shown are not entirely complete to simplify the DMAC interface description and increase the generality of the designs.

3.6.1 TMS 9900 Floppy Disk System

Figure 17 shows the hardware interface required to produce a DMA channel between a TMS 9909 floppy disk controller and a TMS 9900 CPU with memory. Both the TMS 9909 and 9911 require address decode circuitry to select them for initialization and control from the CPU. Interrupt logic is required to supply the CPU with an interrupt request. This logic could be a TMS 9901 Programmable Systems Interface. The TMS 9904 Clock Generator supplies all the necessary clock signals as well as a hardware reset (this has been omitted on subsequent designs for clarity).

One external gate is required to inhibit the $\overline{WE}$ signal to memory during memory read transfers under DMAC control.

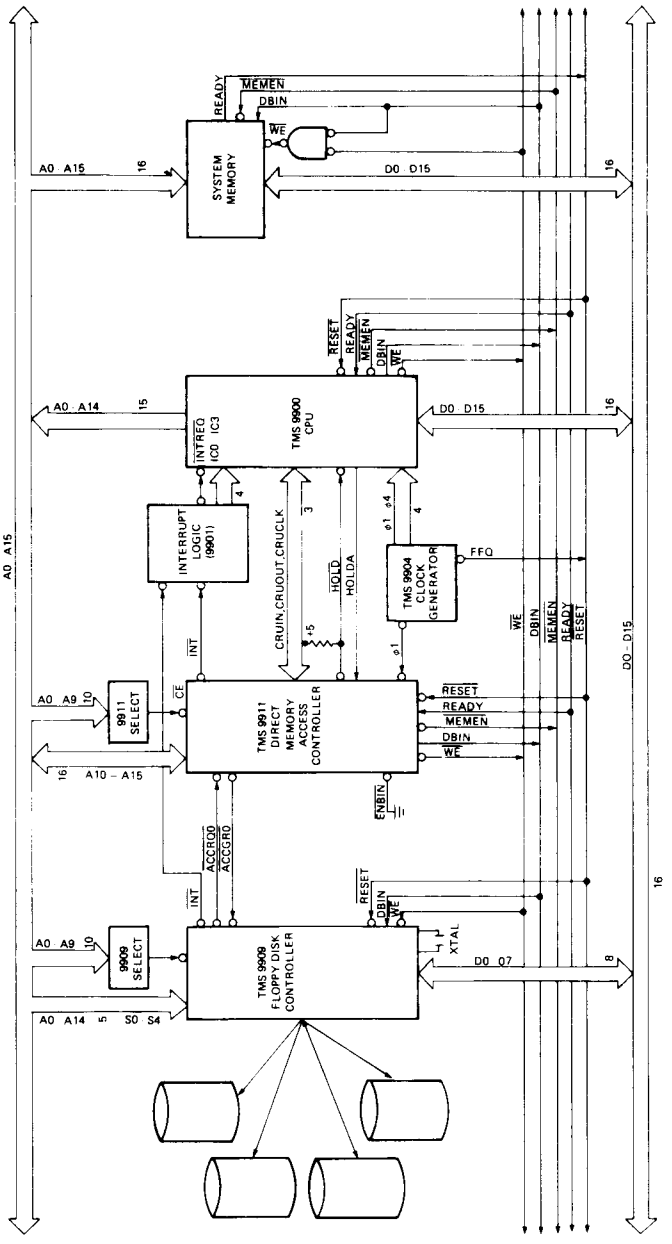


FIGURE 17 – TMS 9900 FLOPPY DISK SYSTEM WITH DMA

3.6.2 TMS 9900 Multiprocessor System

Figure 18 shows a method by which one TMS 9900 (CPU 1) can communicate with another TMS 9900 (CPU 2) via a DMA channel. This configuration requires a TMS 9901 Programmable Systems Interface (PSI) which is used to output a 16-bit address to a synchronous 16-bit counter as well as perform interrupt functions. The counter supplies the transfer address to the slave memory via a 16-bit buffer to isolate it from the slave bus during non-DMA transfers. A 16-bit bi-directional buffer is used to isolate the two system data buses during non-DMA transfers and provide directional control during DMA data transfers. Other logic is required to inhibit $\overline{WE}$ during memory reads and to provide the proper combined HOLDA and READY signal to the DMAC.

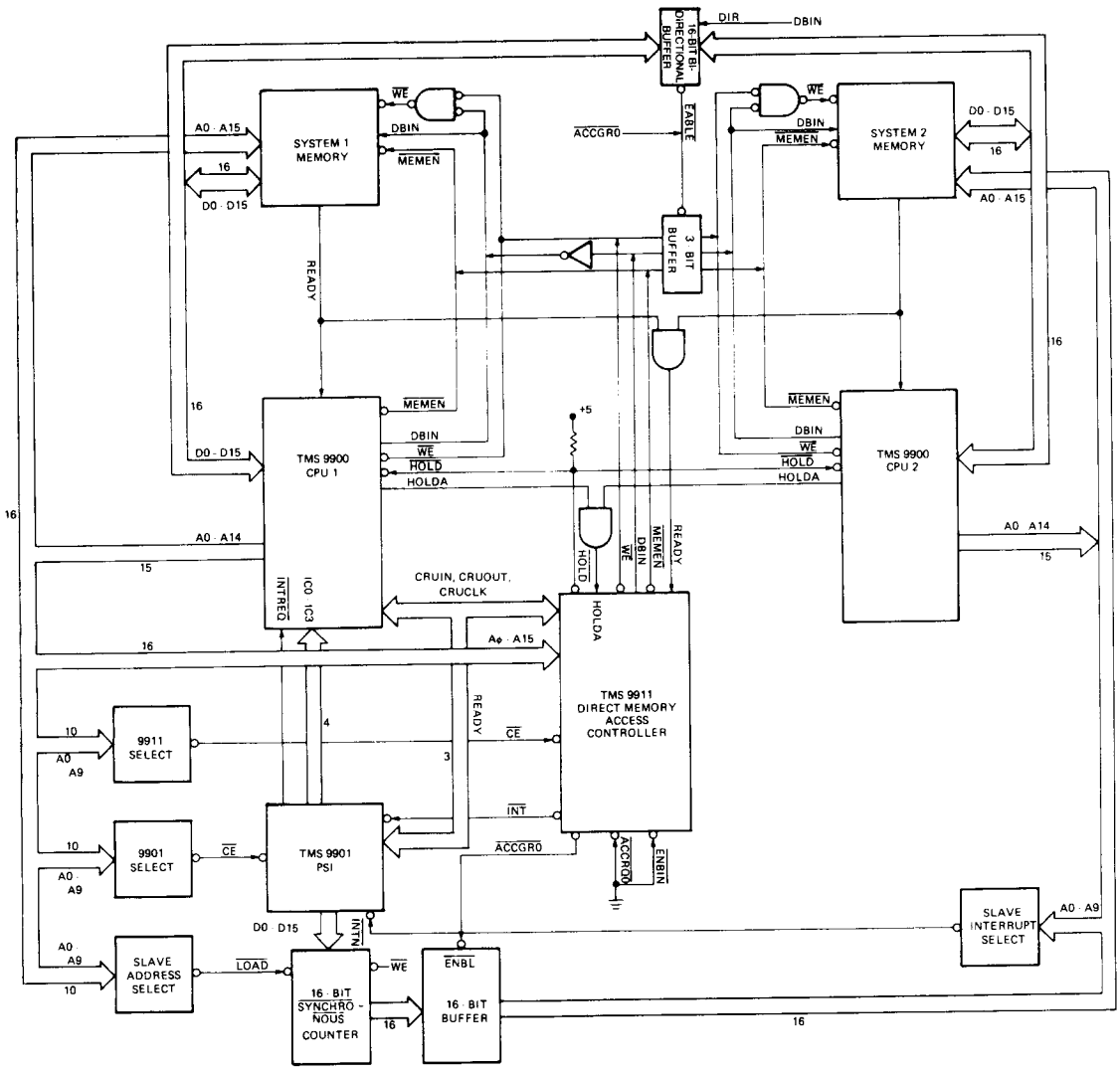


FIGURE 18 – TMS 9900 MULTIPROCESSING SYSTEM USING DMA LINK

Address decode circuitry is required to select the DMAC, PSI, and CPU 2 memory address counter by CPU 1. Another address decoder is used by CPU 2 to generate an interrupt input to CPU 1 via the TMS 9901. Transfers are accomplished by loading the starting memory 2 address into the 16-bit counter via the TMS 9901 and loading the memory 1 starting address and last address into the DMAC, which will then acquire control of both buses when enabled and transfer the entire block of memory. This concept could be expanded by use of multiplexers to connect more than two CPU's.

3.6.3 TMS 9927 Video Controller System

This application uses DMA to write and read data from a dual port RAM associated with the TMS 9927 Video Controller during its vertical sync period. Figure 19 shows how a TMS 9901 PSI is used to produce the interrupt request and associated vector. Address decoding is required to select the DMAC and PSI for CPU command communication via the CRU interface. Other logic is required to provide the proper directional control for the dual port RAM and inhibit $\overline{WE}$ during memory reads. This application differs from the previous memory-to-memory DMA channel in two important ways. First, there is only one system bus included, so no signal buffers are required. Second, there is no counter to supply memory addresses to the dual port RAM. Since the DMAC supplies memory addresses to both memories simultaneously, the 10 least significant bits of the system memory address must correspond to the dual port RAM address. This modification restricts the system software somewhat, but reduces the hardware required to implement it.

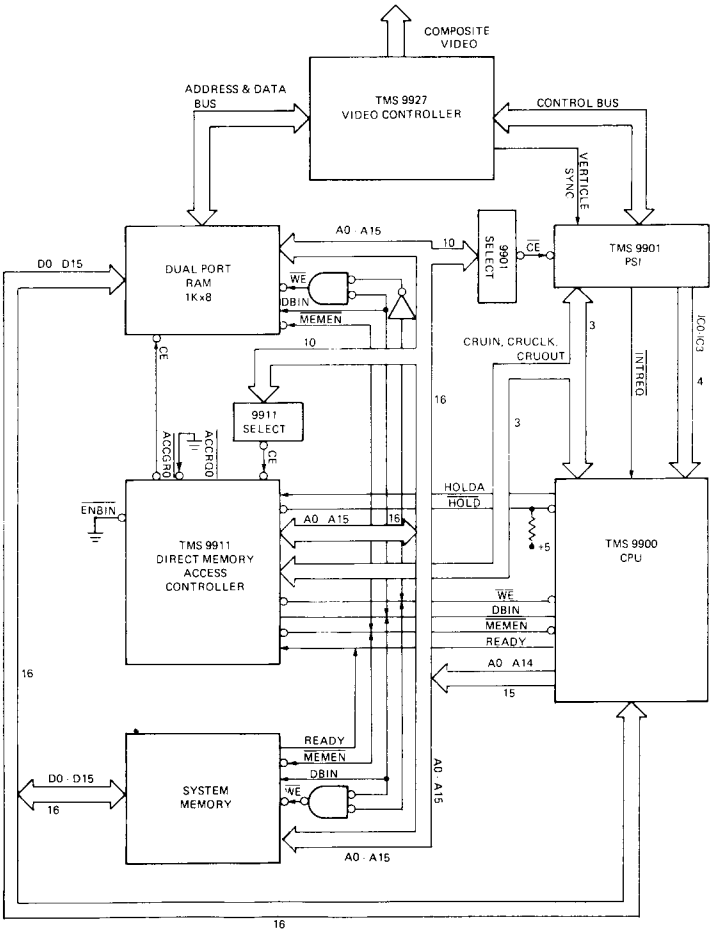


FIGURE 19 – TMS 9900/9927 VIDEO CONTROLLER WITH DMA LINK

3.6.4 TMS 9900 Data Acquisition/Transmission System

Figure 20 illustrates how a high speed analog-to-digital converter and digital-to-analog converter can be connected to a TMS 9900 CPU via a DMA channel. This system requires a clock to time when the conversions should occur and buffers to isolate the DAC and ADC from the system bus when not accessing memory. Gating is required to inhibit $\overline{WE}$ to memory during memory reads and to latch $\overline{HOLD}$ for slow block transfers if necessary (see Section 3.3). Address decode is required to select the DMAC and some type of interrupt logic (such as a TMS 9901 PSI) is required between the DMAC and CPU. The ADC and DAC can be of any precision up to 16 bits and need not be of the same precision, depending on the software and application involved.

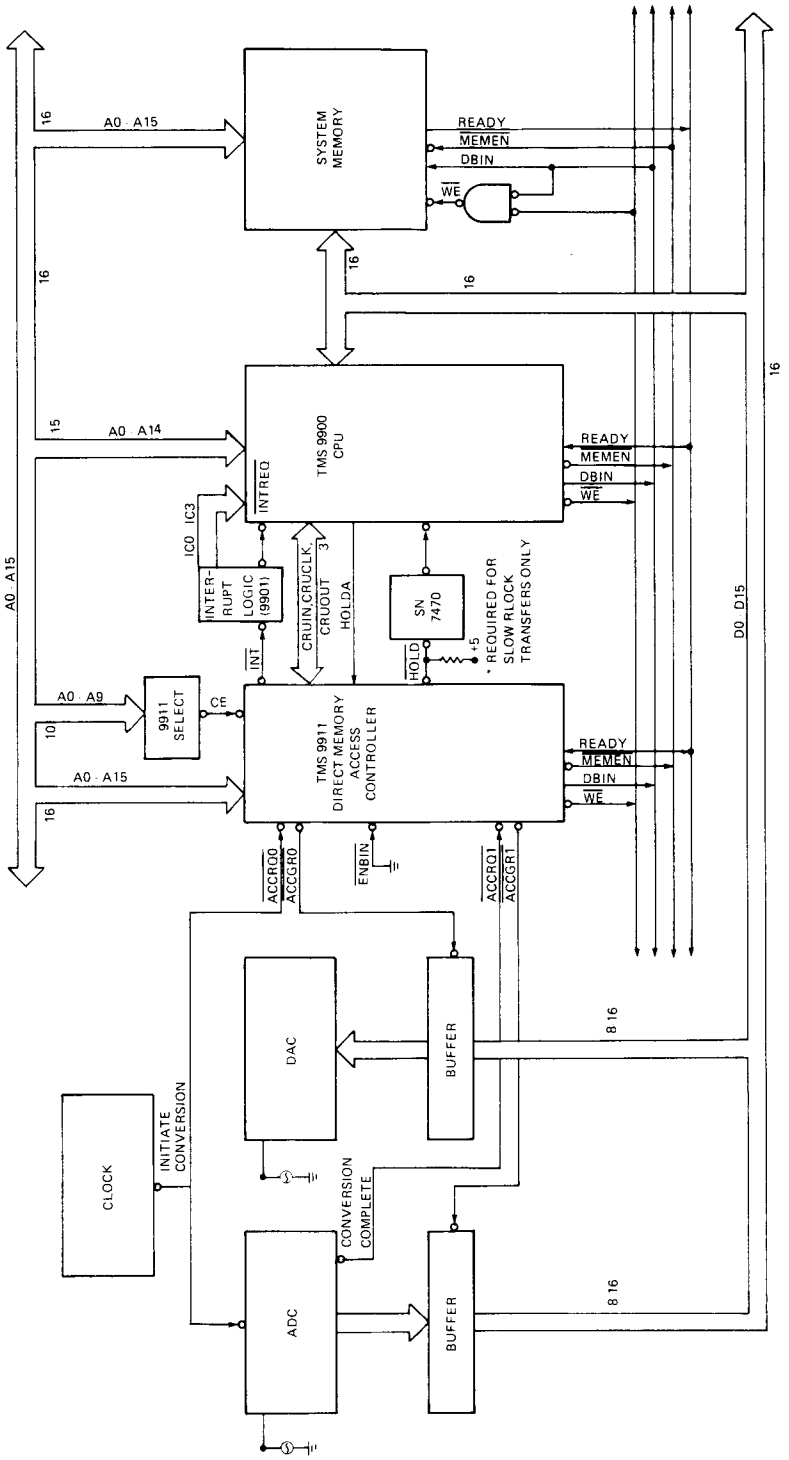


FIGURE 20 — TMS 9900 DMA DATA ACQUISITION SYSTEM

4. TMS 9911 ELECTRICAL SPECIFICATIONS

4.1 TMS 9911 SPECIFICATION

4.1.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1) ..... −0.3 V to 10 V
All input and output voltages ..... −0.3 V to 10 V
Continuous power dissipation ..... 0.65 W
Operating free-air temperature range ..... 0°C to 70°C
Storage temperature range ..... −65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operations of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

NOTE 1: Voltage values are with respect to V_{SS}.

4.1.2 RECOMMENDED OPERATING CONDITIONS

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2.2		$V_{CC}+1$	V
Low-level input voltage, V_{IL}	$V_{SS}-0.3$		0.8	V
Operating free-air temperature, T_A	0		70	°C

4.1.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (UNLESS OTHERWISE NOTED)

	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
I_I	Input current	$V_I = -1$ to $V_{CC}+1$		10	μA
V_{OH}	High-level output voltage	$I_{OH} = -0.4$ mA	2.4		V
V_{OL}	Low-level output voltage	$I_{OL} = 3.2$ mA		0.4	V
$I_{CC(AV)}$	Supply current from V_{CC} Average			140	mA
C_i	Capacitance, any input			15	pf

*Except HOLD which is open drain

4.1.4 TIMING REQUIREMENTS OVER FULL RANGE OF OPERATING CONDITIONS

PARAMETER		MIN	TYP	MAX	UNITS
$t_{c(\phi)}$	Clock cycle time	300	333	667	ns
$t_{r(\phi)}$	Clock rise time	5		40	ns
$t_{f(\phi)}$	Clock fall time	10		40	ns
$t_{w(\phi L)}$	Clock pulse width (low level)	45		300	ns
$t_{w(\phi H)}$	Clock pulse width (high level)	225			ns
$t_{w(CC)}$	CRUCLK pulse width	100	185		ns
t_{su2}	Setup time for A10-A14 and CRUOUT before CRUCLK	180			ns
t_{su1}	Setup time for CE before CRUCLK	150			ns
t_{h1}	Hold time for CE, A10-A14, or CRUOUT after CRUCLK	60			ns
$t_{su\phi}$	Setup time for Ready and ENBIN before ϕ	50			ns
t_{h2}	Hold time for Ready and ENBIN after ϕ	30			ns
t_d	Delay time for ACCRQn after ACCGRn			115	ns

4.1.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{pd1}	Propagation delay, CE to valid CRUIN	C_L = 100 pF			300	ns
t_{pd2}	Propagation delay, A10-A14 to valid CRUIN	C_L = 100 pF			320	ns
t_{PLH},t_{PHL}	Propagation delay,	C_L = 100 pF			180	ns
	low to high, high to low	C_L = 50 pF			115	ns
t_{PZL},t_{PZH}	Propagation delay, HI-Z to active	C_L = 50 pF			100	ns
t_{PLZ},t_{PHZ}	Propagation delay, active to HI-Z	C_L = 50 pF			115	ns

*All switching characteristics assume 12ns maximum $t_{r(\phi)}$ and $t_{f(\phi)}$.

4.2 TMS 9911-40 SPECIFICATIONS

4.2.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1) ..... −0.3 V to 10 V
All input and output voltages ..... −0.3 V to 10 V
Continuous power dissipation ..... 0.70 W
Operating free-air temperature range ..... 0°C to 70°C
Storage temperature range ..... −65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operations of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

NOTE 1: Voltage values are with respect to V_{SS}.

4.2.2 RECOMMENDED OPERATING CONDITIONS

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2.2		$V_{CC}+1$	V
Low-level input voltage, V_{IL}	$V_{SS}-0.3$		0.8	V
Operating free-air temperature, T_A	0		70	°C

4.2.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (UNLESS OTHERWISE NOTED)

	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
I_I	Input current	$V_I = -1$ to $V_{CC}+1$		10	μA
V_{OH}	High-level output voltage	$I_{OH} = -0.4$ mA	2.4		V
V_{OL}	Low-level output voltage	$I_{OL} = 3.2$ mA		0.4	V
$I_{CC(AV)}$	Supply current from V_{CC}, Average			140	mA
C_i	Capacitance, any input			15	pf

*Except HOLD which is open drain

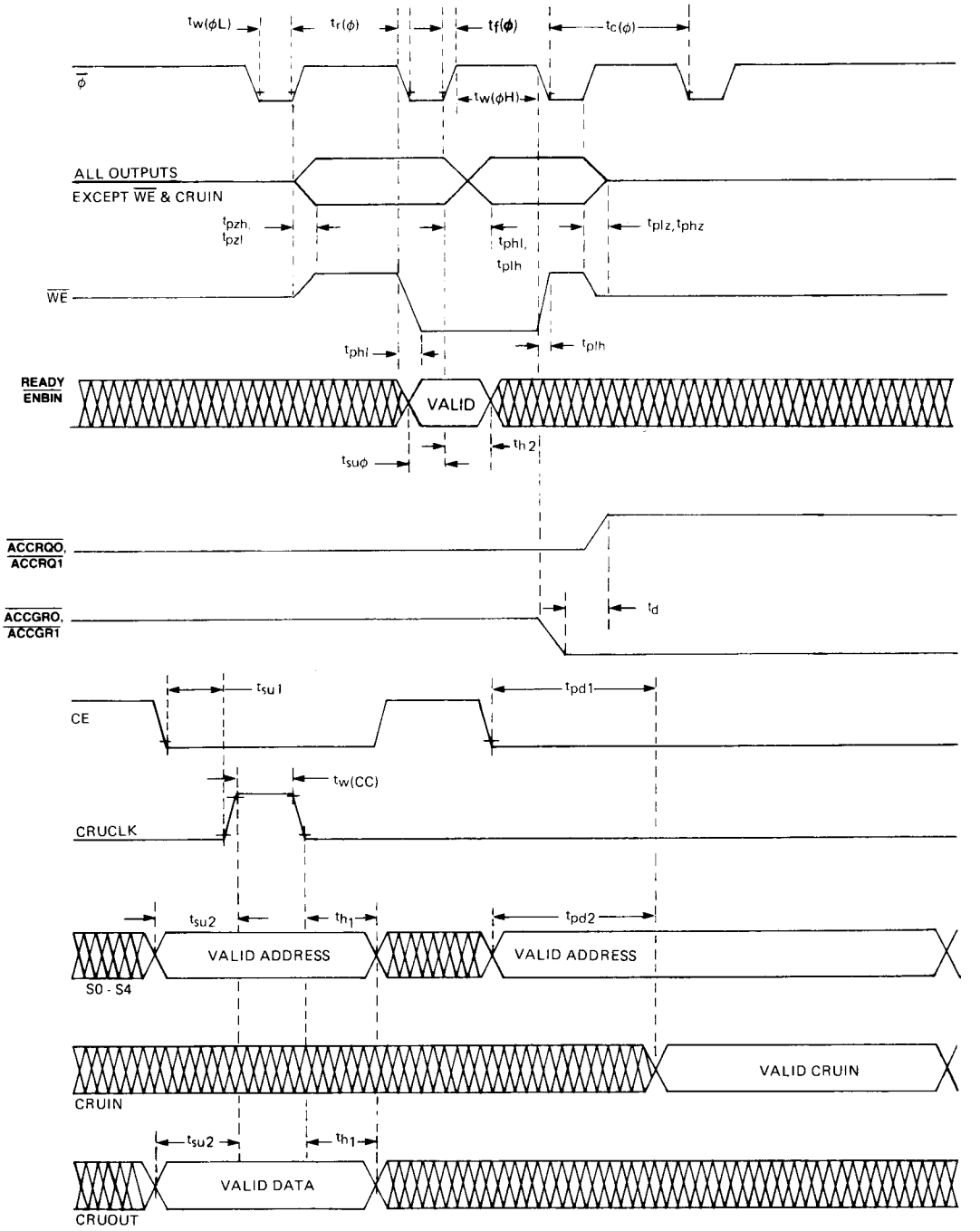
4.2.4 TIMING REQUIREMENTS OVER FULL RANGE OF OPERATING CONDITIONS

	PARAMETER	MIN	TYP	MAX	UNITS
$t_{c(\phi)}$	Clock cycle time	240	250	360	ns
$t_{r(\phi)}$	Clock rise time	5		40	ns
$t_{f(\phi)}$	Clock fall time	10		40	ns
$t_{w(\phi L)}$	Clock pulse width (low level)	40		300	ns
$t_{w(\phi H)}$	Clock pulse width (high level)	180			ns
$t_{w(CC)}$	CRUCLK pulse width	80	185		ns
t_{su2}	Setup time for A10-A14 and CRUOUT before CRUCLK	150			ns
t_{su1}	Setup time for CE before CRUCLK	110			ns
t_{h1}	Hold time for CE, A10-A14, or CRUOUT after CRUCLK	50			ns
$t_{su\phi}$	Setup time for Ready and ENBIN before ϕ	40			ns
t_{h2}	Hold time for Ready and ENBIN after ϕ	20			ns
t_d	Delay time for ACCRQn after ACCGRn			100	ns

4.2.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS

	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{pd1}	Propagation delay, CE to valid CRUIN	C_L = 100 pF			220	ns
t_{pd2}	Propagation delay, A10-A14 to valid CRUIN	C_L = 100 pF			280	ns
t_{PLH},t_{PHL}	Propagation delay,	C_L = 100 pF			160	ns
	low to high, high to low	C_L = 50 pF			100	ns
t_{PZL},t_{PZH}	Propagation delay, HI-Z to active	C_L = 50 pF			100	ns
t_{PLZ},t_{PHZ}	Propagation delay, active to HI-Z	C_L = 50 pF			100	ns

*All switching characteristics assume 12ns maximum $t_{r(\phi)}$ and $t_{f(\phi)}$.



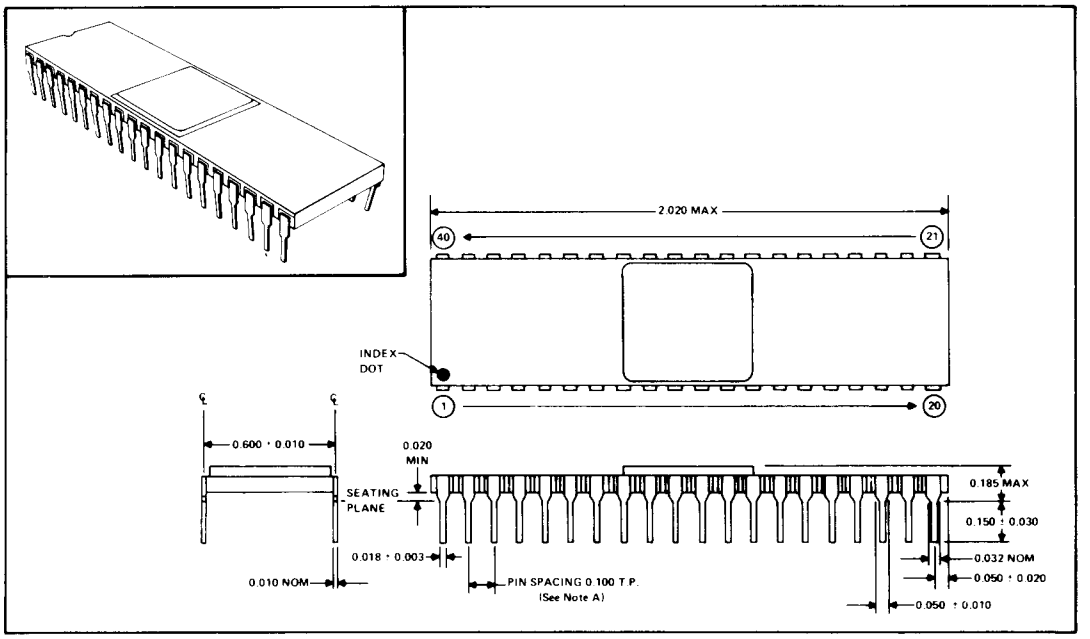
NOTE 1: ALL TIMING MEASUREMENTS ARE FROM 10% AND 90% POINTS.

FIGURE 21 — SWITCHING CHARACTERISTICS

5. MECHANICAL DATA

5.1 TMS 9911JL — 40-PIN CERAMIC PACKAGE

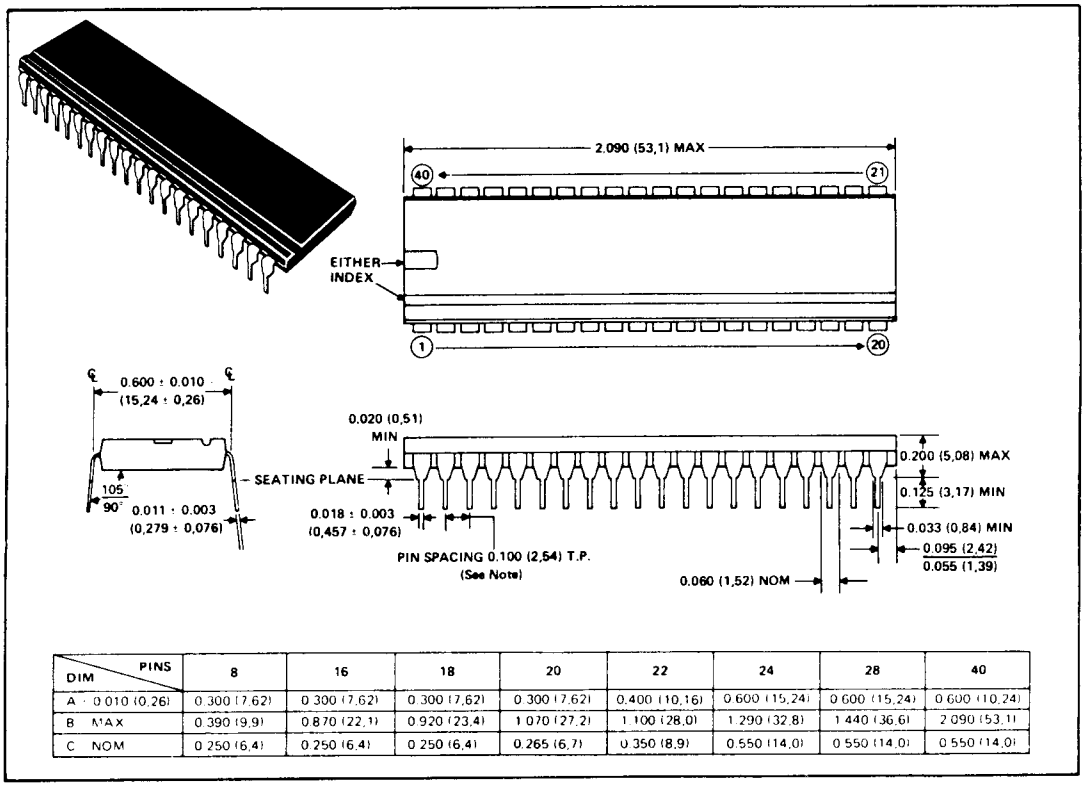
ceramic packages with side-brazed leads and metal or epoxy or glass lid seal



NOTES: A. Each pin centerline is located within 0.010 of its true longitudinal position.
B. All linear dimensions are in inches.

5.2 TMS 9911NL — 40-PIN PLASTIC PACKAGE

plastic packages



DIM \ PINS	8	16	18	20	22	24	28	40
A ± 0.010 (0.26)	0.300 (7.62)	0.300 (7.62)	0.300 (7,62)	0.300 (7.62)	0.400 (10,16)	0.600 (15,24)	0.600 (15,24)	0.600 (10,24)
B MAX	0.390 (9,9)	0.870 (22,1)	0.920 (23,4)	1.070 (27,2)	1.100 (28,0)	1.290 (32,8)	1.440 (36,6)	2.090 (53,1)
C NOM	0.250 (6,4)	0.250 (6,4)	0.250 (6,4)	0.265 (6,7)	0.350 (8,9)	0.550 (14,0)	0.550 (14,0)	0.550 (14,0)

NOTE: Each pin centerline is located within 0.010 (2.54) of its true longitudinal position.
All linear dimensions are in inches and (millimeters).